EUROfusion Integrated Modelling workflows Documentation

Release 3.0

WPCD

Feb 23, 2019

CONTENTS

Contents:

CHAPTER

ONE

INTRODUCTION TO CODE DEVELOPMENT FOR INTEGRATED MODELLING

The EUROfusion Project on Code Development for Integrated Modelling (WP-CD) supports the achievement of the European Fusion Roadmap at Horizon 2020 goals, via the development of existing modelling codes with a particular focus on integrated modelling. The primary objectives of WPCD are: 1. Provide a suite of codes that can be validated on existing machines and used for JT-60SA, ITER and DEMO predictions:

- build on the existing modelling codes developed by the EUROfusion Consortium members including the Integrated Modelling (EU-IM) infrastructure, toolset and codes developed under the former EFDA ITM Task Force,
- add new physics to the existing models
- couple codes into integrated workflows
- optimize codes.

2. Specific ITER simulation work in support of ITER IO and F4E with specified deliverables. WPCD operates under a work plan aiming to provide in the long term a full suite of integrated simulation workflows, incorporating core-edge-SOL/PFC coupling, first-principles models and control elements. A central task is the development of the new modular European Transport Simulator, ETS, which is being deployed to JET modelling infrastructure for validation and application to experimental analysis. In addition to code and workflow development, rigorous code verification is also performed under WPCD, within the EU-IM framework; whereas validation of the released integrated modelling workflows against the experiments is performed under the rerelated Task Forces. EUROfusion WPCD webpage. The EU-IM Team includes both EUROfusion WPCD and WPISA CPT contributors, see EU-IM Team. This list reproduces the status of members in 2015 and is possibly not exhaustive.

1.1 The European Integrated Modelling (EU-IM) approach

The choice of Integrated Modelling made by the former EFDA ITM and pursued now under EUROfusion WPCD is unique and original: it entails the development of a comprehensive and completely generic tokamak simulator including both the physics and the machine, which can be applied for any fusion device. The simulation platform was designed to be fully modular, flexible, and independent of a programming language.The choice of modularity implies that each module contains a single physical model and that the communication between the modules is standardised: a set of common common rules (ontology) clearly specify the format of the data to be consistently exchanged between modules (data-structure). The complexity of coupling the codes together is therefore transferred to the definition of a generic data-structure (allowing to describe and exchange information concerning both physical quantities and technical objects, not assuming the origin of those), extensible to allow the integration of

new physics, as well as more elaborate machine geometries and experimental data in the future. A central project is the development of the so-called **European Transport Simulator (ETS)** aimed to meet all the EU-IM requirements, namely modularity, flexibility and standardized interfaces. In terms of the physics, the ETS is designed to solve the standard set of one-dimensional time dependent equations which describe the evolution of the core plasma. The solver itself is designed with a modular approach enabling the separation of the physics from the numerics, thereby facilitating the testing/usage of the numerical schemes that best suit a particular physical simulation.

1.2 Mission

The EU-IM operated under EFDA from 2004 until 2013. The main mission of EU-IM was to provide a software infrastructure framework for EU integrated modelling activities as well as a validated suite of simulation codes for the modelling of present experiments, ITER and DEMO plasmas. The EU-IM operated until 2013 under a work programme formulated to support this goal, structuring the EU modelling effort around existing experiments and ITER scenario prediction while maintaining a long term strategic aim to provide a validated set of European modelling tools for ITER exploitation.

1.3 Achievements

During the first phase of the EU-IM, surveys and cross-verification of the available European models and numerical codes were performed within the individual IMPs and the data-structure was extensively discussed and defined. Equilibrium, linear MHD stability, core transport and RF wave propagation, as well as the poloidal field systems and a few diagnostics were the first topics addressed. Data structures have been finalised for these and then expanded to address, among others, non-linear MHD, edge physics, turbulence and neutral beam propagation. In parallel to the development of the physics concepts, EU-IM developed the tools to manipulate the data structure and use it in fully flexible and modular simulation workflows. The EU-IM database contains machine descriptions from JET, Tore Supra, MAST, FTU, FAST, AUG, ITER as well as some experimental data from Tore Supra and JET. The EU-IM futher achieved the development of the first release version of a fully modular and versatile simulator, the ETS, with all the essential functionalities. The validation of the ETS simualtor started in 2010 against the state-of-the-art transport codes and ETS now starts to be used for the first physics applications. Next steps are the validation of the simulator for a complete discharge on existing experimental data with the available modules, the integration of more quantitative physics models (“ab-initio”) and the integration of the whole modelling of the device. Some posters that describe the EU-IM were presented at an ITM EXPO at the 2011 EPS fusion conference in Strasbourg.

1.4 Structure

The EFDA ITM was structured into four Integrated Modelling Projects (IMPs) focusing on the following physics areas:

- **IMP12** plasma equilibrium and MHD
- **IMP3** transport code and whole discharge evolution
- **IMP4** transport and micro-instabilities
- **IMP5** heating, current drive (H&CD) and fast particles

The "Infrastructure and Software Integration Project" (**ISIP**) was in charge of developing, maintaining and operating the code platform structure and implementing the EU-IM data-structure. A key function of ISIP was to provide infrastructure support to the IMPs. At present, under EUROfusion, this expertise and tasks are ensured by the **Core Programming Team** under the **Infrastructure and Support Activities Work Package (WP-ISA CPT)**. Two further projects ensured the link with the experimentalists and the provision of the experimental databases:

- **AMNS** the "Atomic, Molecular, Nuclear Surface" Data
- **EDRG** "Experimentalists and Diagnosticians Resource Group"

The "ITER Scenario Modelling Working Group" (**ISM**) was established in 2007 as part of EU-IM with the aim to assist in systematic predictive modelling of all ITER reference scenarios by using the major existing integrated modelling tools, whilst the EU-IM code platform was in development. ISM also supported the verification and validation of the ETS, which aims to become the main tool for EU modelling activity.

1.5 Contributors

EU-IM contributors are defined in the Appendix of G.L. Falchetto et al., Nuclear Fusion 54,043018, 2014. This list reproduces the status of EU-IM members in 2012 and is not exhaustive. A grateful thank you to all those who contributed and promoted EU-IM since its beginnigs.

1.6 Glossary

Collaborative Development Environment (CDE) A **collaborative development environment (CDE)** is an online meeting space where a software development project's stakeholders can work together, no matter what timezone or region they are in, to discuss, document , and produce project deliverables. The name was coined by Grady Booch.

Consistent Physical Object (CPO) A Consistent Physical Object (CPO) is a physics based, hierarchical data structure employed by the EU-IM for a complete description of a physics area, e.g. equilibrium. All EU-IM code modules interact through the exchange of CPOs. The CPOs also form the basic block of data written to the EU-IM database.

Content Management System (CMS) A **content management system (CMS)** is the collection of procedures used to manage work flow in a collaborative environment. These procedures can be manual or computer-based. The procedures are designed to:

- Allow for a large number of people to contribute to and share stored data
- Control access to data, based on user roles. User roles define what information each user can view or edit
- Aid in easy storage and retrieval of data
- Reduce repetitive duplicate input
- Improve the ease of report writing
- Improve communication between usersq

In a CMS, data can be defined as nearly anything - documents, movies, pictures, phone numbers, scientific data, etc. CMSs are frequently used for storing, controlling, revising, semantically enriching, and publishing documentation.

FC2K FC2K is a tool for wrapping a Fortran or C++ source code into a Kepler actor. Before using it, your physics code should be EU-IM-compliant (i.e. use CPOs as input/output).

Gforge Gforge hosts all projects (software and infrastructure) under the EU-IM.

EU-IM Gateway The EU-IM Gateway is a compute cluster located at Portici (near Napoli in Italy). It is uses for development and fusion simulations in the EU-IM.

EU-IM Portal The EU-IM Portal is the web portal for the EU-IM, i.e. it hosts the EU-IM web pages and projects under Gforge.

Integrated Simulation Editor (ISE) The Integrated Simulation Editor ISE allows you to visualize and edit data from an EU-IM database entry. It also allows running a Kepler workflow based on the opened data entry.

Universal Access Layer (UAL) The UAL (Universal Access Layer) is a multi-language library that allows exchanging Consistent Physical Objects (CPOs) between various modules, and to write to an EU-IM database.

actor Actors take execution instructions from a director. In other words, actors specify what processing occurs while the director specifies when it occurs. In the EU-IM, actors are usually modules which contain physics codes like EQUAL or HELENA.

calibration The process of adjusting numerical or physical modelling parameters in the computational model for the purpose of improving agreement with experimental data.

data mapping An XML file containing all the mapping essentials for mapping from a local experimental database for a specific tokamak device to the EU-IM database. The mapping essentials include for instance the download method, local signal names, gains and offsets, time base, and eventual interpolation option to ensure that only one time base is set for each CPO that is built from multiple local signals. A java code (exp2ITM developed under ISIP), with the MD and DM files as inputs, is then run to connect to the local device database, retrieve the required experimental data and populate the EU-IM database instance for that shot/device and dataversion.

director A director controls (or directs) the execution of a workflow, just as a film director oversees a cast and crew.

error A recognisable deficiency in any phase or activity of modelling and simulation that is not due to lack of knowledge.

kepler Kepler is a software application for the analysis and modeling of scientific data. Kepler simplifies the effort required to create executable models by using a visual representation of these processes. These representations, or "scientific workflows", display the flow of data among discrete analysis and modeling components.

machine description The machine description (MD) of a device basically builds on the set of engineering and diagnostic settings characterising a tokamak device. This includes, for instance, the vessel/limiter description, the PF coils and circuiting and lines of sight of diagnostics. In practice, all MD information is encapsulated in an XML file that emanates from the MD tagged datastructure schemas. An MD instance of a given device is then stored into the EU-IM database as shot 0 for that device database.

model A representation of a physical system or process intended to enhance our ability to understand, predict, or control its behaviour.

- A **conceptual model** consists of the observations, mathematical modelling data, and mathematical (e.g., partial differential) equations that describe the physical system. It will also include initial and boundary conditions.

- The **computational model** is the computer program or code that implements the conceptual model. It includes the algorithms and iterative strategies. Parameters for the computational model include the number of grid points, algorithm inputs, and similar parameters, etc.

modelling The process of construction or modification of a model

prediction Use of a model to foretell the state of a physical system under conditions for which the model has not been validated.

simulation The exercise or use of a model.

uncertainty A potential deficiency in any phase or activity of the modelling process that is due to the lack of knowledge.

validation The process of determining the degree to which a model is an accurate representation of the real world form the perspective of the intended uses of the model.

verification The process of determining that a model implementation accurately represents the developer's conceptual description of the model and the solution to the model.

1.7 Support

1.7.1 Getting support for the EU-IM platform and Gateway

The EU-IM provides several ways to get support when you run into problems. Which one to choose depends on the nature of your problem. This page tries to give an overview.

1.7.2 Support for problems related to the EU-IM Gateway

The official documentation of the ITM Gateway can be found at https://wiki.eufus.eu.

1.7.3 Support for problems related to the EU-IM Platform and Software

All ITM-specific software and the whole ITM platform is supported by the Core Programming Team (CPT). You can submit trouble tickets to them via the General Support Project in the GForge system. To get more effective help, have a look at the guidelines prepared here: How to report an issue.

To directly submit a trouble ticket, go to: General Support Tracker (https://gforge6.eufus.eu/gf/project/generalsupport/tracker/).

Use this support tracker if your problem falls in the following categories:

- Problems using the UAL, FC2K, HPC2K or similar tools
- Problems running Kepler or Kepler workflows
- Visualization tools: VisIt, Python
- Integrated Simulation Editor (ISE)
- Any software project that is hosted in GForge
- Any kind of scientific software

1.7.4 Feature requests for EU-IM Software

Feature requests for software developed within the ITM can be submitted to a separate tracker.

To submit a feature request, please go to General Feature Request Tracker.

If you are unsure whether to file a bug report of feature request, have a look at these guidlines: How to report an issue.

1.8 Links to related external projects

- EUFORIA Project
- MAPPER Project
- EFDA High Level Support Team (HLST)
- EFDA Goal Oriented Training in Theory (GOTiT)

CHAPTER

TWO

INFRASTRUCTURE

The part of the EU-IM documentation addresses infrastructure related issues like the EU-IM Gateway, EU-IM database access, the Universal Access Layer (UAL), and useful tools for handling CPOs.

2.1 Universal Access Layer (UAL)

The UAL (Universal Access Layer) is a multi-language library that allows exchanging Consistent Physical Objects (CPOs) between various modules, and to write to an EU-IM database. The documentation here is provided for rather experienced users who want to practice the UAL in their test programs. Regular KEPLER users do not need to know anything about the UAL. KEPLER manages transparently the UAL calls, which are embedded in the physics code wrappers. No UAL calls should be made inside physics modules. Prior using the UAL, the environment must be configured. It is recommended to use the `EU-IMv1` script for this, which simultanesouly sets i) the database environment (to the private database of the user) ii) the UAL libraries environment iii) the Kepler environment.

```
source $EU-IMSCRIPTDIR/EU-IMv1 KEPLERFOLDER
MACHINENAME DATAVERSION
```

e.g.:

```
source $EU-IMSCRIPTDIR/EU-IMv1 kepler tore_supra 4.08a
```

This scripts does not prevent you from using databases from other users or the public one, you must then use the UAL function `euitm_open_env` in your program to do so. The EU-IMv1 script uses the two following scripts: to set the database environment variables (mandatory prior UAL usage), use:

```
source $EU-IMSCRIPTDIR/set_itm_data_env USERNAME
 MACHINENAME DATAVERSION
```

e.g.:

```
source $EU-IMSCRIPTDIR/set_itm_data_env myname jet 4.08a
```

Then to set the path to the right UAL libraries, use:

```
source $EU-IMSCRIPTDIR/set_itm_env DATAVERSION
```

e.g.:

```
source $EU-IMSCRIPTDIR/set_itm_env 4.08a
```

UAL libraries are installed in /afs.eufus.eu/isip/project/switm/ual. The source code is stored in a subversion repository in /afs/edfa-itm.eu/isip/project/portal/gforge/storage/svnroot/ual. To check out a subversion working copy of the repository, storing it in subdirectory *ual*, do

```
svn co https://gforge6.eufus.eu/svn/ual
```

2.2 Handling CPOs

The tools below allow you to perform useful operations on the EU-IM datastructures including entire CPOs in your Fortran90 workflows. They are not meant as a replacement of the Universal Access Layer (UAL) but rather as a complement especially in situations where the UAL is not available or not practical, e.g. in U.S. - EFDA collaborations.

2.2.1 Module deallocate_stuctures

This Fortran90 module allows you to deallocate in a very simple way any EU-IM data structure which is defined in euitm_schemas.f90. Add the following use statement to your code:

```
use deallocate_structures
```

The Fortran syntax for deallocating a cpo is then:

```
call deallocate_cpo(cpo)
```

where cpo is a single time slice or a pointer array of a CPO (or other EU-IM data structure). With

```
call set_deallocate_verbosity(verbosity)
```

you can set a verbosity level for the deallocate routines. verbosity = 0 produces no output, whereas verbosity > 0 produces verbose output. The module deallocate_structures.f90 is hosted by the Gforge project itmshared. Check it out with

```
svn checkout https://gforge6.eufus.eu/svn/itmshared/branches/tools target_dir
```

Two static libraries libdeallocate_pgi.a and libdeallocate_g95.a have been prebuilt on the EU-IM Gateway .

2.2.2 Module copy_structures

This Fortran90 module allows you to copy in a very simple way any EU-IM data structure which is defined in euitm_schemas.f90. Add the following use statement to your code:

```
call set_deallocate_verbosity(verbosity)
```

you can set a verbosity level for the deallocate routines. verbosity = 0 produces no output, whereas verbosity > 0 produces verbose output. The module deallocate_structures.f90 is hosted by the Gforge project itmshared. Check it out with

```
use copy_structures
```

The Fortran syntax for copying a cpo is then:

```
call copy_cpo(cpo_source, cpo_target)
```

where cpo_source and cpo_target are single time slices or arrays of a CPO (or other EU-IM data structure) of the same derived type, real scalars, or real arrays (1D - 7D). The allocation of the elements of the target structure is done automatically. With

```
call set_copy_verbosity(verbosity)
```

you can set a verbosity level for the copy routines. verbosity = 0 produces no output, whereas verbosity > 0 produces verbose output. The module copy_structures.f90 is hosted by the Gforge project itmshared . Check it out with

```
svn checkout https://gforge6.eufus.eu/svn/itmshared/branches/tools target_dir
```

Two static libraries libcopy_pgi.a and libcopy_g95.a have been prebuilt on the EU-IM Gateway .

2.2.3 Module euitm_copy

This Fortran90 module allows you to copy in a very simple way any EU-IM data structure which is defined in euitm_schemas.f90 including entire trim traces. Add the following use statement to your code:

```
use euitm_copy
```

The Fortran syntax for copying a cpo via assignment is then:

```
cpo_target = cpo_source
```

where cpo_source and cpo_target are single time slices or arrays of a CPO (or other EU-IM data structure) of the same derived type. The allocation of the elements of the target structure is done automatically. The program files are hosted by the Gforge project itmshared . Check them out with

```
svn checkout https://gforge6.eufus.eu/svn/itmshared/branches/perlcopy target_dir
```

To build the Fortran90 module, run

```
creacopy.pl
```

It takes euitm_schemas.f90 from the directory $UAL/fortraninterface/ . Or supply the file from a given directory

```
creacopy.pl $MYDIR/euitm_schemas.f90
```

2.2.4 Module is_set_structures

This Fortran90 module can be used to check whether EU-IM data structures including entire CPOs have been set. The subroutines in is_set_structures.f90 write out the name of each element in the data structure together with 'T' if it has been set or 'F' if not. Add the following use statement to your code:

```
use is_set_structures
```

The Fortran syntax for checking a cpo is then:

```
call is_set_cpo(cpo, "name of cpo")
```

where cpo is a single time slice or an array of a CPO (or other EU-IM data structure) and "name of cpo" is a string containing the name of the CPO. The module is_set_structures.f90 is hosted by the Gforge project itmshared . Check it out with

```
svn checkout https://gforge6.eufus.eu/svn/itmshared/branches/tools target_dir
```

Two static libraries libis_set_pgi.a and libis_set_g95.a have been prebuilt on the EU-IM Gateway .

2.2.5 Module size_of_structures

The subroutines in size_of_structures.f90 write out the name of each element in the data structure with its size, the size of each entire substructure, and the size of the entire CPO. The size can be given in bytes or in a more human friendly format depending on the value of the logical parameter human_readable . The indentation is done in steps of 2 blanks with an initial indentation of 1. The maximum depth to which the results are displayed is specified by a call to set_size_of_maxlevel . Output of empty fields can be suppressed by setting the verbosity to zero with a call to set_size_of_verbosity . In all cases, sums are carried out over all levels. Add the following use statement to your code:

```
use size_of_structures
```

The Fortran syntax for calculating the size of a cpo is then:

```
call size_of_cpo(cpo, total_size, human_readable, "name of cpo")
```

where cpo is a single time slice or an array of a CPO (or other EU-IM data structure) and "name of cpo" is a string containing the name of the CPO. total_size is an integer and should be set to zero before the call. human_readable is a flag (true => human friendly format). Set the verbosity with:

```
call set_size_of_verbosity(verbosity)
```

verbosity = 0 => no output of empty fields verbosity > 0 => full output Set the maximum depth with:

```
call set_size_of_maxlevel(level)
```

with level being an integer. The module size_of_structures.f90 is hosted by the Gforge project itmshared . Check it out with

```
svn checkout https://gforge6.eufus.eu/svn/itmshared/branches/tools target_dir
```

Two static libraries libsize_of_pgi.a and libsize_of_g95.a have been prebuilt on the EU-IM Gateway.

2.2.6 Module write_structures

This Fortran90 module can be used to write EU-IM data structures including entire CPOs to disk. The corresponding file is opened with

```
call open_write_file(unit_no, file_name)
```

where unit_no is the file handle (integer) and file_name a string with the file name (possibly including the path). The file is closed with

```
call close_write_file
```

Add the following use statement to your code:

```
use write_structures
```

The Fortran syntax for writing a cpo to disk is then:

```
call write_cpo(cpo, "name of cpo")
```

where cpo is a single time slice or an array of a CPO (or other EU-IM data structure) and "name of cpo" is a string containing the name of the CPO. With

```
call set_write_verbosity(verbosity)
```

you can set a verbosity level for the write routines. verbosity = 0 produces no output, whereas verbosity > 0 produces verbose output. The module write_structures.f90 is hosted by the Gforge project itmshared . Check it out with

```
svn checkout https://gforge6.eufus.eu/svn/itmshared/branches/tools target_dir
```

Two static libraries libwrite_pgi.a and libwrite_g95.a have been prebuilt on the EU-IM Gateway .

2.2.7 Module read_structures

This Fortran90 module can be used to read EU-IM data structures including entire CPOs from disk. The corresponding file is opened with

```
call open_read_file(unit_no, file_name)
```

where unit_no is the file handle (integer) and file_name a string with the file name (possibly including the path). The file is closed with

```
call close_read_file
```

Add the following use statement to your code:

```
use read_structures
```

The Fortran syntax for reading a cpo from disk is then:

call read_cpo(cpo, "name of cpo")

where cpo is a single time slice or an array of a CPO (or other EU-IM data structure) and "name of cpo" is a string containing the name of the CPO. The module automatically deallocates any fields already allocated in cpo and allocates all required fields automatically. It is absolutely essential that "name of cpo" is identical with the one chosen when the cpo was written. With

```
call set_read_verbosity(verbosity)
```

you can set a verbosity level for the read routines. verbosity = 0 produces no output, whereas verbosity > 0 produces verbose output. The module read_structures.f90 is hosted by the Gforge project itmshared . Check it out with

```
svn checkout https://gforge6.eufus.eu/svn/itmshared/branches/tools target_dir
```

Two static libraries libread_pgi.a and libread_g95.a have been prebuilt on the EU-IM Gateway .

2.2.8 Module diff_structures

This Fortran90 module can be used to compare two CPOs or other EU-IM data structures. It was developed to facilitate benchmarks and automated test suites for the code development. It was kept flexible through the use of function arguments in the argument list of the subroutines of diff_structures. This allows the user to specify his own function set for the analysis and evalutation of the differences between the two CPOs. A call to diff_cpo simply writes out the result of this user defined function. Add the following use statements to your code:

```
use diff_structures
use error_analysis
```

The Fortran syntax for calculating the differences between two cpos is then:

```
call diff_cpo(reference_cpo, test_cpo, name_root, func)
```

where reference_cpo is the reference CPO or other EU-IM data structure and test_cpo is the test CPO or other EU-IM data structure. name_root is a string which defines the root of the field names to be displayed, e.g. 'equilibrium'. func is a function argument to the subroutine diff_cpo. It can be any user defined function with the following constraints:

- It must be defined inside the module error_analysis (an example version with various error analysis functions is provided in error_analysis.f90).
- It follows the structure (dummy arguments, interface, overloading) as demonstrated in error_analysis.f90. The function always has a header function with a list of optional dummy arguments. Depending on which actual arguments are specified, this functions calls the overloaded function with the correct arguments. The interim function is required because of Fortran90/95 limitations. The actual error analysis is carried out inside the overloaded functions. Two fields of these functions are intent(inout) variables:

```
   diff_counter : to count the number of difference
   error_level  : to allow for sums or averages over entire CPOs (see examples)

These two variables are private to the error_analysis module.
To access them please use the functions
```

```
     get_diff_counter()

and

.. code-block:: fortran

     get_error_level()

The function

.. code-block:: fortran

     set_error_level(err_level)

may be used to specify an initial value for the variable
error_level
.
```

With

```
call set_diff_verbosity(verbosity)
```

you can set a verbosity level for the diff routines. verbosity = 0 produces no output, whereas verbosity > 0 produces verbose output. The file check_equilibrium.f90 represents a simple example for a program to

compare two equilibrium CPOs one of which is used as a reference for test cases in code development. It clearly demonstrates the use of the diff_structures module. The module diff_structures.f90 and the auxiliary file error_analysis.f90 and check_equilibrium.f90 are hosted by the Gforge project itmshared . Check them out with

```
svn checkout https://gforge6.eufus.eu/svn/itmshared/branches/tools target_dir
```

Two static libraries libdiff_pgi.a and libdiff_g95.a have been prebuilt on the EU-IM Gateway .

CHAPTER

THREE

CORE TRANSPORT SIMULATOR (ETS)

3.1 ETS source in FORTRAN

You can checkout the FORTRAN ETS workflow from gforge / project ETS following instructions from ETS User Guide

If you did not use ETS before, first you need to request access to the code via the EFDA EU-IM Portal by following the GForge tab, following the project ETS and requesting access.

Once you have access to the code, it can be checked out of SVN using

```
svn co https://gforge6.eufus.eu/svn/ets
```

to access the whole repository, or

```
svn co https://gforge6.eufus.eu/svn/ets/trunk/ETS
```

to access just the trunk version of the ETS.

The ETS project on Gforge also includes:

- A wiki
- Some documentation
- Trackers
- News
- Mailing lists
- The SVN repository (web interface)

3.2 Documentation for the ETS

- Current ETS Timeline (PDF)(MS Project)
- Description of the ETS
- Form of the standardize equations
- ETS User Guide
- ETS Status
- ETS Doxygen Documentation (PDF)(HTML)

- Pellets in ETS

3.3 Presentations that discuss the ETS

- Presentation at ICNSP-2009 on the ETS
- Movie from the presentation showing the evolution of the flux surfaces
- Movie from the presentation showing the evolution of the plasma

3.4 ETS Verification & Validation

3.4.1 Roadmap for the ETS verification and benchmarking procedure (based on G. Pereverzev's proposal)

- Proposal for ETS verification and benchmarking procedure (PDF)
- ETS verification and benchmarking (ASTRA results) (PDF)

3.4.2 Part I. Cylindrical geometry. Consistency check.

3.4.3 Results obtained in 2011 for UAL-version 4.08b

NOTE: For the particle diffusion coefficient D and pinch velocity v, the values below correspond in fact to the individual coefficients for each of the three transport models considered within ETS. For example, a v=1 translates into an effective pinch velocity of 3 in the density transport equation and of 0+3/2+5/2=4 in the ion flux contributing to heat transport.

Case	Description	Subcase	Solver	Comments	Graphic output (PDF)
I.1.1	Constant densities and temperatures	N/A	3	Convergence ok. Errors within machine precision	I.1.1_s3
			7	Convergence ok. Errors within machine precision	I.1.1_s7

Continued on next page

Table 3.1 – continued from previous page

Case	Description	Subcase	Solver	Comments	Graphic output (PDF)
			10	Convergence ok. Errors within machine precision	I.1.1_s10
I.1.2	Interpretative time-dependent solution for densities	N/A	3	Under analysis: no time evolution for the temperatures	I.1.2_s3
I.1.3	Conservation laws(particle and energy) with diffusion	N/A	N/A	This case was treated under I.1.5	N/A
I.1.4	Conservation laws(particle and energy) with discontinuous diffusion coefficient	N/A	3	Convergence ok.	I.1.4_s3
			7	Convergence ok.	I.1.4_s7
			10	Failed to initiate	

Continued on next page

Table 3.1 – continued from previous page

Case	Description	Subcase	Solver	Comments	Graphic output (PDF)
I.1.5	Conservation laws(particle and energy) with diffusion and convection	D=0.1 and v=0	3	Convergence in density reached; asymptotic solution (for density) reached; convergence in temperature never fully reached	I.1.5_s3_101pts_1e-3s I.1.5_s3_51pts_1e-3s I.1.5_s3_101pts_1e-2s
			7		I.1.5_s7_101pts_1e-3s I.1.5_s7_51pts_1e-3s I.1.5_s7_101pts_1e-2s
			10		
		I.1.5.a D=0.1 and v=0.1	3	Solvers 3 and 7: Convergence in temperature reached; convergence in density reached; asymptotic solution never reached (after 50s and 100s). Solver 10: does not converge (number of iterations exceeds 1000).	I.1.5.a_s3_101pts_1e-3s I.1.5.a_s3_51pts_1e-3s I.1.5.a_s3_101pts_1e-2s

Continued on next page

Table 3.1 – continued from previous page

Case	Description	Subcase	Solver	Comments	Graphic output (PDF)
			7		I.1.5.a_s7_101pts_1e-3s
			10		

Continued on next page

Table 3.1 – continued from previous page

Case	Description	Subcase	Solver	Comments	Graphic output (PDF)
		I.1.5.b D=0.1 and v=0.3	3	No convergence for density and temperature (worse for density); deviation from asymptotic solution increases with time. Errors are similar for solvers 3 and 7; the latter reduces the energy error with time, at short time-steps. Increasing number of points reduces deviation from asymptotic solution (tested for solver 3)	I.1.5.b_s3_101pts_1e-3s I.1.5.b_s3_51pts_1e-3s I.1.5.b_s3_101pts_1e-2s

Continued on next page

Table 3.1 – continued from previous page

Case	Description	Subcase	Solver	Comments	Graphic output (PDF)
			7	Neither solver yields zero steady-state particle- and energy- flux. The on-axis behavior of Te is different for solvers 3 and 7. Convergence proceeds through decreasing (solver 3 and solver 7 at long time-steps) and increasing (solver 7 at short time-steps) fluxes near the wall	I.1.5.b_s7_101pts_1e-3s I.1.5.b_s7_51pts_1e-3s I.1.5.b_s7_101pts_1e-2s
			10	Failed to initiate	

Continued on next page

Table 3.1 – continued from previous page

Case	Description	Subcase	Solver	Comments	Graphic output (PDF)
		I.1.5.c D=0.1 and v=1.0	3	Solvers 3 and 7: For v>1 the number of iterations exceeds the max value of 1000, apparently because ETS doesn't conserve the number of particles. The situation is mitigated (not solved) by increasing Np and decreasing dt. Solver 10 fails to initiate.	I.1.5.c_s3_101pts_1e-3s I.1.5.c_s3_51pts_1e-3s I.1.5.c_s3_101pts_1e-2s
			7		I.1.5.c_s7_101pts_1e-3s I.1.5.c_s7_101pts_1e-2s
			10		
		I.1.5.d D=0.1 and v=2.0	3		I.1.5.d_s3_501pts_1e-3s I.1.5.d_s3_501pts_1e-4s
			7		

Continued on next page

Table 3.1 – continued from previous page

Case	Description	Subcase	Solver	Comments	Graphic output (PDF)
			10	Failed to initiate	
		I.1.5.h D=0.1 and v=-0.1	3	Solvers 3 and 7: Convergence in density and temperature reached; asymptotic solution (for density) almost always reached; Solver 10: fails to converge (number of iterations exceeds 1000).	I.1.5.h_s3_101pts_1e-3s I.1.5.h_s3_51pts_1e-3s I.1.5.h_s3_101pts_1e-2s
			7		I.1.5.h_s7_101pts_1e-3s I.1.5.h_s7_101pts_1e-2s
			10		
		I.1.5.i D=0.1 and v=-0.3	3	Convergence in density reached; convergence in temperature never fully reached; asymptotic solution never fully reached	I.1.5.i_s3_101pts_1e-3s I.1.5.i_s3_51pts_1e-3s I.1.5.i_s3_101pts_1e-2s

Continued on next page

Table 3.1 – continued from previous page

Case	Description	Subcase	Solver	Comments	Graphic output (PDF)
			7		I.1.5.i_s7_101pts_1e-3s I.1.5.i_s7_51pts_1e-3s
			10		
		I.1.5.j D=0.1 and v=-1.0	3	Solvers 3, 7 and 10: fail to converge (number of iterations exceeds 1000) long before the total execution time of 4s (at best they go until 1.8s)	I.1.5.j_s3_101pts_1e-3s I.1.5.j_s3_51pts_1e-3s I.1.5.j_s3_101pts_1e-2s
			7		I.1.5.j_s7_101pts_1e-3s I.1.5.j_s7_101pts_1e-2s
			10		I.1.5.j_s10_101pts_1e-3s I.1.5.j_s10_51pts_1e-3s

Continued on next page

Table 3.1 – continued from previous page

Case	Description	Subcase	Solver	Comments	Graphic output (PDF)
		I.1.5.k D=0.1 and v=-2.0	3	No convergence for both density and temperature (worse for density); deviation from asymptotic solution increases with time deviation from	I.1.5.d_s3_501pts_1e-3s
			7		
			10		

3.4.4 Results obtained in 2012 for UAL-version 4.09a

NOTE: Solver 4 is the one with the best performance. Solvers 3, 7 and 10 are to be disregarded in the future. ALL solvers fail to converge (i.e. demanding more than 1000 iterations) for convection v >= 1 m/s.

Case	Description	Subcase	Solver	Comments	Graphic output (PDF)
I.1.1	Constant densities and temperatures	N/A	3	Convergence ok. Errors within machine precision, except for the on-axis value	I.1.1_101pts_1e-2s

Continued on next page

Table 3.2 – continued from previous page

Case	Description	Subcase	Solver	Comments	Graphic output (PDF)
			4	Convergence ok. Errors within machine precision, except for the on-axis value	
			7	NANs found	
			10	Convergence ok. Errors within machine precision	
I.1.2	Interpretative time-dependent solution for densities (D=0 and v=0)	No external sources; no internal sources; Te = Ti @ t=0 (no plasma collision source)	4	No time evolution for the pressure in the absence of all sources	I.1.2_101pts_1e-2s

Continued on next page

Table 3.2 – continued from previous page

Case	Description	Subcase	Solver	Comments	Graphic output (PDF)
		I.1.2.a No external sources; internal (convection) sources limited to 10%; Te = Ti @ t=0 (no plasma collision source)	4	Time evolution for the pressure due to internal sources	I.1.2.a_101pts_1e-2s
		I.1.2.b No external sources; no internal sources; Te != Ti @ t=0 plasma collision source)	4	Pressure evolves to constant values with time, due to e-i energy exchange	I.1.2.b_101pts_1e-2s
I.1.3	Conservation laws(particle and energy) with diffusion	N/A	N/A	This case was treated under I.1.5	N/A
I.1.4	Conservation laws(particle and energy) with discontinuous diffusion coefficient	N/A	3	Convergence obtained; asymptotic solution (for density) reached.	I.1.4_101pts_1e-2s

Continued on next page

Table 3.2 – continued from previous page

Case	Description	Subcase	Solver	Comments	Graphic output (PDF)
			4	Convergence obtained; asymptotic solution (for density) reached.	
			7	Convergence obtained, with a problem on the axis.	
			10	Convergence obtained, but for a very different asymptotic solution (for density).	
I.1.5	Conservation laws(particle and energy) with diffusion and convection (D in m2/s; v in m/2)	D=0.1 and v=0	3	Convergence obtained; asymptotic solution (for density) reached; conservation laws poorly satisfied.	I.1.5_101pts_1e-3s I.1.5_51pts_1e-3s I.1.5_101pts_1e-2s
			4	Convergence obtained; asymptotic solution (for density) reached.	

Continued on next page

Table 3.2 – continued from previous page

Case	Description	Subcase	Solver	Comments	Graphic output (PDF)
			7	Convergence obtained, with a poor prediction of the asymptotic (density) solution and with a problem on the axis.	
			10	Failed to converge (the number of iterations exceeds the max value of 1000).	
		I.1.5.a D=0.1 and v=0.1	3	Convergence obtained; asymptotic solution (for density) reached; conservation laws poorly satisfied.	I.1.5.a_101pts_1e-3s I.1.5.a_51pts_1e-3s I.1.5.a_101pts_1e-2s
			4	Convergence obtained; asymptotic solution (for density) reached.	

Continued on next page

Table 3.2 – continued from previous page

Case	Description	Subcase	Solver	Comments	Graphic output (PDF)
			7	Convergence obtained for 101 points only, with a poor prediction of the asymptotic (density) solution and with a problem on the axis; conservation laws poorly satisfied.	
			10	Convergence obtained, but for a very different asymptotic solution (for density); conservation laws not satisfied.	
		I.1.5.b D=0.1 and v=0.3	3	Convergence obtained; asymptotic solution (for density) reached; conservation laws poorly satisfied.	I.1.5.b_s3_101pts_1e-3s I.1.5.b_s3_51pts_1e-3s I.1.5.b_s3_101pts_1e-2s

Continued on next page

Table 3.2 – continued from previous page

Case	Description	Subcase	Solver	Comments	Graphic output (PDF)
			4	Convergence obtained; asymptotic solution (for density) reached.	
			7	Convergence obtained for 1e-2s points only, with a poor prediction of the asymptotic (density) solution and with a problem on the axis; conservation laws poorly satisfied.	
			10	Convergence obtained, but for a very different asymptotic solution (for density); conservation laws not satisfied.	

Continued on next page

Table 3.2 – continued from previous page

Case	Description	Subcase	Solver	Comments	Graphic output (PDF)
		I.1.5.c D=0.1 and v=1.0	3	Failed to converge (the number of iterations exceeds the max value of 1000).	I.1.5.c_101pts_1e-3s I.1.5.c_51pts_1e-3s I.1.5.c_101pts_1e-2s
			4	Failed to converge (the number of iterations exceeds the max value of satisfied.	
			7	Failed to converge (the number of iterations exceeds the max value of satisfied.	
			10	Failed to converge (the number of iterations exceeds the max value of satisfied.	

Continued on next page

Table 3.2 – continued from previous page

Case	Description	Subcase	Solver	Comments	Graphic output (PDF)
		I.1.5.h D=0.1 and v=-0.1	3	Convergence obtained; asymptotic solution (for density) reached.	I.1.5.h_101pts_1e-3s I.1.5.h_51pts_1e-3s I.1.5.h_101pts_1e-2s
			4	Convergence obtained; asymptotic solution (for density) reached.	
			7	Convergence obtained, with a poor prediction of the asymptotic (density) solution and with a problem on the axis; conservation laws poorly satisfied.	
			10	Convergence obtained, but for a very different asymptotic solution (for density); conservation laws not satisfied.	

Continued on next page

Table 3.2 – continued from previous page

Case	Description	Subcase	Solver	Comments	Graphic output (PDF)
		I.1.5.i D=0.1 and v=-0.3	3	Convergence obtained; asymptotic solution (for density) reached.	I.1.5.i_101pts_1e-3s I.1.5.i_51pts_1e-3s I.1.5.i_101pts_1e-2s
			4	Convergence obtained; asymptotic solution (for density) reached.	
			7	Convergence obtained for 1e-2s only, with a poor prediction of the asymptotic (density) solution and with a problem on the axis; conservation laws poorly satisfied.	

Continued on next page

Table 3.2 – continued from previous page

Case	Description	Subcase	Solver	Comments	Graphic output (PDF)
			10	Convergence obtained for 1e-3s only, but for a very different asymptotic solution (for density); conservation laws not satisfied.	
		I.1.5.j D=0.1 and v=-1.0	3	Failed to converge (the number of iterations exceeds the max value of 1000).	I.1.5.j_101pts_1e-3s I.1.5.j_51pts_1e-3s I.1.5.j_101pts_1e-2s
			4	Failed to converge (the number of iterations exceeds the max value of satisfied.	

Continued on next page

Table 3.2 – continued from previous page

Case	Description	Subcase	Solver	Comments	Graphic output (PDF)
			7	Failed to converge (the number of iterations exceeds the max value of satisfied.	
			10	Failed to converge (the number of iterations exceeds the max value of satisfied.	

3.5 Other ETS related information

- Visualization of the repository activity (x264)
- Visualization of the repository activity (wmv2)

3.6 ETS workflows in KEPLER

The ETS workflow is used for 1-D transport simulation of a tokamak core plasma.

ETS workflows in KEPLER:

- use actors and composite actors from other IMPs, thus for the most recent versions of them please check with relevant project
- complex, but clearly structured workflow, which offers user friendly interface for configuring the simulation
- allow for easy modifications (connecting new modules, or reconnecting parts of the workflow) through an easy graphical interface
- provide users with all updates through the version control system
- still in active development tool

There are currently 2 workflows being developed within EU-IM-IMP3 project:

- ETS_A_4.10b Contact person: Denis Kalupin *(Skype:dkalupin)*
- ETS_A_4.10a Contact person: Denis Kalupin *(Skype:dkalupin)*
- ETS_C Contact person: Vincent Basiuk, Philippe Huynh (Status)

3.6.1 ETS_A 4.10b

3.6.1.1 Obtaining the ETS

Contact person: Denis Kalupin *(Skype: dkalupin)*

3.6.1.1.1 Installing the ETS

The default ETS release is the tag4.10b10.3

Before installation make sure that:

- you have your private data base for the version of the UAL required by the workflow
- you have the version of KEPLER required by the workflow installed. Quick start on kepler required for the ETS can be found here
- inside the window, where you will be downloading the ETS the source command:

```
>source $EU-IMSCRIPTDIR/EU-IMv1 Kepler_Version Data_Base_Name UAL_Version
```

is executed.

To install your local copy of the ETS workflow please do:

```
>svn co https://gforge6.eufus.eu/svn/keplerworkflows/tags/ets_4.10b10.3/ETS
>cd ETS
>make import_ets
```

Press the play button on the workflow.

The workflow shall run! If it does not, please use the contact from above.

Starting the workflow: If you have the workflow already installed, there are there are several ways tio execute it:

- For execution via kepler GUI:

```
>kepler.sh workflow_path/workflow_name.xml
```

- For execution in none GUI mode:

```
>kepler.sh -runwf -nogui -redirectgui $EU-IMHOME/some_dir_name workflow_path/workflow_name.xml
```

- For execution in batch mode: it is essential to keep the workflow inside your $EU-IMWORK area

  it is essential to switch to scripts/R2.2 module

```
>module switch scripts/R2.2
>submit_batch_kepler.sh run_dircetory 1 $EU-IMWORK/workflow_path/workflow_name.xml $EU-
 ↪IMSCRIPTDIR/batch_submission/ParallelKepler.bsub
```

3.6.1.1.2 ETS revisions

Revision Name:	*UAL version:*	*KEPLER:*	*Short Sumary:*	*Comments:*
4.10b0.1	4.10b8_R2.1.0	any, up to 4.10b3.5	Contains:Fixed boundary equlibrium; Simple transport models; full HCD package; Impurity; Pellets; Sawtooth	Test 4.10b release, restricted module choice, restricted physics capabilities, work around of coredelta
4.10b8.1	4.10b8_R2.1.0	central installation 4.10b3_central is preferred; local installation 4.10b3.6 or above	Contains:Fixed boundary equlibrium; Simple transport models; full HCD package; Impurity; Pellets; Sawtooth	Test 4.10b release, restricted module choice, restricted physics capabilities, work around of coredelta, produces scenario output on request
4.10b10.1	4.10b10	central installation 4.10b3_central is preferred; local installation 4.10b3.6 or above	MODIFICATIONS COMPATIBLE WITH 4.10b10 DATA STRUCTURE	UNDER CONSTRUCTION: release at the Code Camp in Prague
4.10b10.2	4.10b10_branches R2.1.r1380	central installation 4.10b3_central is preferred; local installation 4.10b3.6 or above	Added synchrotron radiation, some of neoclassical actors, reworked combiners	UNDER CONSTRUCTION: release at the Code Camp in Prague
4.10b10.3	4.10b10_branches	central	Added	compared to

3.6.1.2 Configuring the ETS run

3.6.1.2.1 Workflow parameters

3.6.1.2.1.1 General Parameters

- USER - your userid
- MACHINE - machine name (database name) for which comutations are done
- SHOT_IN - input shot number
- RUN_IN - input run number
- SHOT_OUT - output shot number
- RUN_OUT - output run number
- NUMERICAL_SOLVER - choice of the numerics solving transport equations (RECOMENDED SELECTION: 3 or 4)

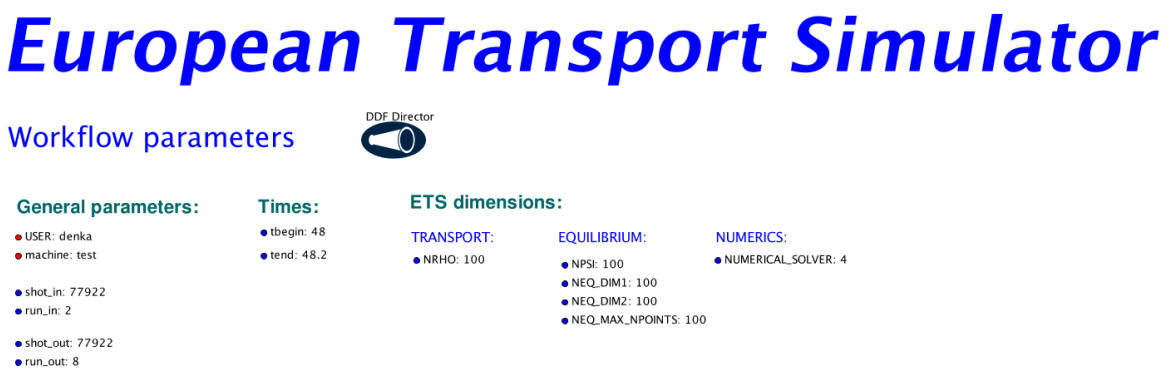
3.6.1.2.1.2 Space resolution

- NRHO - number of radial points for transport equations
- NPSI - number of points for equilibrium 1-D arrays
- NEQ_DIM1 - number of points for equilibrium 2-D arrays, first index
- NEQ_DIM2 - number of points for equilibrium 2-D arrays, second index
- NEQ_MAX_NPOINTS - maximum number of points for equilibrium boundary

3.6.1.2.1.3 Time resolution

Start and End time:

- TBEGIN - Computations start time
- TEND - Computattions end time



Time step:

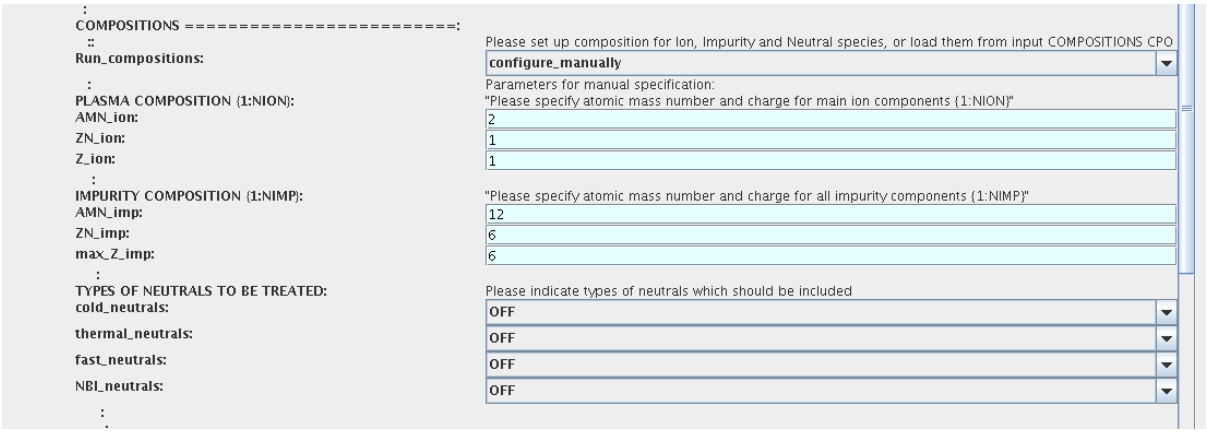
- right click on the box BEFORE THE TIME EVOLUTION
- select Configure actor
- TAU :specify value of the time step in [s]
- TAU_OUT : specify value of the output time interval in [s]
- Commit

3.6.1.2.2 Ion, Impurity and Neutral Composition

Before starting the run you need to define types of main ions, impurity (optional) and neutrals (optional) to be included in simulations.

To define plasma composition:

- right click on the box BEFORE THE TIME EVOLUTION
- select **Configure actor**
- choose one of modes for setting Run_compositions
  - from_input_CPO - will pick up the COMPOSITIONS structure of the COREPROF CPO saved to the input shot;
  - configure_manually - will force the composition from the values specified below
- specify values of atomic mass (AMN_ion), nuclear charge (ZN_ion) and charge (Z_ion , from the first ion to the last [1:NION] , separated by commas
- (optional) specify values of atomic mass (AMN_imp), nuclear charge (ZN_imp) and maximal ionization state (max_Z_imp) for impurity ions, from the first to the last [1:NIMP] , separated by commas
- (optional)for neutrals activate, by switchen them to **ON**, the types which shall be followed by neutral solver
- press **Commit**



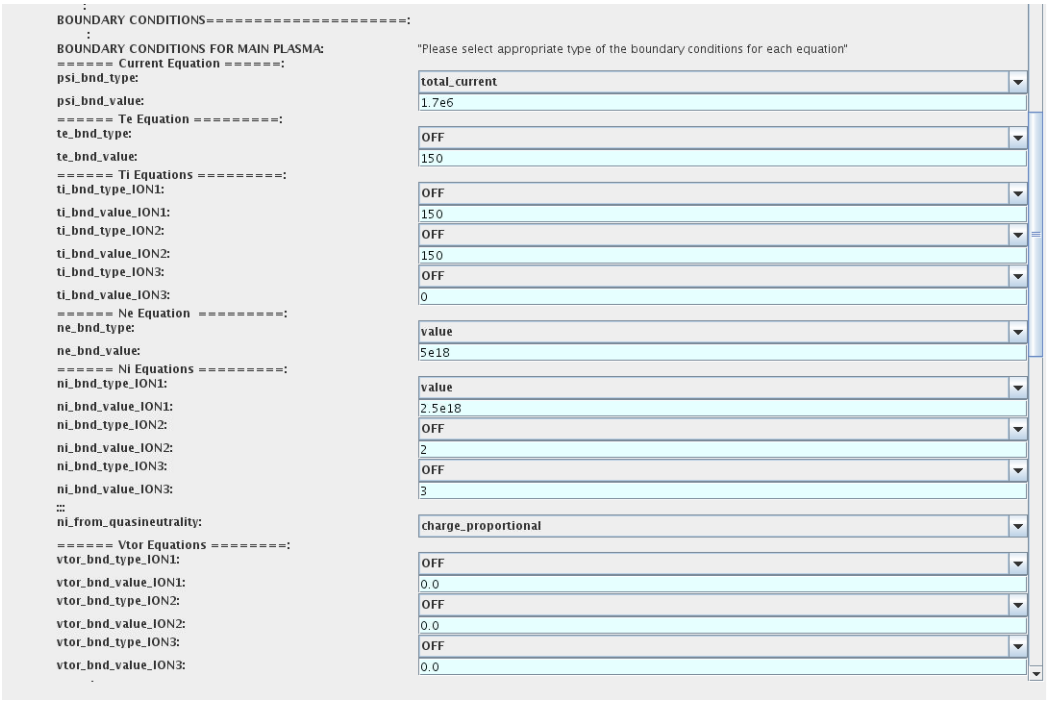
3.6.1.2.3 Equations to be solved and boundary conditions

3.6.1.2.3.1 Main Plasma

Before starting the run you need to select the type and value of the boundary conditions for all equations. Please note that the value should correspond to the type. All equations allow for following types of boundary conditions:

- OFF - equation is not solved, initial profiles will be kept for whole run
- value - edge value should be specified
- gradient - edge gradient should be specified
- scale_length - edge scale length should be specified
- generic - generic form: a1*y´ + a2*y = a3 of the boundary condition is assumed, 3 coefficients (a1, a2, a3) should be provided
- value_from_input_CPO - equation is solved, edge value evolution will be red from input shot
- profile_from_input_CPO - equation is not solved, profile evolution will be red from input shot

The particular equation will be activated if the boundary condition type for it is other than *OFF*



To set up boundary conditions:

- right click on the box BEFORE THE TIME EVOLUTION
- select **Configure actor**
- select appropriate boundary condition for each equation
- specify values for boundary conditions corresponding to the type and to the ion component
- **Commit**

The workflow will not allow the user all particle components (ions[1:NION]+electrons) to be run predictively. At least one of them shall be set to OFF (this component will be computed from quasi-neutrality condition).

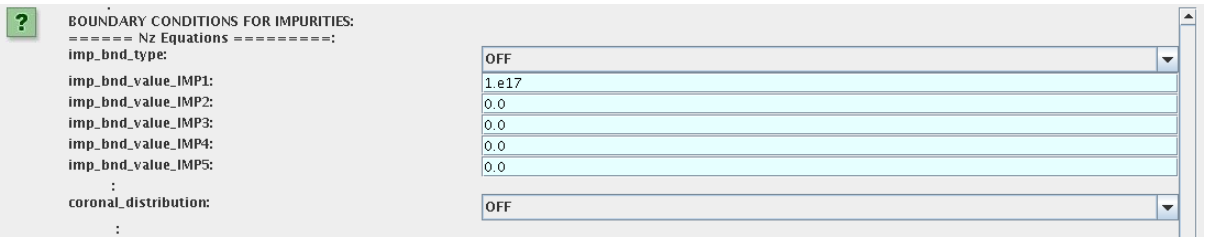
!!! If electron density is solved, all ions with ni_bnd_type=OFF will be computed from the quasineutrality condition and scaled proportional to specified *ni_bnd_value* or inversely proportional to their charge, *charge_proportional*. This is defined by option: *ni_from_quasineutrality*.

3.6.1.2.3.2 Impurity

You can set up the boundary conditions for impurity ions in a similar way as for main ions. !!! Note, that at the moment only types: *OFF*; *value* and *value_from_input_CPO* are accepter by impurity solver.

To set up boundary conditions:

- right click on the box BEFORE THE TIME EVOLUTION
- select **Configure actor**
- select appropriate boundary condition for each impurity species (OFF-equation is not solved)
- specify values for boundary density of each impurity component [1:MAX_Z_IMP], separated by commas
- **Commit**



Interface for impurity boundary condition has additional option, *coronal_distribution*, that allow to preset the edge values or entire profiles of individual ionization states from coronal distribution. In tis case only single value is required to be specified for each impurity boundary value. The options are:

- OFF - the boundary values for impurity densities will be as they are specified above;
- boundary_conditions - the boundary densities will be renormalized with corona, using the first element from above as a total density
- boundary_conditions_and_profiles - the boundary densities and starting profiles will be renormalized with corona, using the first element from above as a total density

3.6.1.2.3.3 Neutrals

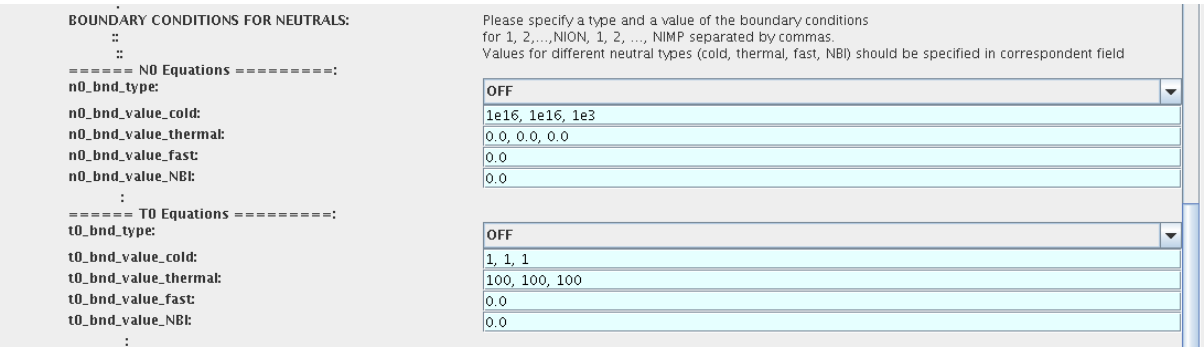
!!! AT THE MOMENT BOUNDARY CONDITIONS FOR NEUTRAL VELOCITIES ARE DISABLED, MIGHT BE ADDED ON REQUEST

Note, that ALL values should be specified in the order: {*1, 2, 3 . . . NION, 1, 2, 3, . . . NIMP*}

To set up boundary conditions:

- right click on the box BEFORE THE TIME EVOLUTION

- select **Configure actor**
- select appropriate boundary condition for each neutral species (OFF-equation is not solved)
- specify values for boundary density and temperature of each neutral component [1, 2, 3 . . . NION, 1, 2, 3, . . . NIMP], separated by commas
- **Commit**



3.6.1.2.3.4 Input profiles interpolation

You are going to start the ETS run from some input shot, which might contain some conflicting rho grids saved to different CPOs. Thus there is a choice for the user to decide on the grid on which the starting profiles should be load by the worflow,

Interpolation_of_input_profiles.

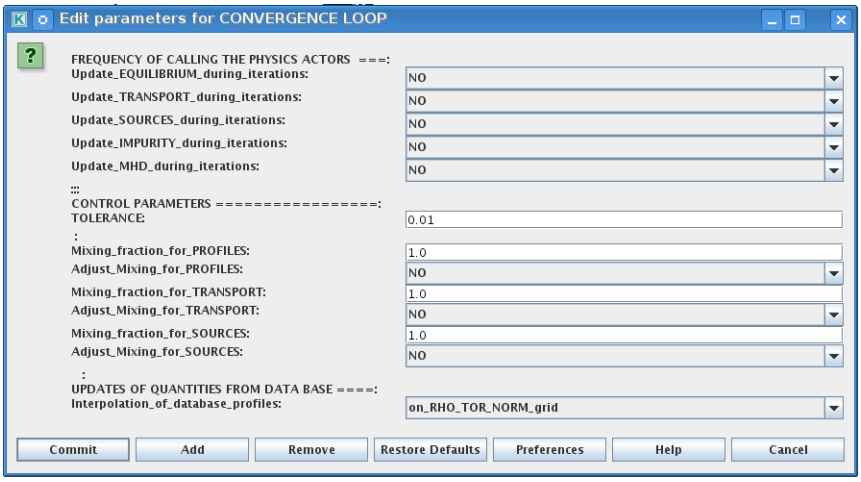
To define the interpolation grid select:

- on_RHO_TOR_grid - interpolate input profiles based on the grid specyfied in [m];
- on_RHO_TOR_NORM_grid - interpolate input profiles based on normalised rho grid [0:1]

3.6.1.2.4 Convergence loop

ETS updates input from different physics actors in a sequence, which is finished by solving the transport equations. Ther are possible none-linear couplings between different parts of the system. These nonelinearities are trited by the ETS using iterations. The decision to step in time is made by the ETS based on the criteria that the maximum relative deviation of main plasma profiles is lower than some predefined tolerance. There is a number of settings and sitches in the ETS that are used by the iterative scheme. To edit them do:

- right click on the box CONVERGENCE LOOP
- select **Configure actor** to edit settings
- choose your settings
- **Commit**



Switches in the field *FREQUENCY OF CALLING THE PHYSICS ACTORS* define how many times the the actors of a certain cathegory (equilibrium, transport, etc.) should be called in a single time step. By selecting *YES* all actors of this cathegory will be called every iteration By selecting *NO* all actors of this cathegory will be called only ones in a time step

Switches and parameters in the field *CONTROL PARAMETERS* define how iterations are done

- Tolerance - defines the maximum relative error of profiles change compared to previous iteration. If it is achieved the time steping is done.

For highly none-linear case the required precision can be achieved faster by the iterative scheme if only fraction of the new solution is mixed to the previous state. The following scheme is adopted by the ets to reduce none-linearities in profiles, transport coefficients and sources:

```
Y = (Amix * Y+) + ((1-Amix)*Y-)
```

where Amix is the mixing fraction You can activate the mixing of profiles, transport coefficient and sources by selecting the corresponding *Mixing_fraction_...* to be between [0:1] You also can activate the authomatic ajustment of this fraction by selecting: *Ajust_Mixing_for_...* to *YES*

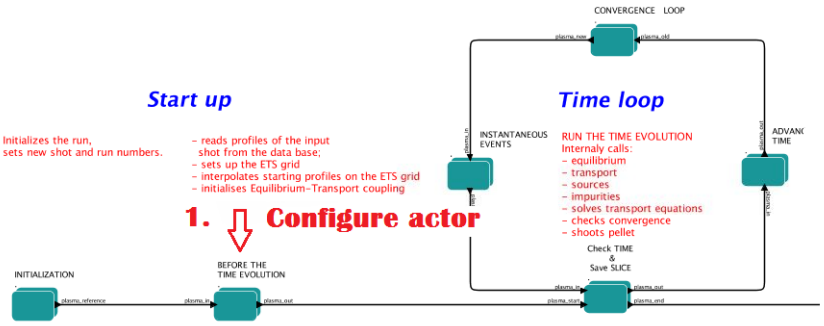
3.6.1.2.5 Equilibrium

3.6.1.2.5.1 Initialization Settings

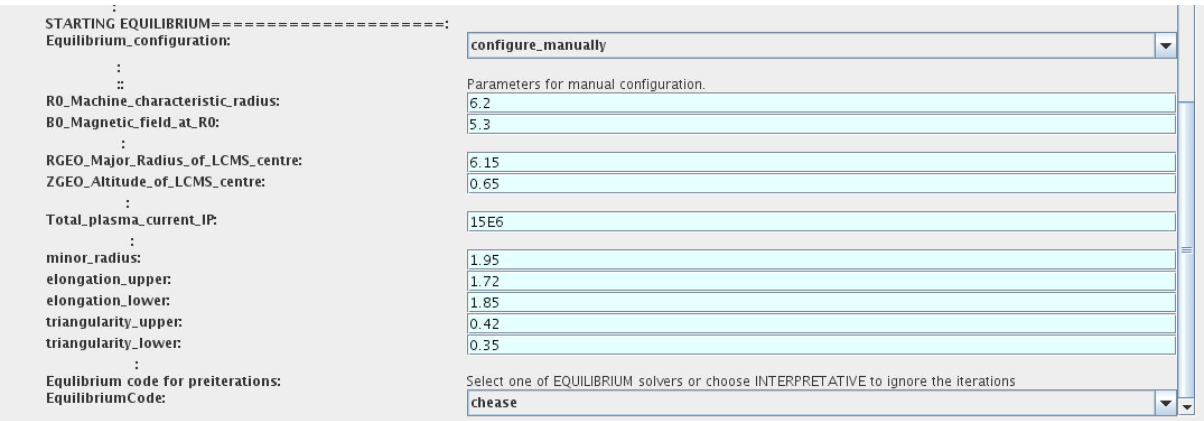
Before starting the run you need to set up your initial equlibrium. There are several options to do it: if your input shot contains the consistent equilibrium with all necessary parameters - you can start immediately from it; if your input shot contains the equilibrium but it is not consistent or some parameters are missing you can check it automatically; if your input equilibrium is corrupt or not present - you can define the starting equlinbrium by tree moment description. To select your starting equilibrium please do:

- right click on the box BEFORE THE TIME EVOLUTION
- select **Configure actor** to edit settings
- Select your settings or specify values
- **Commit**

SETTINGS:



- Equilibrium_configuration - select configure_manually if you like to specify configuration below; select from_input_CPO if all quantities should be picked up from the input CPO
- R0_Machine_characteristic_radius - Characteristic radius of the machine, here B0 is measured [m]
- B0_Magnetic_field_at_R0 - Magnetic field measured at the position R0 [T]
- RGEO_Major_Radius_of_LCMS_centre - R coordinate of the geometrical centre of the LCMS [m]
- ZGEO_Altitude_of_LCMS_centre - Z coordinate of the geometrical centre of the LCMS [m]
- Total_plasma_current_IP - plasma current within the LCMS [A]
- Minor_radius - minor radius of the LCMS [m]
- Elongation - elongation of the LCMS [-]
- Triangularity_upper - upper triangularity of the LCMS [-]
- Triangularity_lower - lower triangularity of the LCMS [-]
- Equilibrium code - select one of available equilibrium solvers to check the consistency between starting equilibrium and current profile; use INTERPRETATIVE if you trust your input data (in this case the check will be ignorred).

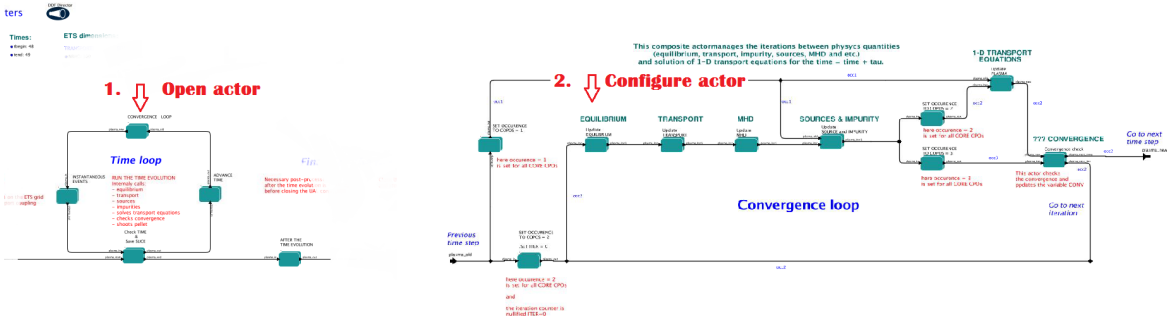


Please note, that different equilibrium solvers might require slightly different input. Thus it is a user responsibility to check that the information inside input shot/run is enough to run selected equilibrium solver.

3.6.1.2.5.2 Run Settings

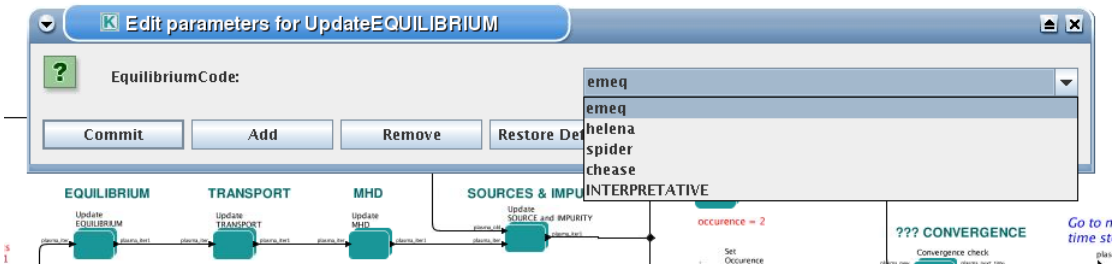
There are several equilibrium solvers connected to the ETS. You can select the one of them.Therefore please do:

- right click on the box CONVERGENCE LOOP
- select **Open actor**
- right click on the box EQUILIBRIUM
- select **Configure actor** to edit settings
- choose your equilibrium solver
- **Commit**



INTERPRETATIVE means that the ETS will not update the equilibrium, instead it will be using the initial equilibrium.

Please note, that it is better to select the same code as you used for pre-iterrations. Because outputs of different equilibrium solver are not necessary done with the same resolution. Therefore the routine saving the information to the data base might brake due to uncompatible sizes of some signals.

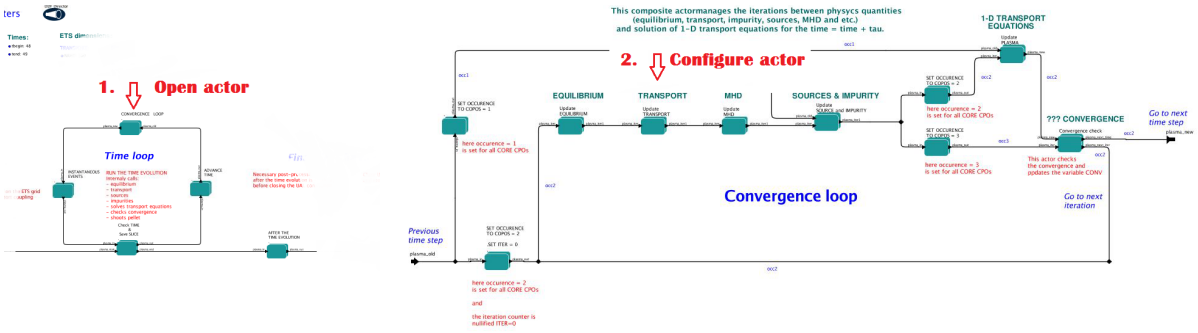


3.6.1.2.6 Transport

The settings for TRANSPORT can be done inside the CONVERGENCE LOOP composite actor. Therefore please do:

- right click on the box CONVERGENCE LOOP
- select **Open actor**
- right click on the box TRANSPORT
- select **Configure actor** to edit settings

- choose your settings
- press **Commit**



3.6.1.2.6.1 Transport models

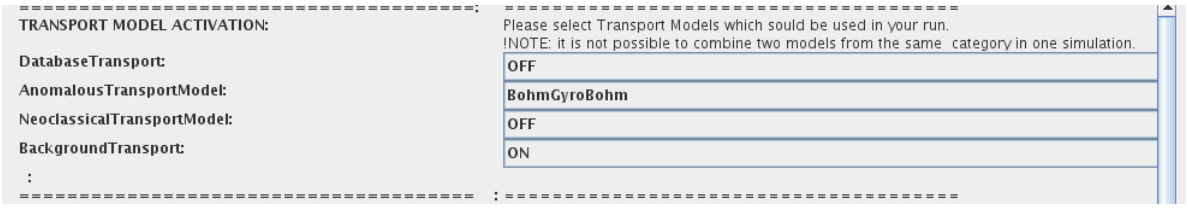
ETS constructs the total transport coefficients from the combination of Anomalous transport (model choice), Neoclassical transport (model choice), Database transport (transport coefficients be saved to the input shot) and Background transport (Transport coefficients defined through the GUI interface)

D_tot = D_DB*M_DB + D_AN*M_AN + D_NC*M_NC + D_BG*M_BG

You should choose from the list of available models in each cathegory or switch it **OFF**

Individual multipliers for all channels shall be specified on the lower level through the code parameters of Transport Combiner

The list of available transport models can be found here.

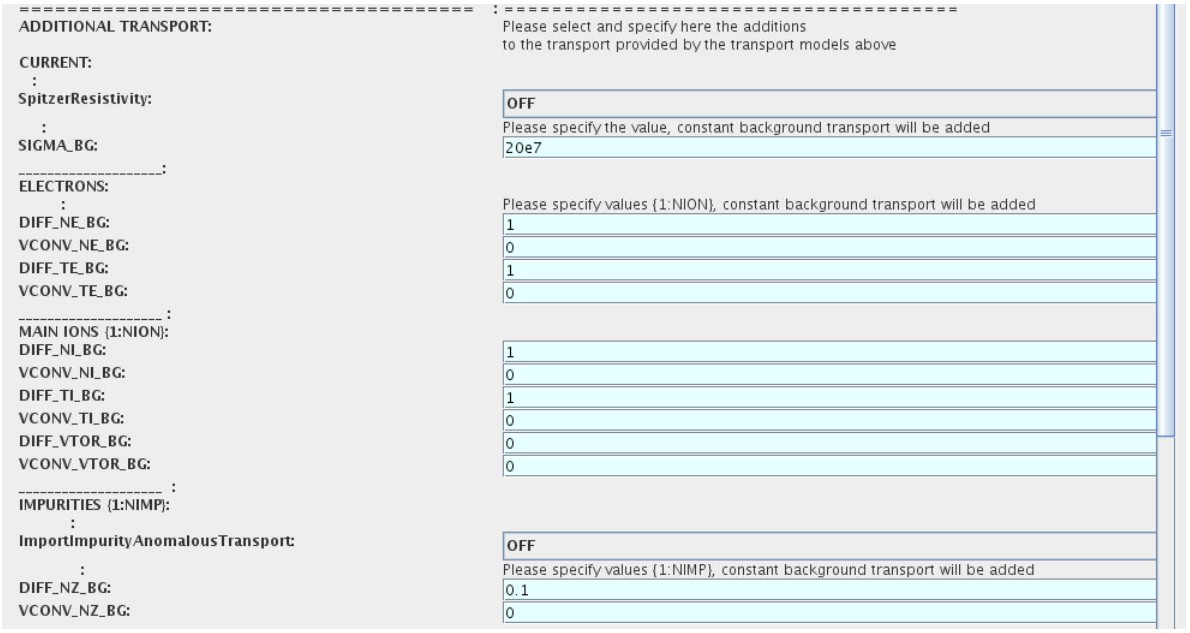


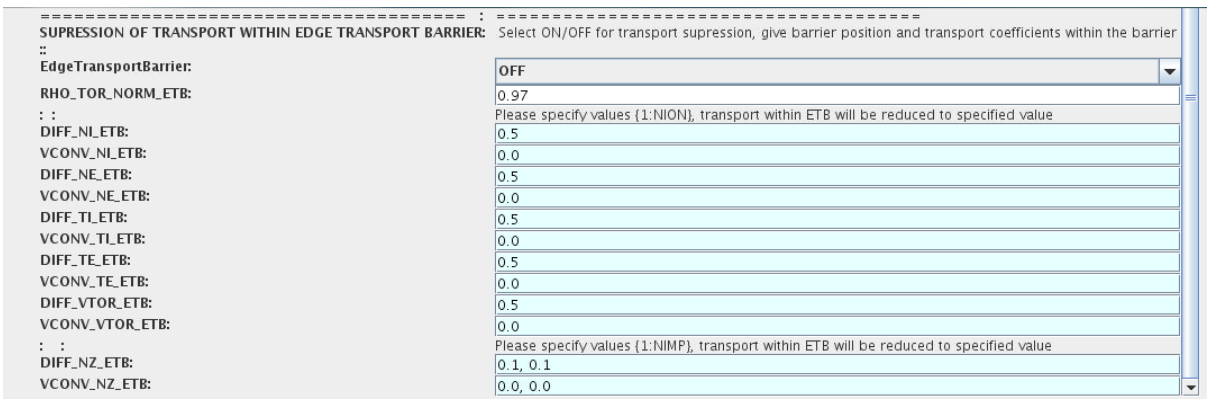
3.6.1.2.6.2 Background transport

You can add the constant background level for each coefficient (ion and impurity coefficients are expected to be the strings of [1:NION] and [1:NIMP] elements respectively, separated by commas)

3.6.1.2.6.3 Edge transport barrier

In this section you can artificially supress the transport outside of specified *RHO_TOR_NORM_ETB*. Total transport coefficients for all transport channels (ne, ni, nz, Te, Ti,...) will be reduced to constant values specified below (ion and impurity coefficients are expected to be the strings [1:NION] and [1:NIMP] respectively)

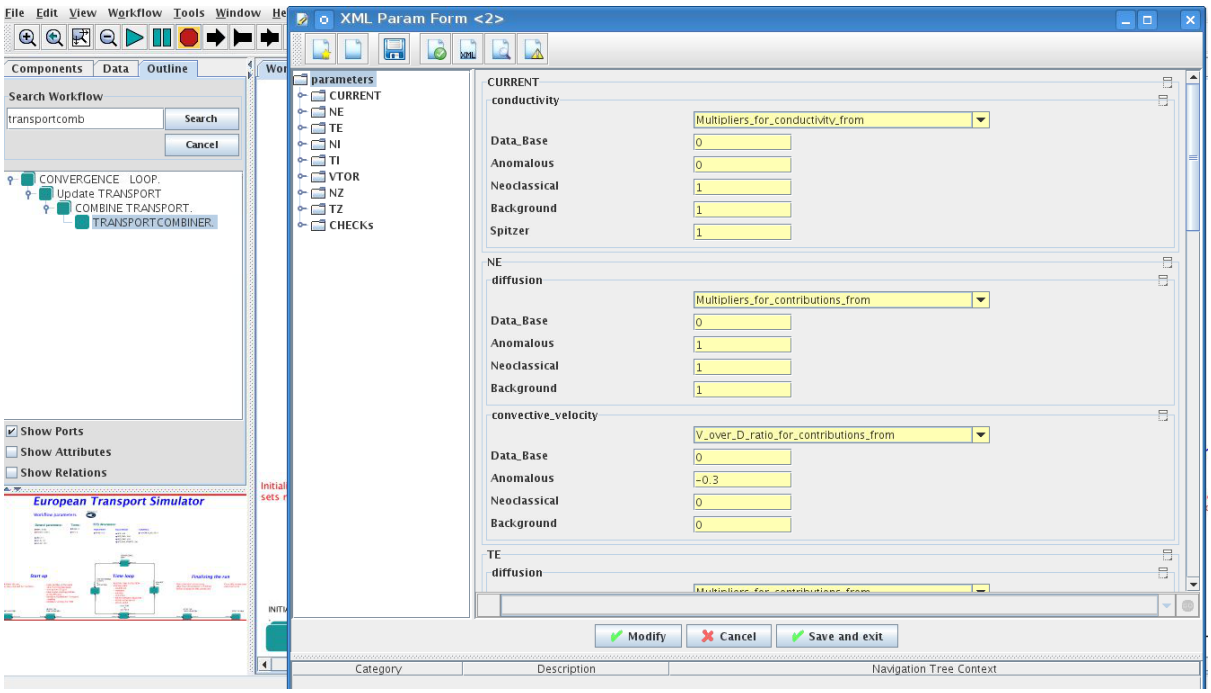

ADDITIONAL TRANSPORT:
Please select and specify here the additions
to the transport provided by the transport models above
CURRENT:
SpitzerResistivity:
OFF
Please specify the value, constant background transport will be added
SIGMA_BG:
20e7
ELECTRONS:
Please specify values {1:NION}, constant background transport will be added
DIFF_NE_BG:
1
VCONV_NE_BG:
0
DIFF_TE_BG:
1
VCONV_TE_BG:
0
MAIN IONS {1:NION}:
DIFF_NI_BG:
1
VCONV_NI_BG:
0
DIFF_TI_BG:
1
VCONV_TI_BG:
0
DIFF_VTOR_BG:
0
VCONV_VTOR_BG:
0
IMPURITIES {1:NIMP}:
ImportImpurityAnomalousTransport:
OFF
Please specify values {1:NIMP}, constant background transport will be added
DIFF_NZ_BG:
0.1
VCONV_NZ_BG:
0


SUPRESSION OF TRANSPORT WITHIN EDGE TRANSPORT BARRIER:
Select ON/OFF for transport supression, give barrier position and transport coefficients within the barrier
EdgeTransportBarrier:
OFF
RHO_TOR_NORM_ETB:
0.97
Please specify values {1:NION}, transport within ETB will be reduced to specified value
DIFF_NI_ETB:
0.5
VCONV_NI_ETB:
0.0
DIFF_NE_ETB:
0.5
VCONV_NE_ETB:
0.0
DIFF_TI_ETB:
0.5
VCONV_TI_ETB:
0.0
DIFF_TE_ETB:
0.5
VCONV_TE_ETB:
0.0
DIFF_VTOR_ETB:
0.5
VCONV_VTOR_ETB:
0.0
Please specify values {1:NIMP}, transport within ETB will be reduced to specified value
DIFF_NZ_ETB:
0.1, 0.1
VCONV_NZ_ETB:
0.0, 0.0

3.6.1.2.6.4 Total transport coefficients

The fine tuning of of transport coefficients can be done through editing the XML code parameters of the **transport combiner** actor:

- In Outline browse for transportcombiner
- select **Configure actor**
- click **Edit Code Parameters**
-
  - If you select **OFF** contributions from all transport models to this channel will be nullified;
  - If you select **Multipliers_for_contributions_from** the transport channel will be activated, and the total transport coefficient will be combined from active tranport models. You gust need to specify multiplier against each channel;
  - For convective velocity there is an additional option **V_over_D_ratio_for_contributions_from**. With this option selected the combiner will ignore the convective components provided by transport models. The convective velocity will be determined from the diffusion coefficient by applying fixed V/D ratio (for inward pinch the values should be negative!).
- **Save and exit**
- **Commit**

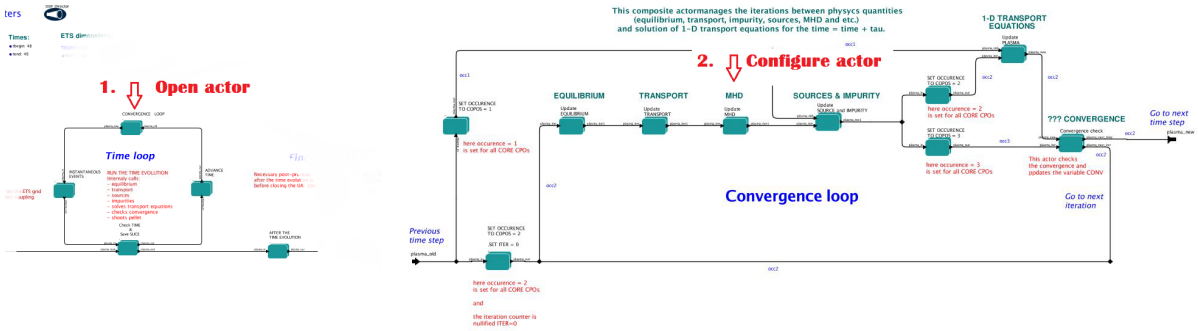


3.6.1.2.7 MHD

The settings for MHD type of events can be done inside the CONVERGENCE LOOP composite actor. Therefore please do:

- right click on the box CONVERGENCE LOOP
- select **Open actor**

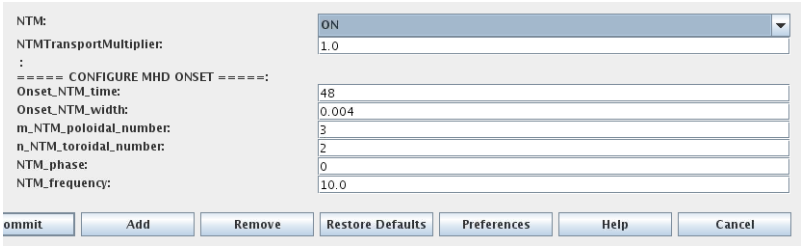
- right click on the box MHD
- select **Configure actor** to edit settings
- choose your settings
- **Commit**



At the moment ETS allows only for NTM to be activated. The sawtooth module is expected to be deployed before EU-IM Code Camp in Slovenia.

User can ajust the following NTM settings:

- NTM – **ON** means that ETS will add the NTM driven transport to the total transport coefficient; **OFF** -ignored
- NTMTransportMultiplier – the transport contrinution from NTM will be multiplied with this value
- Onset_NTM_time - activation time for the NTM mode
- Onset_NTM_width - starting width of the mode
- m_NTM_poloidal_number
- n_NTM_toroidal_number
- NTM_phase
- NTM_frequency

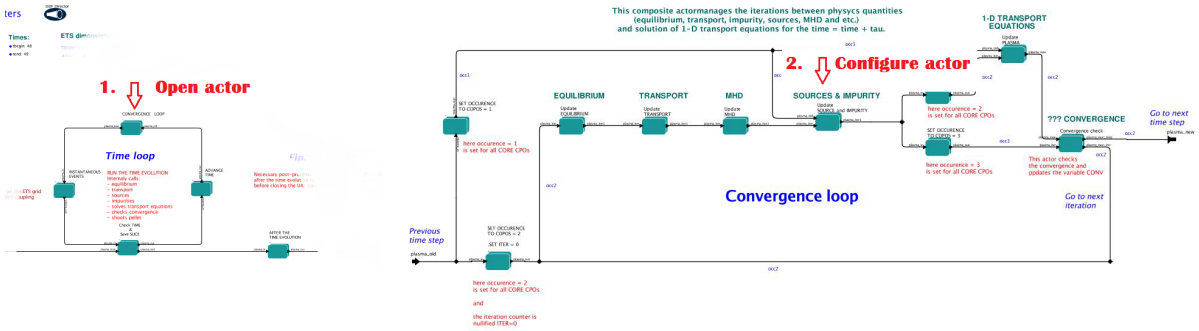


3.6.1.2.8 Sources and impurity

The settings for SOURCES AND IMPURITY can be done inside the CONVERGENCE LOOP composite actor. Therefore please do:

- right click on the box CONVERGENCE LOOP
- select **Open actor**
- right click on the box SOURCES AND IMPURITY

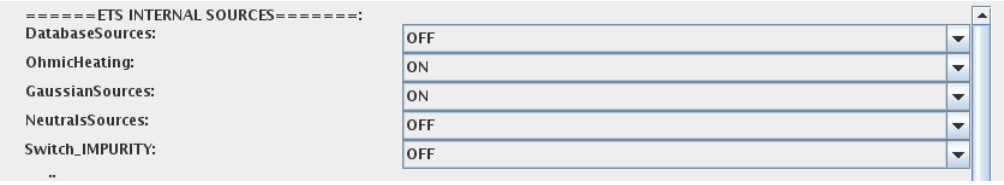
- select **Configure actor** to edit settings
- choose your settings
- **Commit**



3.6.1.2.8.1 Analytical & Impurity sources

There is a number of sources developed by IMP3 project, which are actors or internal routines of the transport solver. You can activate them by selecting **ON / OFF** in front of corresponding source:

- Database Sources – **ON** - ETS will pick up the evolution of source profiles saved to your input shot/run; **OFF** -ignored
- Ohmic Heating – **ON** - ETS will compute Ohmic heating internaly; **OFF** -ignored
- Gaussian Sources – **ON** - ETS will add sources from the Gaussian source actor (you can configure heat and particle deposition profiles by editing the code parameters of the actor); **OFF** -ignored
- Neutral Sources – **ON** - Fluid neutrals will be solved according to the boundary conditions specified on ¨Before_time_evolution¨ composite actor interface; **OFF** -ignored
- Switch_IMPURITY – **ON** - Impurity density and radiative sources will be computed; **OFF** - ignored; **INTERPRETATIVE** – profiles of impurity density will be read from input shot/run



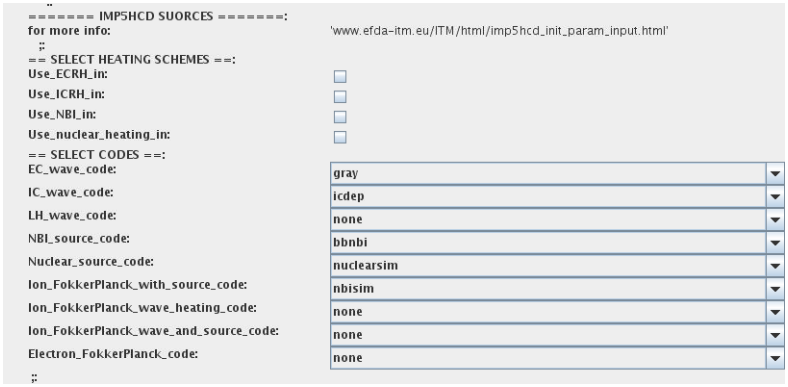
3.6.1.2.8.2 HCD sources

There is a number of sources developed by HCD project, that are incorporated by the ETS workflow.

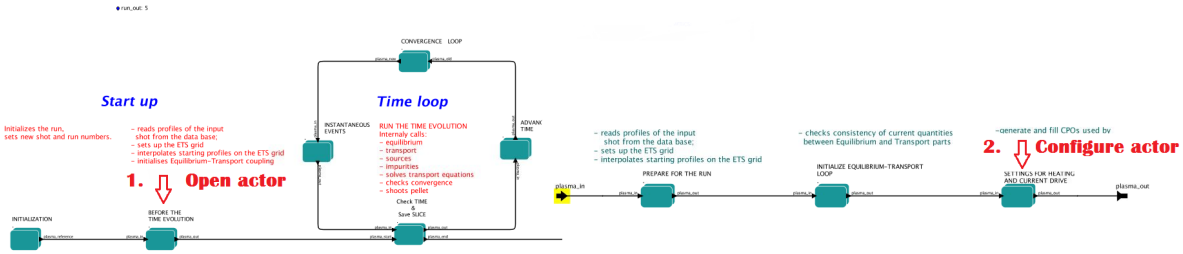
For the HCD sources please activate the type of heating source, by ticking the box in front of it, and select the code to simulate it.

You also need to configure initial IMP5HCD settings. Therefore please:

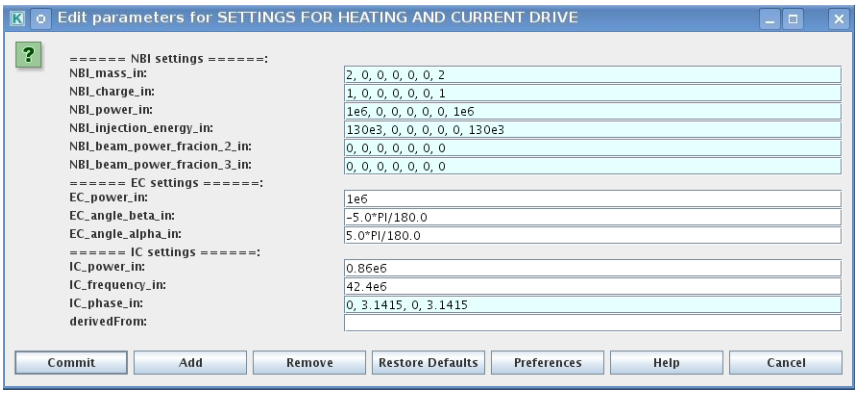
- right click on the box BEFORE THE TIME EVOLUTION
- select **Open Actor**
- right click on the box SETTINGS FOR HEATING AND CURRENT DRIVE



- select **Configure actor**
- edit the stettings
- **Commit**



The detailed information on initial IMP5HCD settings can be found here. Please note that settings for NBI are done independent for each PINI. Therefore, for NBI settings, please insert the values separated by commas. The number of the element in the array corresponds to the number of activated PINI. Maximum accepted number of PINIs = 16.

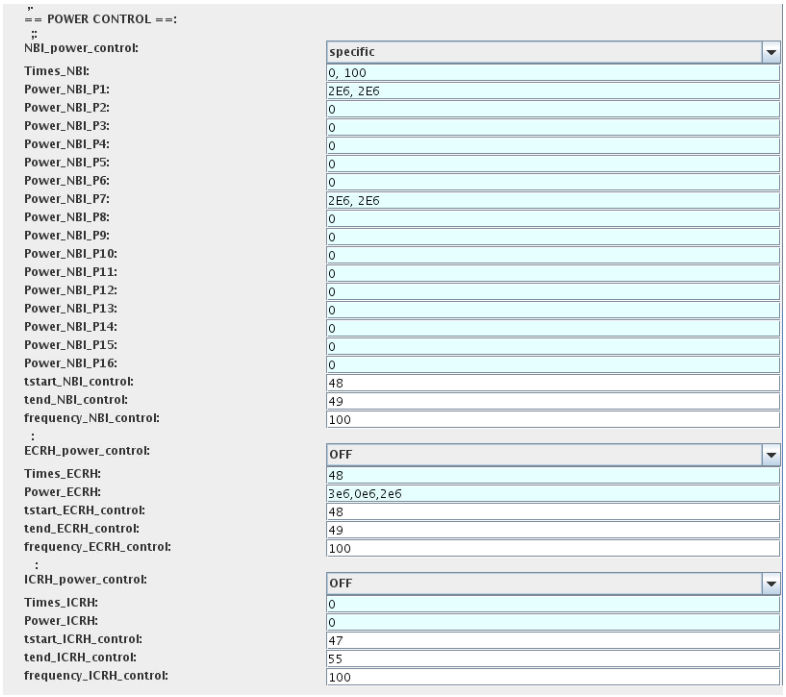


3.6.1.2.8.3 Power control

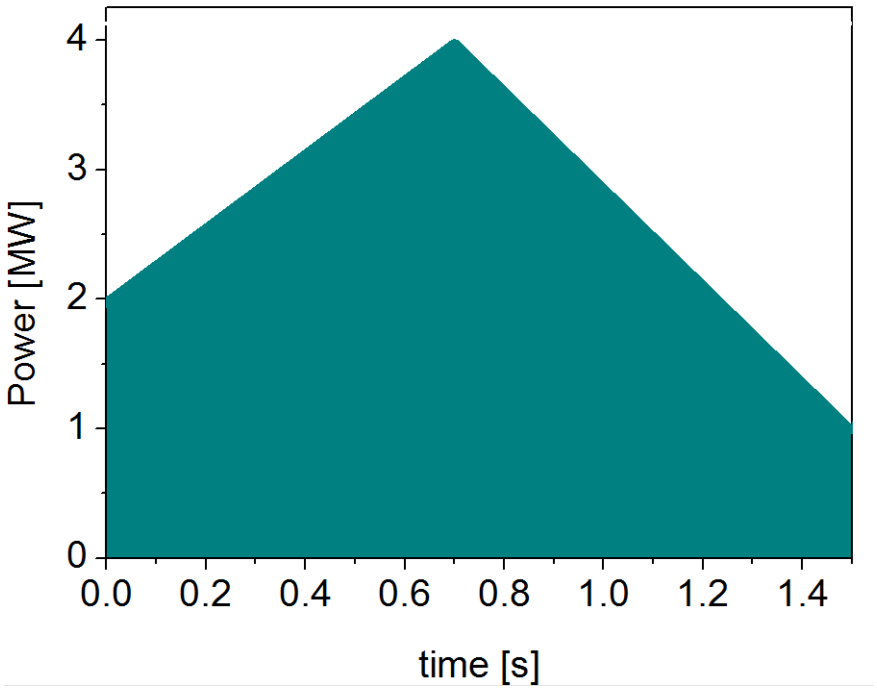
You also can activate the power control for the IMP5HCD sources.

If the POWER_CONTROL is not **OFF**, there are two modes of operation: **specific** and **frequency**

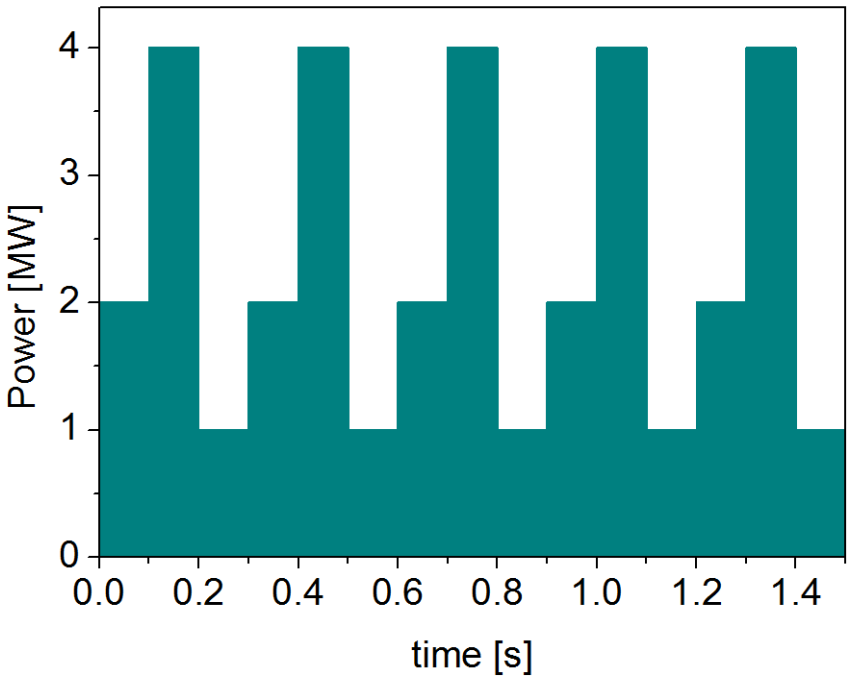
For **specific** you should specify the time sequence separated by commas and the corresponding power sequence (where first power level corresponds to the first time, second to second and etc.). Linear



interpolation will be done between the sequence points. For example: if you give the power **sequence** = 2e6,4e6,1e6 and **times** = 0.0, 0.7, 1.5 (s) the delivered power would be:



For **frequency** you should specify the power levels sequence separated by commas, start and end time of the power control and the frequency of switching between these levels. For example: if you give the power **sequence** = 2e6,4e6,1e6 and **frequency** = 10 (Hz) **tstart** = 0.0 (s) **tend** = 1.5 (s) the delivered power would be:



3.6.1.2.8.4 Total power

Profiles of the total source for each channel are obtained from the the individual contributions (Data Base, Gaussian, Neutrals, Impurity and HCD) as a summ of all activated sources multiplied with coefficients specified on the interface of the composite actor.

S_tot = S_DS*DSM + S_GS*GSM + S_Neu*NeuSM + S_IMP*IMPSM + S_HCD*HCDSM

The fine tuning of of sources can be done through editing the XML code parameters of the source combiner actor:

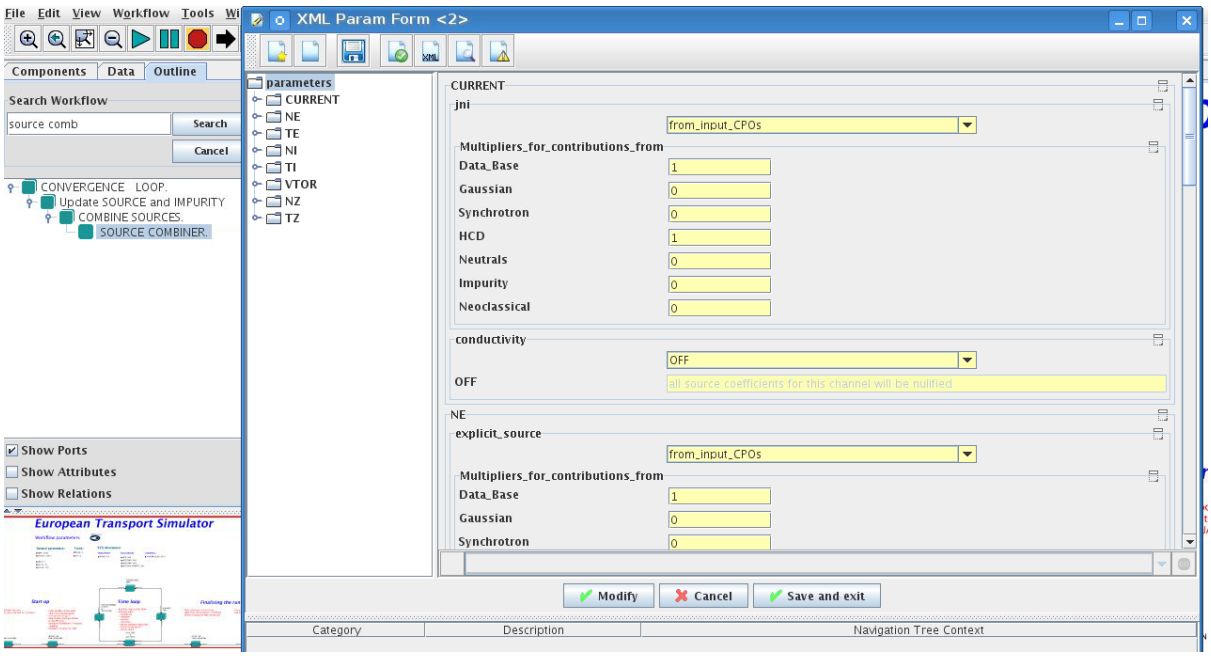
- In the Outline browse for source combiner
- select **Configure actor**
- click **Edit Code Parameters**
- If you like the sources to the particular equation being activated - select **from_input_CPOs**, and then, put the multipliers against each contribution; if you select **OFF** contributions from all sources to this channel will be nullified.
- save and exit
- **Commit**

3.6.1.2.9 Instantaneous events & Actuators

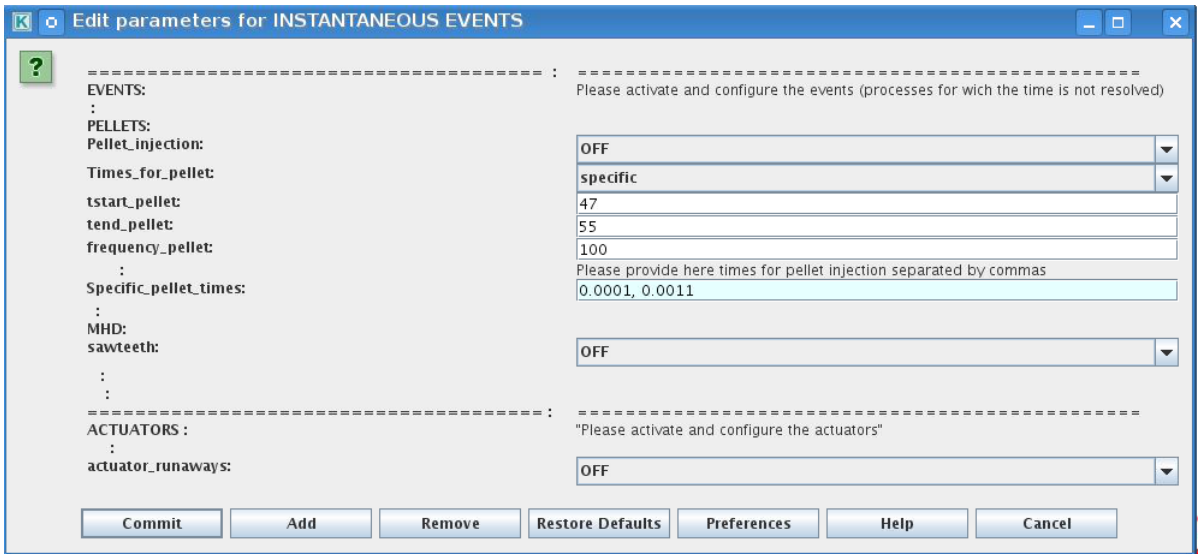
At the moment, user can swith **ON** and **OFF** two types of events: PELLET and SAWTOOTH

3.6.1.2.9.1 Pellet

At the top level of the workflow you can configure times for pellet injection

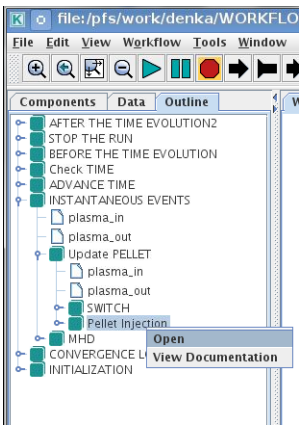


- right click on the box INSTANTANEOUS EVENTS & ACTUATORS
- select **Configure actor** to edit settings
- Select Pellet_injection equal **ON** if you like to use pellet in your simulation
- Select mode of operation:
  - Times_for_pellets equals **specific** – pellets will be shut at exact times specified in array times_pellet
  - Times_for_pellets equals **frequency** – pellets will be shut from tstart_pellet until tend_pellet with a frequency_pellet
- **Commit**

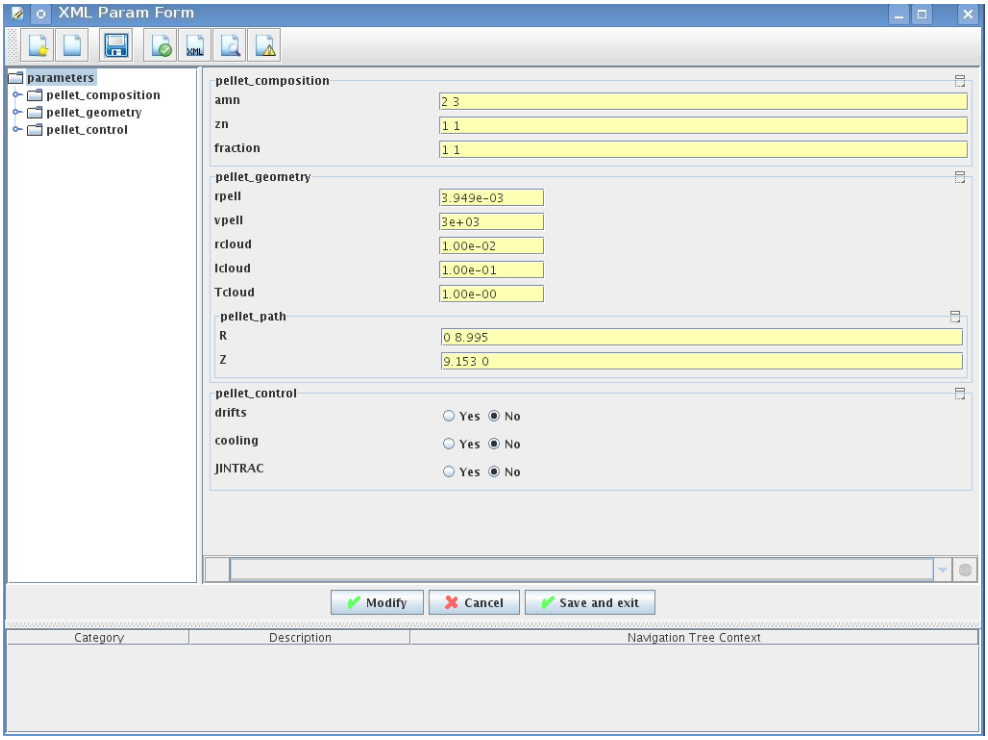


Parameters of individual pellet need to be configured through the code_parameters of the PELLET actor. To access it go to **Outline** on the right upper corner and open the following:

- right click on the actor PELLET



- select **Configure actor**
- click **Edit Code Parameters**
- edit parameters and click **save and exit**
- **Commit**



amn – atomic mass number: array of elements separated by space (1:nelements) [-]

zn – nuclear charge: array of elements separated by space (1:nelements) [-]

fraction – fraction of each element in the pellet, based on the number of atoms: array of elements separated by space (1:nelements) [-]

rpell – radius of the pellet [m]

vpell – velocity of the pellet [m/s]

rcloud – radius of the pellet cloud [m], radial extension of the cloud = 2*rp0

lcloud – length of the pellet cloud along the field line [m]

Tcloud – temperature of the pellet cloud [eV]

Pellet path is specified by two points, for which R and Z coordinated should be specified

R – R coordinates of the pivot and second points of the pellet path, separated by space [m]

Z – Z coordinates of the pivot and second points of the pellet path, separated by space [m]

Control switches allow to activate:

- drifts - YES - will activate radial displacement of deposition profile, same for all path points
- cooling - YES - will activate cooling of the other side of the plasma due to parallel heat transport (essential for large pellets, which might cross the same flux surface twice)
- JINTRAC - YES - will provide temperature reduction consistent with the model used in JETTO

3.6.1.2.9.2 Sawtooth

At the top level of the workflow you can switch ON/OFF possible MHD events

- right click on the box INSTANTANEOUS EVENTS & ACTUATORS
- select **Configure actor** to edit settings
- Select SAWTOOTH **ON** if you like to use them in your simulation
- **Commit**

3.6.1.2.9.3 Actuators

At the top level of the workflow you can switch ON/OFF actuator for runaways

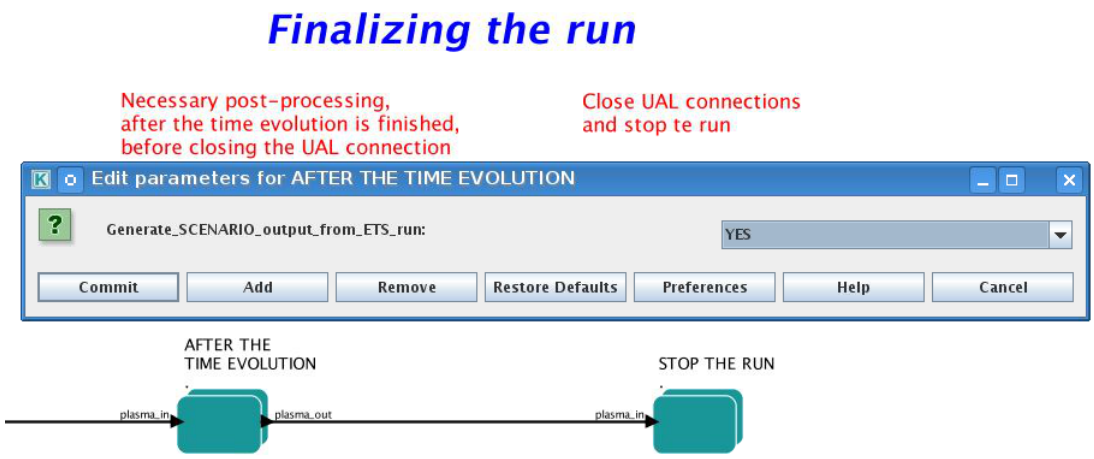
- right click on the box INSTANTANEOUS EVENTS & ACTUATORS
- select **Configure actor** to edit settings
- Select actuator_runaways **ON** if you like to use them in your simulation
- **Commit**

3.6.1.2.10 Scenario output

You can summarize the ETS run by activating the output to SCENARIO CPO (as post-processing of the run).

To activate the SCENARIO output:

- right click on the box AFTER THE TIME EVOLUTION
- select **Configure actor**
- select Generate_SCENARIO_output_from_ETS_run equal **YES**
- **Commit**



3.6.1.2.11 Visualization

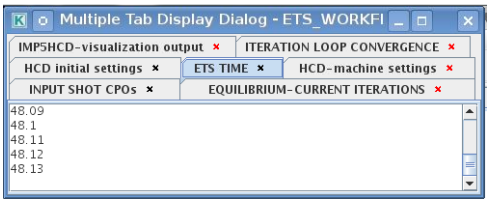
There is a number tools visualizing the ETS run.

3.6.1.2.12 Multiple Tab Display

The display appeares automaticaly when the ETS workflow is launched. It displays diagnostic text messages from the workflow on following topics:

- Input data statement
- Iterations to check the initial convergence between EQUILIBRIUM and CURRENT
- Time evolution
- Convergence of iteratinos within the time step
- IMP5HCD settings
- Power used by IMP5HCD actors durung the run

Also the error messages from execution of the workflow will be displayed here.



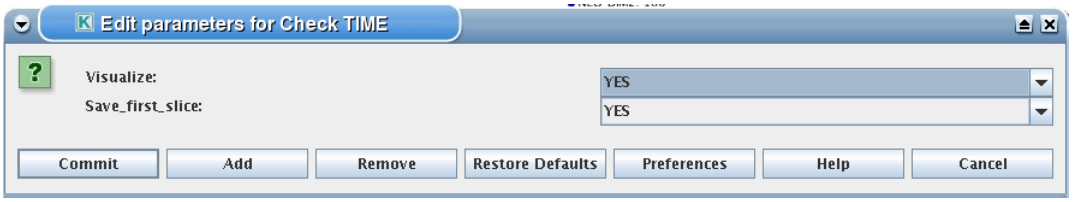
3.6.1.2.13 Python Visualization Display

Please note, if you plan to use python based vizualization **nomatlab** argument is essential by the opening of the workflow.

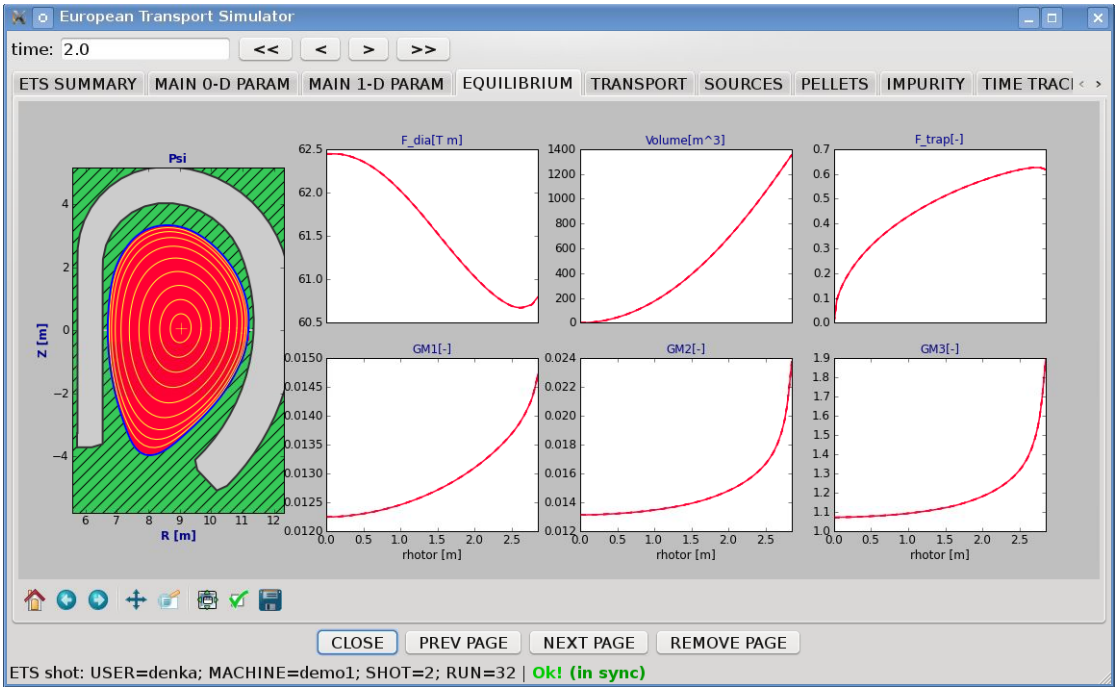
```
>kepler.sh nomatlab workflow_path/workflow_name.xml
```

You can activate the graphical visualization of your run evolution:

- right click on the box Check Time & Save Slice
- select **Configure actor**
- select visualisation **YES** or **NO**
- **Commit**



Then evolution of main discharge parameters will be shown in this window:



3.6.1.3 List of Actors

UNDER DEVELOPMENT

3.6.1.3.1 Equilibrium actors

Code name	Code Category	Contact persons	Short description
chease	Grad-Shafranov solver	Olivier Sauter	Chease is a fixed boundary Grad-Shafranov solver based on cubic hermitian finite elements see H. Lütjens, A. Bondeson, O. Sauter, Computer Physics Communications 97 (1996) 219-260
emeq	/	/	
spider	/	/	

3.6.1.3.2 Core transport actors

Code name	Code Category	Contact persons	Short description
ETS	Transport solver	Denis Kalupin	
BohmGB	Bohm/gyro-Bohm transport coefficients	/	
TCI/Weiland	Transport coefficient from coefficients	Pär Strand	
TCI/GLF23	Transport coefficient from drift wave turbulence	/	
TCI/RITM	Transport coefficient from drift wave turbulence	/	
TCI/MMM (not yet in ETS)	Transport coefficient from drift wave turbulence	/	
TCI/EDWM (not yet in ETS)	Transport coefficient from drift wave turbulence	/	
nclass (not yet in ETS)	Neoclassical transport coefficients	Pär Strand	
neos (not yet in ETS)	Neoclassical transport coefficients	Olivier Sauter	

3.6.1.3.3 Edge transport actors

3.6.1.3.4 Heating and current drive actors

Code name	Code Category	Contact persons	Short description
gray	EC/waves	Lorenzo Figini	GRAY is a quasi-optical ray-tracing code for electron cyclotron heating & current drive calculations in tokamaks. Code-parameter documentation can be found
travis	EC/waves	Nikolai Marushchenko and Lorenzo Figini	Travis is a ray-tracing code for electron cyclotron heating & current drive calculations in tokamaks.
Torray-FOM	EC/waves	Egbert Westerhof	Torray-FOM is a ray-tracing code for electron cyclotron heating & current drive calculations in tokamaks.
bbnbi	NBI/source	Otto Asunta	Calculate the deposition rates of neutrals beam particles, i.e. the input source for Fokker-Planck solvers (not the heating and current drive). Note that the number of markers generated by BBNBI is described by the kepler variable num-ber_nbi_markers_in.
nemo	NBI/source		

3.6.1.3.5 Events actors

Code name	Code Category	Contact persons	Short description
pelletactor	pellet	Denis Kalupin	
pellettrigger	pellet	Denis Kalupin	
sawcrash_slice	sawteeth	Olivier Sauter	
sawcrit	sawteeth	Olivier Sauter	
runaway_indicator	runaway	Roland Lohneroch Gergo Pokol	Indicating the presence of runaway electrons: 1) Indicate, whether electric field is below the critical level, thus runaway generation is impossible. 2) Indicate, whether runaway electron growth rate exceeds a preset limit. This calculation takes only the Dreicer runaway generation method in account and assumes a velocity distribution close to Maxwellian, therefore this result should be considered with caution. The growth rate limit can be set via an input of the actor. Limit value is set to \(10^{12} \) particle per second by default. (This growth rate generates a runaway current of approximately 1kA considering a 10 seconds long discharge.)

3.6.1.3.6 Non-physics actors

The ETS uses the following list of non-physics actors: addECant, addICant, backgroundtransport, calculateRHO, changeocc, changepsi, changeradii, checkconvergence, controlAMIX, coredelta2coreprof, correctcurrent, deltacombiner, emptydistribution, emptydistsource, emptywaves, eqinput, etsstart, fillcoreimpur, fillcoreneutrals, fillcoreprof, fillcoresource, fillcoretransp, fillequilibrium, fillneoclassic, filltoroidfield, gausiansources, geomfromcpo, hcd2coresource, ignoredelta, ignoreimpurity, ignoreneoclassic, ignoreneutrals, ignorepellet, ignoresources, ignoretransport, IMP4dv, IMP4imp, importimptransport, itmimpurity, itmneutrals, merger4distribution, merger4distsource, merger4waves, nbifiller, neoclassic2coresource, neoclassic2coretransp, parabolicprof, plasmacomposition, PowerFromArray, PowerModulation, profilesdatabase, readjustprof, sawupdate_slice, scaleprof, sourcecombiner, sourcedatabase, transportcombiner, transportdatabase, wallFiller and waves2sources.

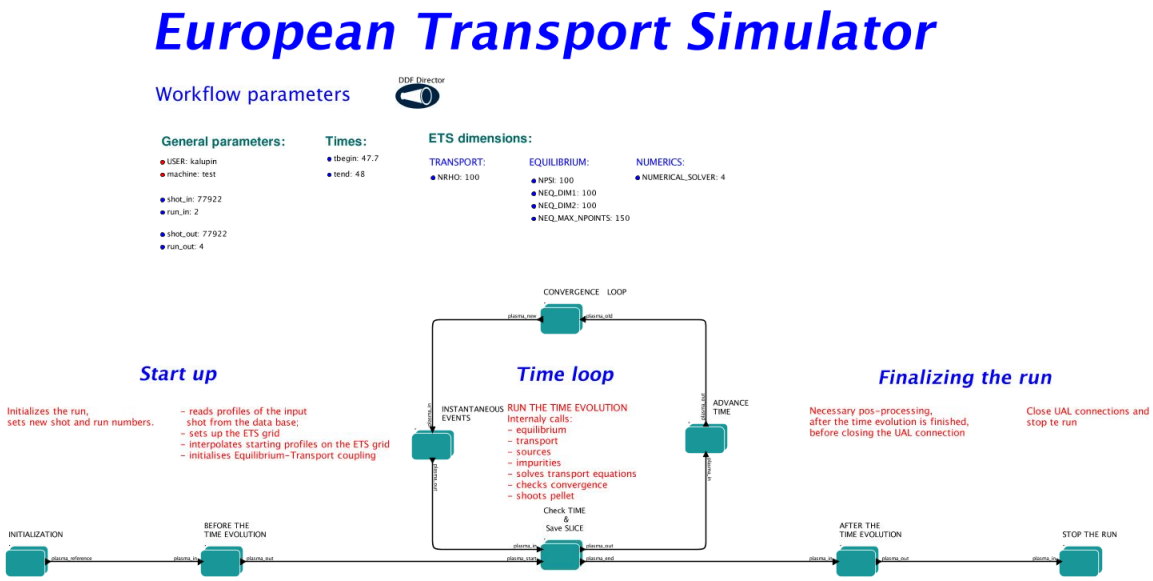
3.6.2 ETS_A 4.10a

ETS_A workflow in KEPLER:

- uses as actors and composite actors from other IMPs, thus for the most recent versions of them please check with relevant project
- complex, but clearly structured workflow, which offers user friendly interface for configuring the simulation
- **allows for easy modifications (connecting new modules, or reconnecting** the parts of the workflow) through the easy graphical interface
- provides users with all updates through the version control system
- still actively developing tool

The list and status of available physics models for the ETS_A can be found here.

Contact person: Denis Kalupin (Skype: dkalupin)*



3.6.2.1 Obtaining the ETS

Copy the ETS workflow to your space:

```
>svn co https://gforge6.eufus.eu/svn/keplerworkflows/trunk/4.10a/imp3/ets $EU-IMSCRATCH/ETS_
 ↪WORKFLOWS
```

Compile ETS actors:

```
>cd $EU-IMSCRATCH/ETS_WORKFLOWS
>make import_ets
```

3.6.2.2 Updating the ETS

If you have already a copy of the ETS you do not need to check it out again!!!

If you like to update everything (WORKFLOW + ACTORS + VISUALIZATION + INPUT DATA)

```
>cd $EU-IMSCRATCH/ETS_WORKFLOWS
>svn update
>make import_ets
```

To update ETS actors go inside your ETS_ACTORS:

```
>cd $EU-IMSCRATCH/ETS_WORKFLOWS
>svn update
>make import_actors
```

To update the workflow go inside your ETS_WORKFLOWS:

```
>cd $EU-IMSCRATCH/ETS_WORKFLOWS
>svn update
```

To update visualization scripts go inside your $KEPLER/kplots:

```
>svn update
```

This is ALL you need to do for updates!

3.6.2.3 Executing the ETS

Open ETS workflow in Kepler:

```
>kepler.sh $EU-IMSCRATCH/ETS_WORKFLOWS/ETS_WORKFLOW.xml
```

on the top of the workflow, change the parameter “user” to your user_ID.

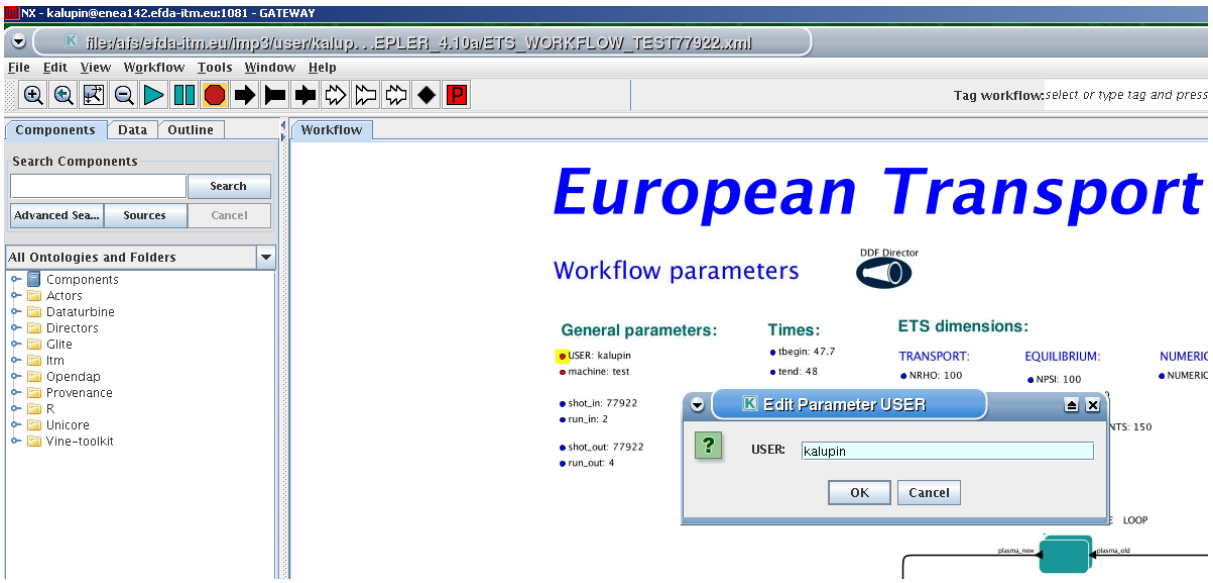
You can run the workflow!!!

3.6.2.4 Configuring the ETS run

3.6.2.4.1 Workflow Parameters

3.6.2.4.1.1 General Parameters

- USER - your userid



- MACHINE - machine name (database name) for which comutations are done
- SHOT_IN - input shot number
- RUN_IN - input run number
- SHOT_OUT - output shot number
- RUN_OUT - output run number
- NUMERICAL_SOLVER - choice of the numerics solving transport equations (RECOMENDED SELECTION: 3 or 4)

3.6.2.4.1.2 Space resolution

- NRHO - number of radial points for transport equations
- NPSI - number of points for equilibrium 1-D arrays
- NEQ_DIM1 - number of points for equilibrium 2-D arrays, first index
- NEQ_DIM2 - number of points for equilibrium 2-D arrays, second index
- NEQ_MAX_NPOINTS - maximum number of points for equilibrium boundary

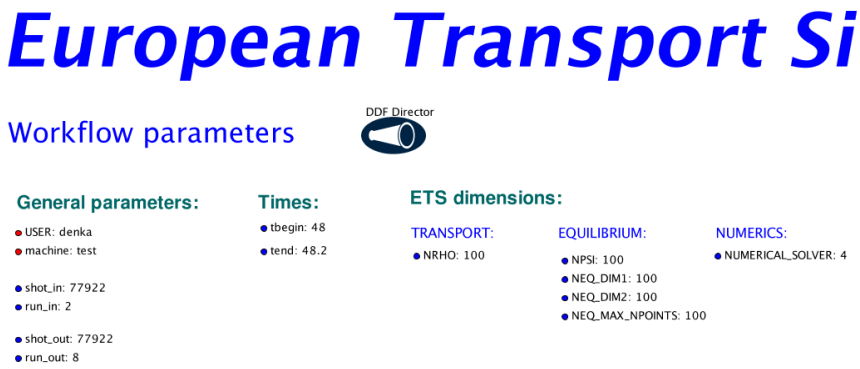
3.6.2.4.1.3 Time resolution

Start and End time

- TBEGIN - Computations start time
- TEND - Computattions end time

Time spep

- right click on the box ‘BEFORE THE TIME EVOLUTION’
- select ‘Configure actor’
- TAU:specify value of the time step in [s]



- TAU_OUT: specify value of the output time interval in [s]
- Commit

3.6.2.4.2 Plasma, Impurity and Neutrals Composition

Before starting the run you need to define types of main and impurity ions and types of neutrals to be included in simulations.

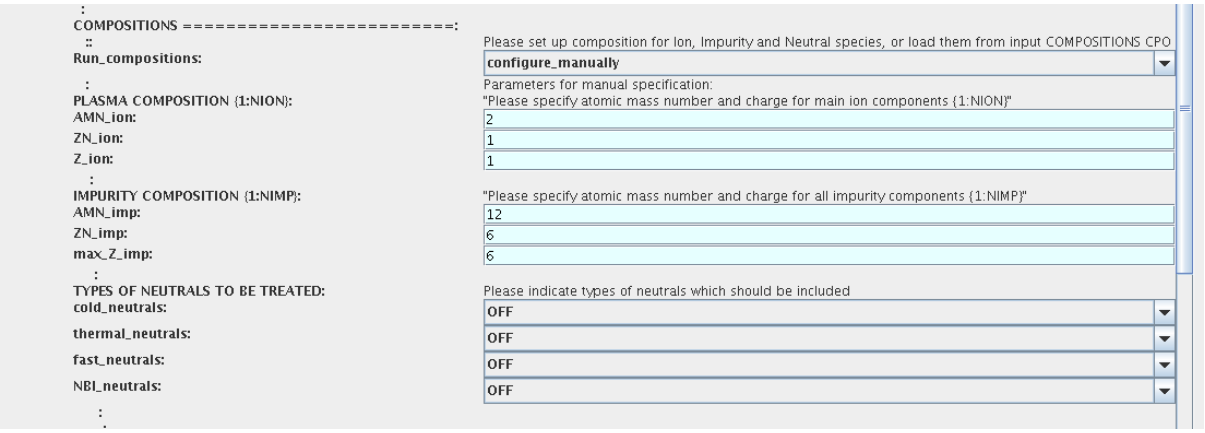
To set up the composition:

- right click on the box 'BEFORE THE TIME EVOLUTION'
- select 'Configure actor'
- choose one of modes for setting "Run_compositions" "from_input_CPO" - will pick up the COMPOSITIONS structure of the COREPROF CPO from the input shot; "configure_manually" - will force the composition from the values specified below
- specify values of AMN_ion, ZN_ion and Z_ion for ions, from the first ion to the last [1:NION], separated by commas
- specify values of AMN_imp, ZN_imp and max_Z_imp for impurity ions, from the first to the last [1:NIMP], separated by commas
- choose the neutrals types, which should be switched "ON"
- Commit

3.6.2.4.3 Equations to be solved and boundary conditions

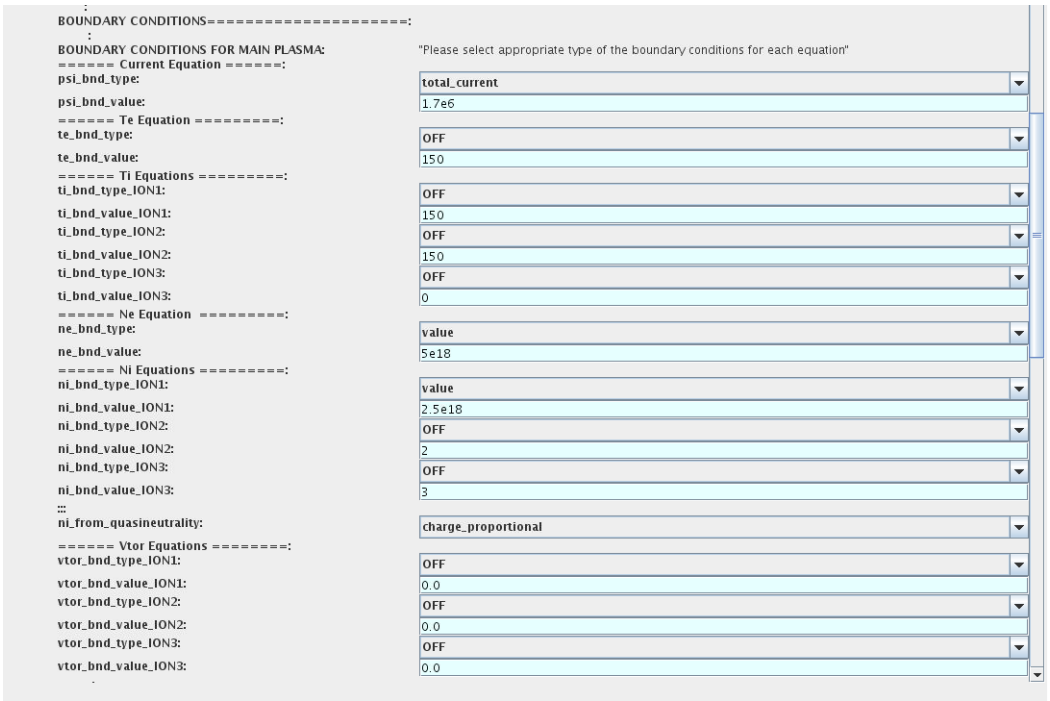
3.6.2.4.3.1 Main plasma

Before starting the run you need to select the type and value of the boundary conditions for all equations. Please note that the value should correspond to the type. All equations allow for following types of boundary conditions:



- OFF - equation is not solved, initial profiles will be kept for whole run
- value - edge value should be specified
- gradient - edge gradient should be specified
- scale_length - edge scale length should be specified
- generic - 3 coefficients (a1,a2,a3) should be provided: a1*y´ + a2*y = a3
- value_from_input_CPO - equation is solved, edge value evolution will be red from input shot
- profile_from_input_CPO - equation is not solved, profile evolution will be red from input shot

The particular equation will be activated if the boundary condition type for it is other than *OFF*!



To set up boundary conditions:

- right click on the box ‘BEFORE THE TIME EVOLUTION’
- select ‘Configure actor’
- select appropriate boundary condition for each equation

- specify values for boundary conditions corresponding to the type and to the ion component
- Commit

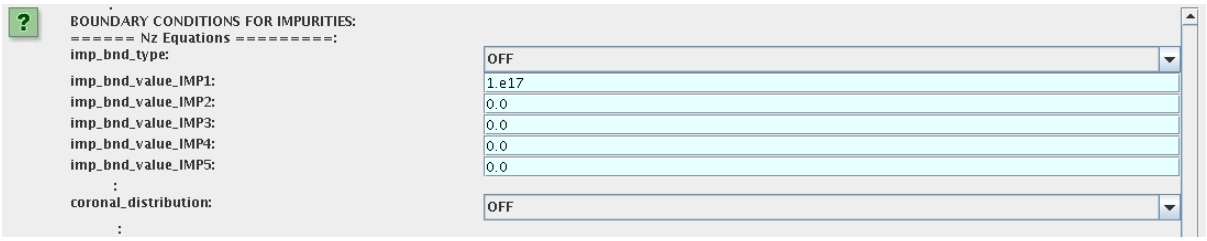
!!! If electron density is solved, all ions with ni_bnd_type=OFF will be computed from the quasineutrality condition and scaled proportional to specified *ni_bnd_value* or inversely proportional to their charge *(charge_proportional)*. This is defined by option: *ni_from_quasineutrality*.

3.6.2.4.3.2 Impurity

You can set up the boundary conditions for impurity ions in a similar way as for main ions. !!! Note, that at the moment only types: *OFF*; *value* and *value_from_input_CPO* are accepter by impurity solver.

To set up boundary conditions:

- right click on the box 'BEFORE THE TIME EVOLUTION'
- select 'Configure actor'
- select appropriate boundary condition for each impurity species (OFF-equation is not solved)
- specify values for boundary density of each impurity component [1:MAX_Z_IMP], separated by commas
- Commit



Interface for impurity boundary condition has additional option , *coronal_distribution*, that allow to preset the edge values or entire profiles of individual ionization states from coronal distribution. In tis case only single value is required to be specified for each impurity boundary value. The options are:

- OFF - the boundary values for impurity densities will be as they are specified above;
- boundary_conditions - the boundary densities will be renormalized with corona, using the first element from above as a total density
- boundary_conditions_and_profiles - the boundary densities and starting profiles will be renormalized with corona, using the first element from above as a total density

3.6.2.4.3.3 Neutrals

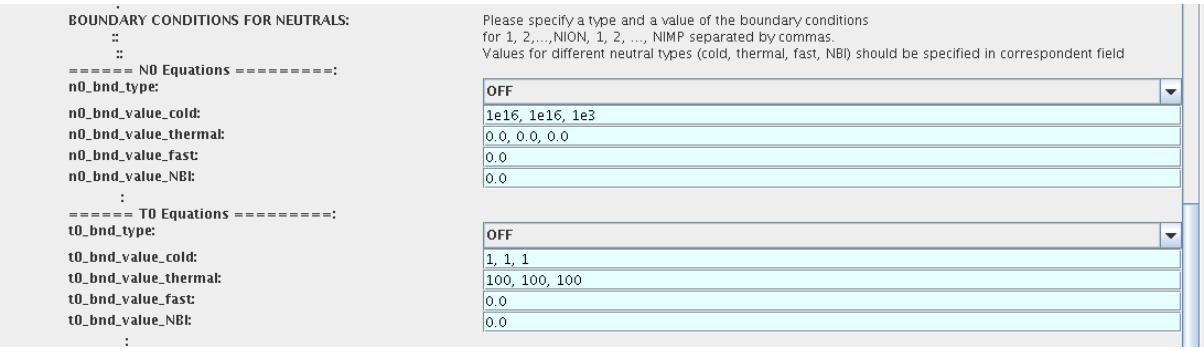
!!! AT THE MOMENT BOUNDARY CONDITIONS FOR NEUTRAL VELOCITIES ARE DISABLED, MIGHT BE ADDED ON REQUEST

Note, that ALL values should be specified in the order: *{1, 2, 3 . . . NION, 1, 2, 3, . . . NIMP}*

To set up boundary conditions:

- right click on the box 'BEFORE THE TIME EVOLUTION'
- select 'Configure actor'

- select appropriate boundary condition for each neutral species (OFF-equation is not solved)
- specify values for boundary density and temperature of each neutral component [1, 2, 3 . . . NION, 1, 2, 3, . . . NIMP], separated by commas
- Commit



3.6.2.4.3.4 Input Profiles Interpolation

You are going to start the ETS run from some input shot, which might contain some conflicting rho grids. Thus there is a choice for the user to decide on the grid on which the starting profiles should be load by the worflow, *Interpolation_of_input_profiles*.

To define the interpolation grid select:

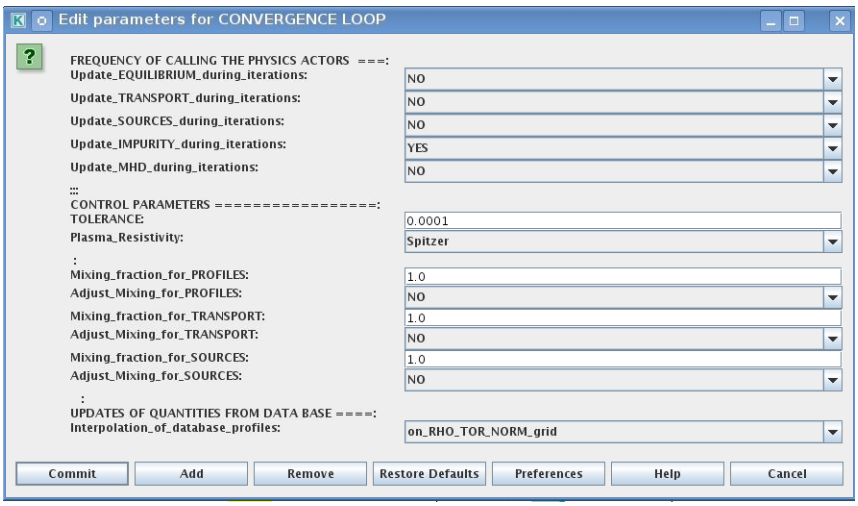
- on_RHO_TOR_grid - interpolate input profiles based on the grid specyfied in [m];
- on_RHO_TOR_NORM_grid - interpolate input profiles based on normalised rho grid [0:1]

3.6.2.4.4 Convergence loop

ETS updates input from different physics actors in a sequence, which is finished by solving the transport equations. Ther are possible none-linear couplings between different parts of the system. These nonelinearities are trited by the ETS using iterations. The decision to step in time is made by the ETS based on the criteria that the maximum relative deviation of main plasma profiles is lower than some predefined tolerance. There is a number of settings and sitches in the ETS that are used by the iterative scheme. To edit them do:

- right click on the box 'CONVERGENCE LOOP'
- select 'Configure actor' to edit settings
- choose your settings
- Commit

Switches in the field *FREQUENCY OF CALLING THE PHYSICS ACTORS* define how many times the the actors of a certain cathegory (equilibrium, transport, etc.) should be called in a single time step. By selecting *YES* all actors of this cathegory will be called every iteration By selecting *NO* all actors of this cathegory will be called only ones in a time step



Switches and parameters in the field *CONTROL PARAMETERS* define how iterations are done

- Tolerance - defines the maximum relative error of profiles change compared to previous iteration. If it is achieved the time steping is done.

For highly none-linear case the required precision can be achieved faster by the iterative scheme if only fraction of the new solution is mixed to the previous state. The following scheme is adopted by the ets to reduce none-linearities in profiles, transport coefficients and sources:

```
Y = (Amix * Y+) + ((1-Amix)*Y-)
```

where Amix is the mixing fraction You can activate the mixing of profiles, transport coefficient and sources by selecting the corresponding *Mixing_fraction_...* to be between [0:1] You also can activate the authomatic ajustment of this fraction by selecting: *Ajust_Mixing_for_...* to *YES*

3.6.2.4.5 Equilibrium

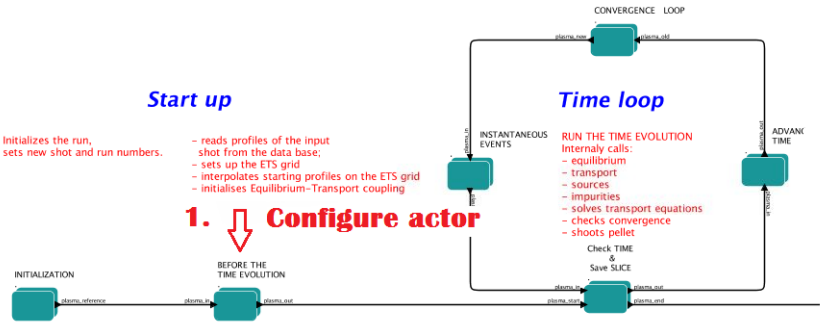
3.6.2.4.5.1 Starting Settings

Before starting the run you need to set up your initial equlibrium. There are several options to do it: if your input shot contains the consistent equilibrium with all necessary parameters - you can start immediately from it; if your input shot contains the equilibrium but it is not consistent or some parameters are missing you can check it automatically; if your input equilibrium is corrupt or not present - you can define the starting equlinbrium by tree moment description. To select your starting equilibrium please do:

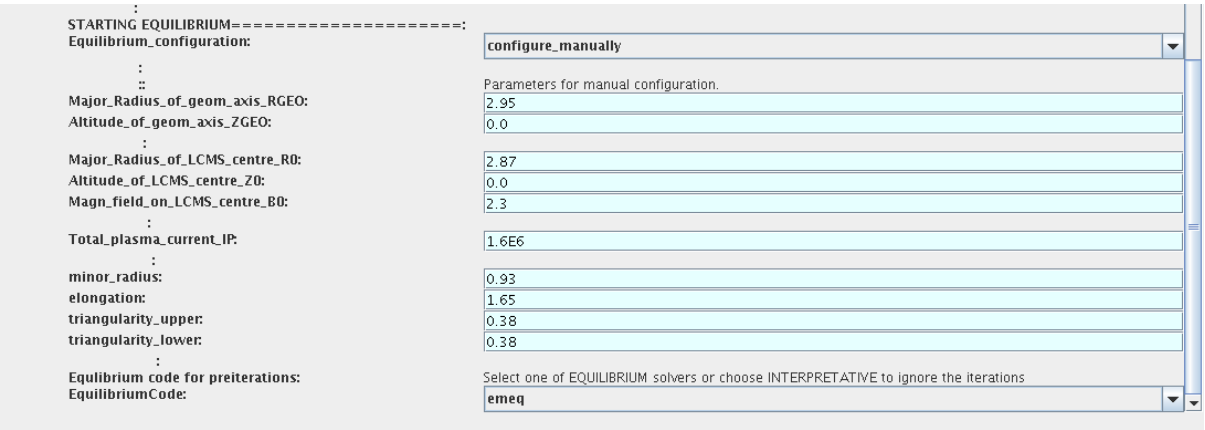
- right click on the box 'BEFORE THE TIME EVOLUTION'
- select 'Configure actor' to edit settings
- Select your settings or specify values
- Commit

SETTINGS:

- Equilibrium_configuration - select configure_manually if you like to specify configuration below; select from_input_CPO if all quantities should be picked up from the input CPO
- Major_Radius_of_geom_axis_RGEO - radius of the geometrical centre of the vessel [m]



- Altitude_of_geom_axis_ZGEO - altitude of the geometrical centre of the vessel [m]
- Major_Radius_of_LCMS_centre_R0 - radius of the plasma centre [m]
- Altitude_of_LCMS_centre_Z0 - altitude of the plasma centre [m]
- Magn_field_on_LCMS_centre_B0 - vacume magnetic field at R0 [T]
- Total_plasma_current_IP - plasma current within the LCMS [A]
- Minor_radius - minor radius of the LCMS [m]
- Elongation - elongation of the LCMS [-]
- Triangularity_upper - upper triangularity of the LCMS [-]
- Triangularity_lower - lower triangularity of the LCMS [-]
- Equilibrium code - select one of available equilibrium solvers to check the consistency between starting equilibrium and current profile; use INTERPRETATIVE if you trust your input data (in this case the check will be ignorred).

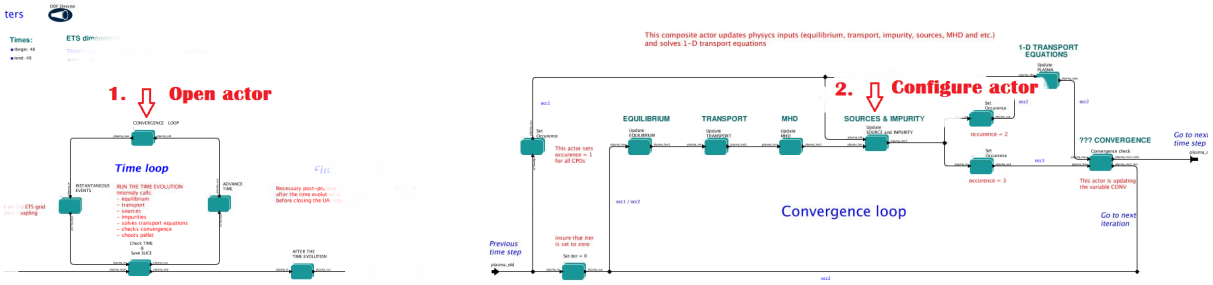


Please note, that different equilibrium solvers might require slightly different input. Thus it is a user responsibility to check that the information inside input shot/run is enough to run selected equilibrium solver.

3.6.2.4.5.2 Run Settings

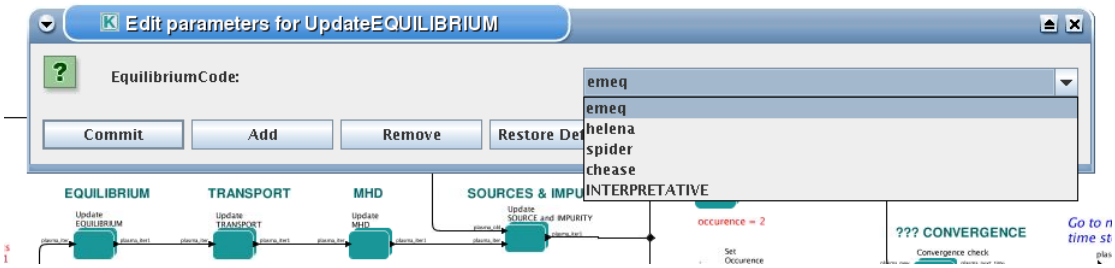
There are several equilibrium solvers connected to the ETS. You can select the one of them.Therefore please do:

- right click on the box 'CONVERGENCE LOOP'
- select 'Open actor'
- right click on the box 'EQUILIBRIUM'
- select 'Configure actor' to edit settings
- choose your equilibrium solver
- Commit



INTERPRETATIVE means that the ETS will not update the equilibrium, instead it will be using the initial equilibrium.

Please note, that it is better to select the same code as you used for pre-iterrations. Because outputs of different equilibrium solver are not necessary done with the same resolution. Therefore the routine saving the information to the data base might brake due to uncompatible sizes of some signals.

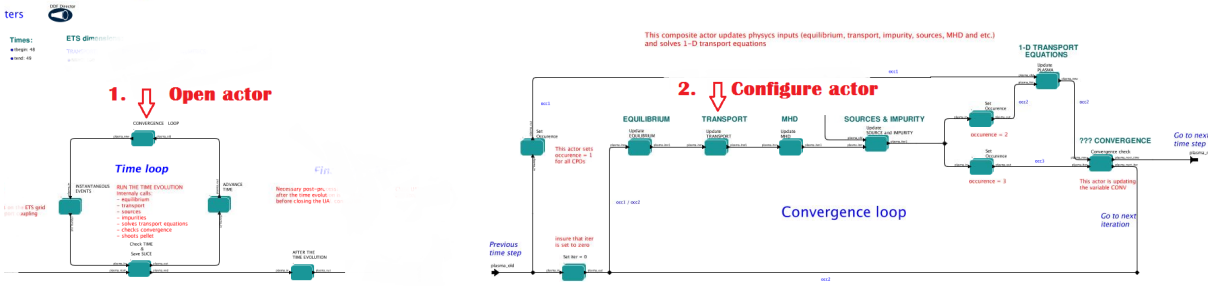


3.6.2.4.6 Transport

The settings for TRANSPORT can be done inside the CONVERGENCE LOOP composite actor. Therefore please do:

- right click on the box 'CONVERGENCE LOOP'
- select 'Open actor'
- right click on the box 'TRANSPORT'
- select 'Configure actor' to edit settings
- choose your settings

- Commit



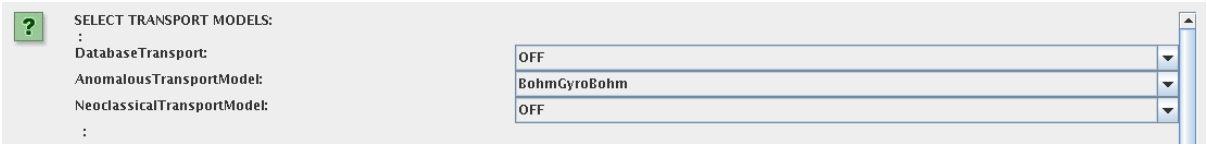
3.6.2.4.6.1 Choice of transport model

ETS constructs the total transport coefficients from the combination of Anomalous transport (model choice), Neoclassical transport (model choice) and Database transport (transport coefficients be saved to the input shot)

```
D_tot = D_DB*M_DB + D_AN*M_AN + D_NC*M_NC
```

You should choose from the list of available models in each cathegory or switch it OFF

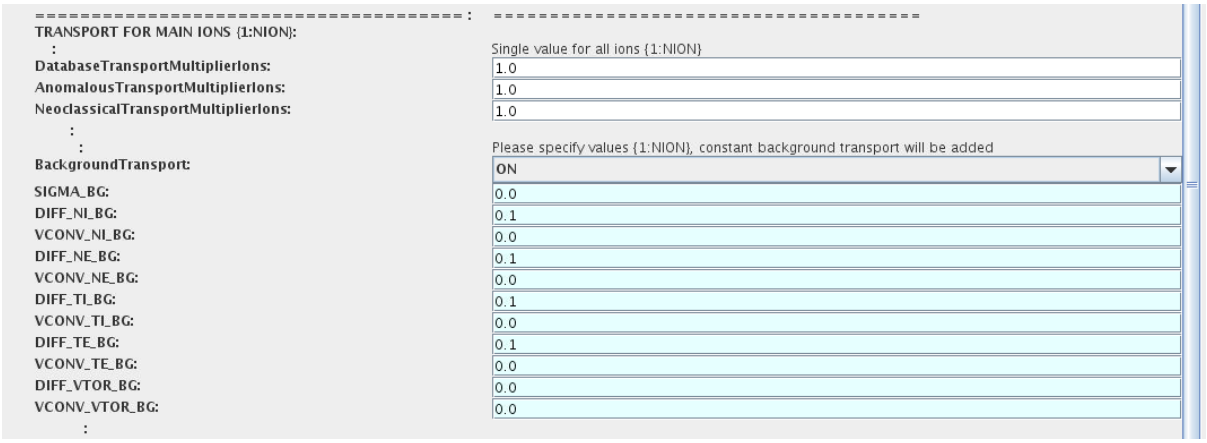
The list of available transport models can be found here.



3.6.2.4.6.2 Main plasma transport

In this section you define how total transport coefficients for main ions should be constructed from contributions provided by different models. You need to provide the multipliers for Anomalous, Neoclassical and Database contributions, which will determine their weights in total transport coefficient.

You also can add the constant background level for each coefficient (ion coefficients are expected to be the string {1:NION}, separated by commas)

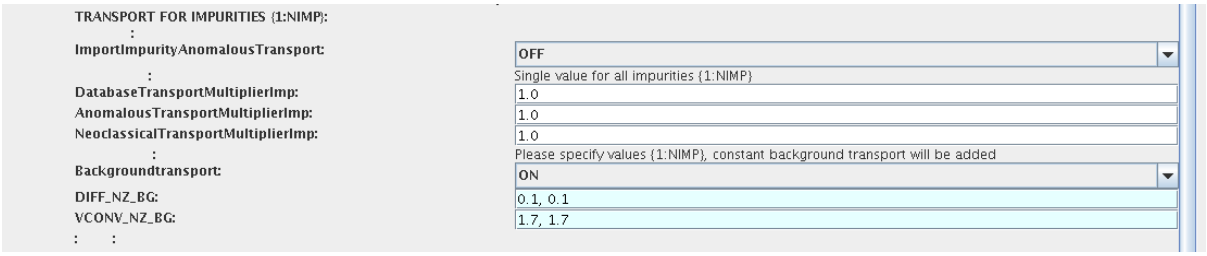


3.6.2.4.6.3 Impurity transport

In this section you define how total transport coefficients for impurity ions should be constructed from contributions provided by different models. You need to provide the multipliers for Anomalous, Neoclassical and Database contributions, which will determine their weights in total transport coefficient.

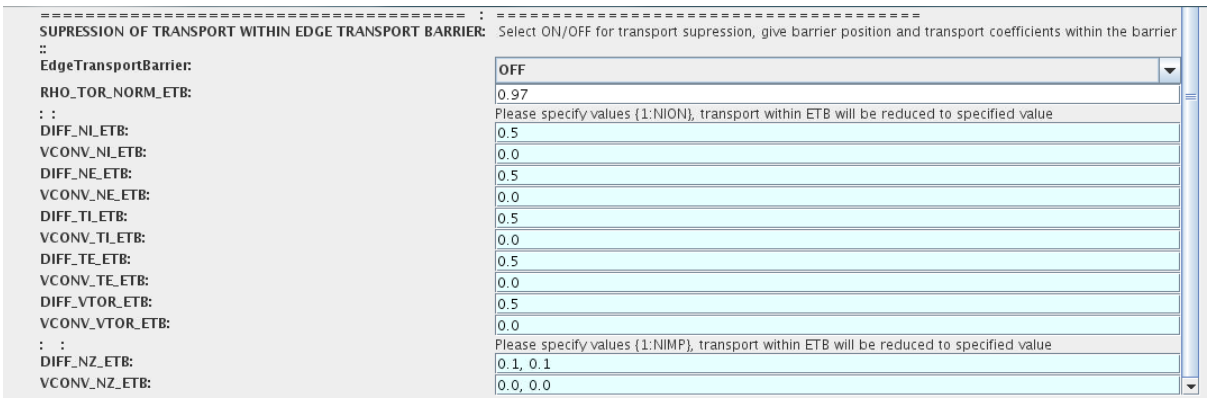
You also can add the constant background level for each coefficient (coefficients are expected to be the string {1:NIMP}, separated by commas)

In addition, there is an option to import the Anomalous component of transport coefficient *from_first_ion* or *from_electrons* (the same anomalous contribution will be added to all impurity components, all ionization states)



3.6.2.4.6.4 Edge transport barrier

In this section you can artificially supress the transport outside of specified *RHO_TOR_NORM_ETB*. Total transport coefficients for all transport channels (ne, ni, nz, Te, Ti,...) will be reduced to constant values specified below (ion and impurity coefficients are expected to be the strings {1:NION}) and {1:NIMP} respectively)



3.6.2.4.6.5 Total transport coefficients

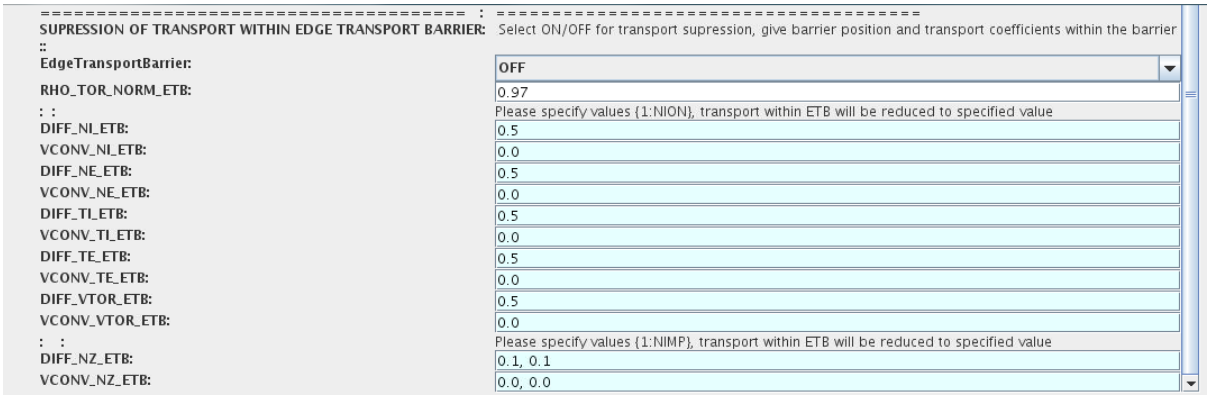
Profiles of the total transport coefficient for each channel are obtained from the the individual contributions (Data Base, Anomalous, Neoclassical and Background) as a summ of all activated transport models multiplied with coefficients specified on the interface of the composite actor.

```
X_tot = X_DB*DBM + X_AN*ANM + X_NC*NCM + X_BG*BGM
```

!!! Note, that contributions to all transport equations will be multiplied with the same value. For example: if AnomalousTransportMultiplier=3.0, then contibutions from selected anomalous transport model to each transport equation will be multiplied with 3.0

The fine tuning of of transport coefficients can be done through editing the XML code parameters of the transport combiner actor:

- right click on the box 'TRANSPORT'
- select 'Open actor' to edit settings
- right click on the box 'Transport Combiner'
- select 'Open actor' to edit settings
- right click on the box 'transportcombiner'
- select 'Configure actor'
- click 'Edit Code Parameters'
- If you select *OFF* contributions from all transport models to this channel will be nullified; If you select *from_input_CPOs* the transport channel will be activated, and the total transport coefficient will be combined from active tranport models; For convective velocity there is an additional option *fixed_V_over_D_ratio*, by selecting this the combiner will ignore the convective components provided by transport nmodels. The convective velocity will be determined from the total diffusion coefficient by applying fixed V/D ratio (*for inward pinch the values should be negative!*). For all active channels you can adjust multipliers for combining contributions from different transport models (array of four space separated values is expected):
  - first position - Data Base transport coefficients;
  - second position – Anomalous transport coefficients;
  - third position – Neoclassical transport coefficients;
  - fourth position – Background (constant level) transport coefficients;
- save and exit
- Commit

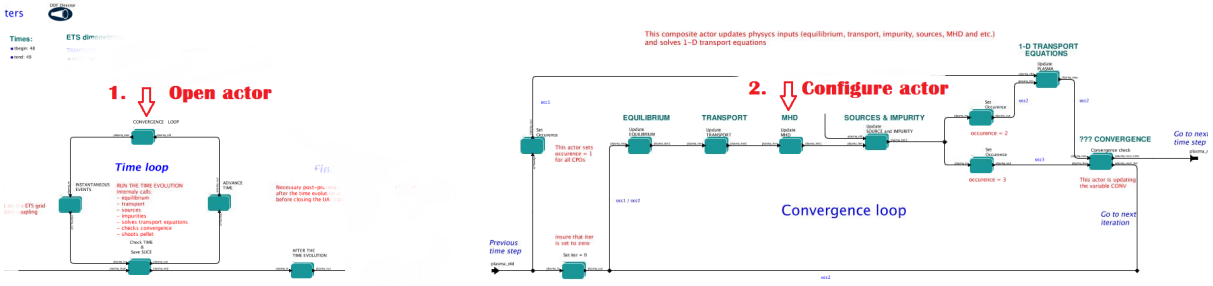


3.6.2.4.7 MHD

The settings for MHD type of events can be done inside the CONVERGENCE LOOP composite actor. Therefore please do:

- right click on the box 'CONVERGENCE LOOP'
- select 'Open actor'

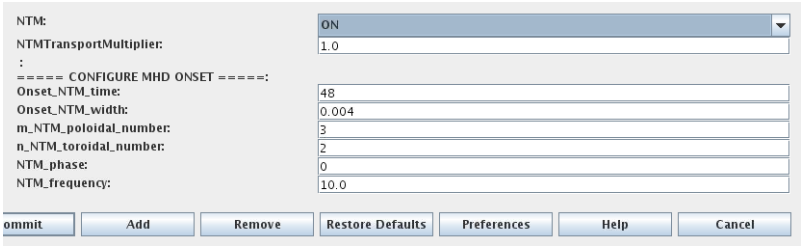
- right click on the box ‘MHD’
- select ‘Configure actor’ to edit settings
- choose your settings
- Commit



At the moment ETS allows only for NTM to be activated.

User can ajust the following NTM settings:

- NTM – ON means that ETS will add the NTM driven transport to the total transport coefficient; OFF-ignored
- NTMTransportMultiplier – the transport contrinution from NTM will be multiplied with this value
- Onset_NTM_time - activation time for the NTM mode
- Onset_NTM_width - starting width of the mode
- m_NTM_poloidal_number
- n_NTM_toroidal_number
- NTM_phase
- NTM_frequency

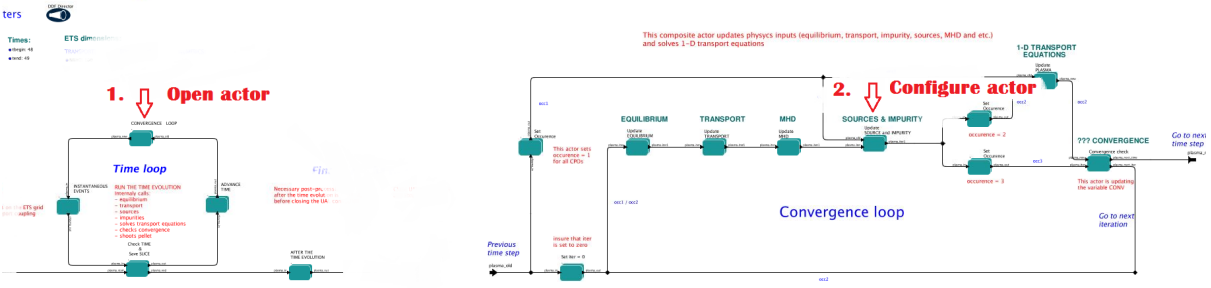


3.6.2.4.8 Sources and impurity

The settings for SOURCES AND IMPURITY can be done inside the CONVERGENCE LOOP composite actor. Therefore please do:

- right click on the box ‘CONVERGENCE LOOP’
- select ‘Open actor’
- right click on the box ‘SOURCES AND IMPURITY’
- select ‘Configure actor’ to edit settings

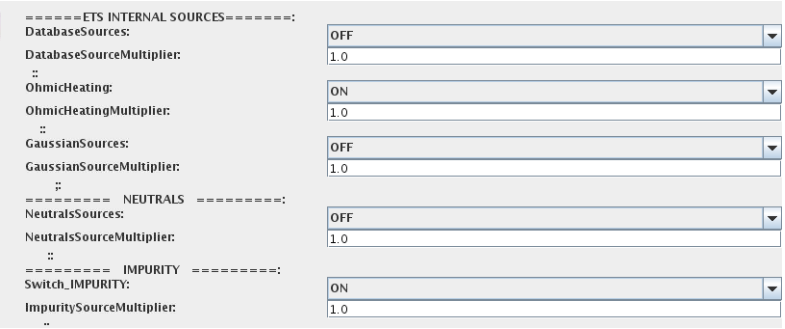
- choose your settings
- Commit



3.6.2.4.8.1 IMP3 sources

There is a number of sources developed by IMP3 project, which are actors or internal routines of the transport solver. You can activate them by selecting *ON / OFF* in front of corresponding source:

- Database Sources – ON - ETS will pick up the evolution of source profiles saved to your input shot/run; OFF -ignored
- Ohmic Heating – ON - ETS will compute Ohmic heating internaly; OFF-ignored
- Gaussian Sources – ON - ETS will add sources from the Gaussian source actor (you can configure heat and particle deposition profiles by editing the code parameters of the actor); OFF-ignored
- Neutral Sources– ON - Fluid neutrals will be solved according to the boundary conditions specified on ¨Before_time_evolution¨ composite actor interface; OFF -ignored
- Switch_IMPURITY– ON - Impurity density and radiative sources will be computed; OFF - ignored; INTERPRETATIVE – profiles of impurity density will be read from input shot/run



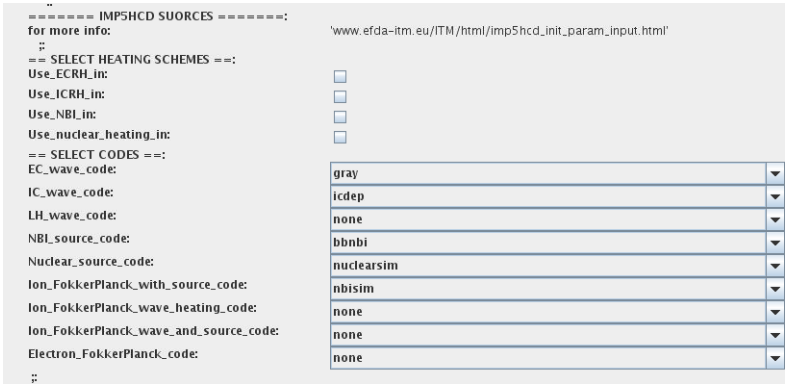
3.6.2.4.8.2 IMP5HCD sources

There is a number of sources developed by IMP5 project, that are incorporated by the ETS workflow.

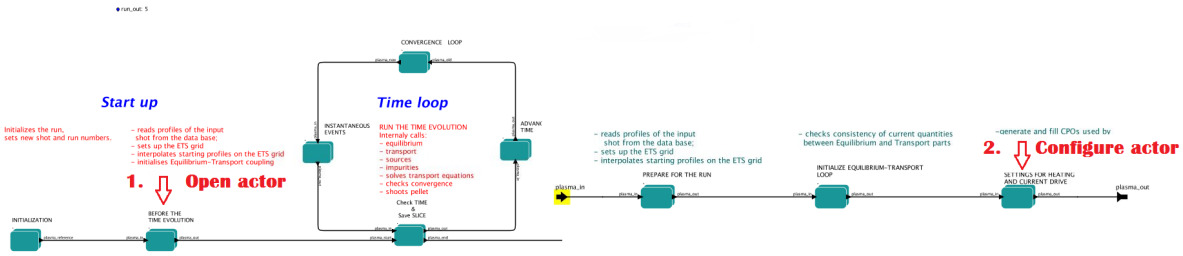
For the IMP5HCD sources please activate the type of heating source, by ticking the box in front of it, and select the code to simulate it.

You also need to configure initial IMP5HCD settings. Therefore please:

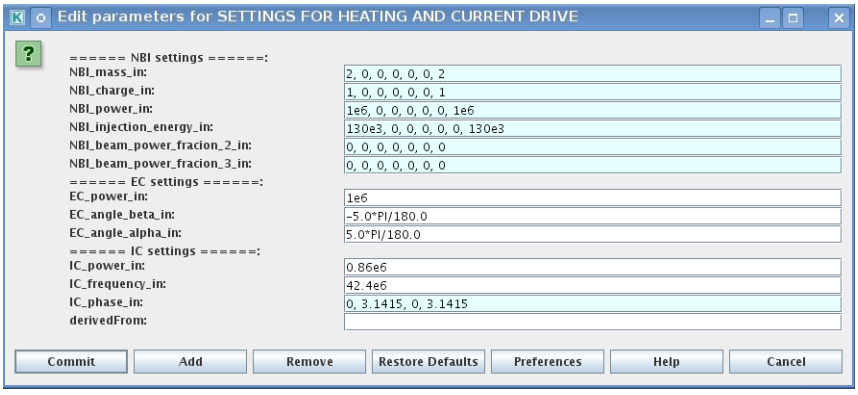
- right click on the box ‘BEFORE THE TIME EVOLUTION’



- select 'Open Actor'
- right click on the box 'SETTINGS FOR HEATING AND CURRENT DRIVE'
- select 'Configure actor'
- edit the stettings
- Commit



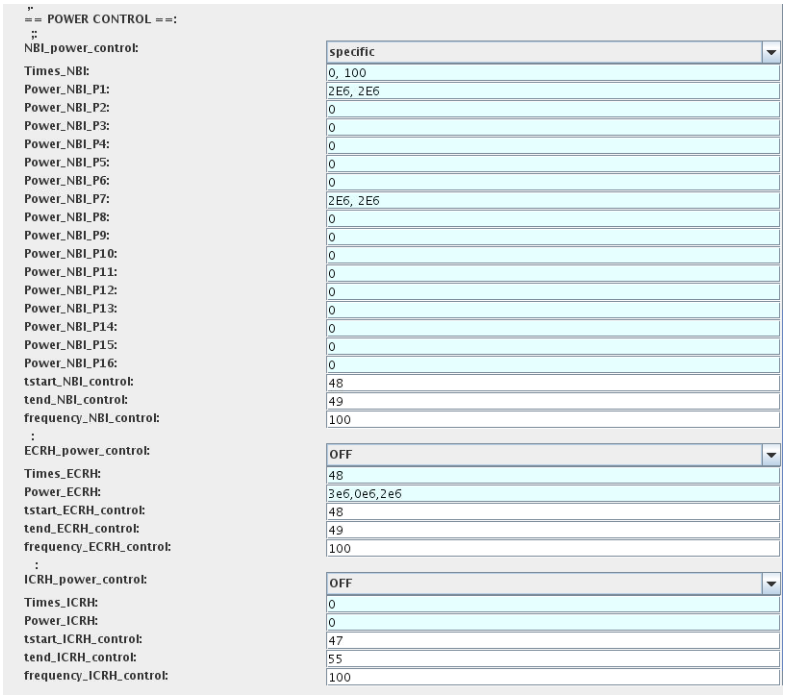
The detailed information on initial IMP5HCD settings can be found here. Please note that settings for NBI are done independent for each PINI. Therefore, for NBI settings, please insert the values separated by commas. The number of the element in the array corresponds to the number of activated PINI. Maximum accepted number of PINIs = 16.



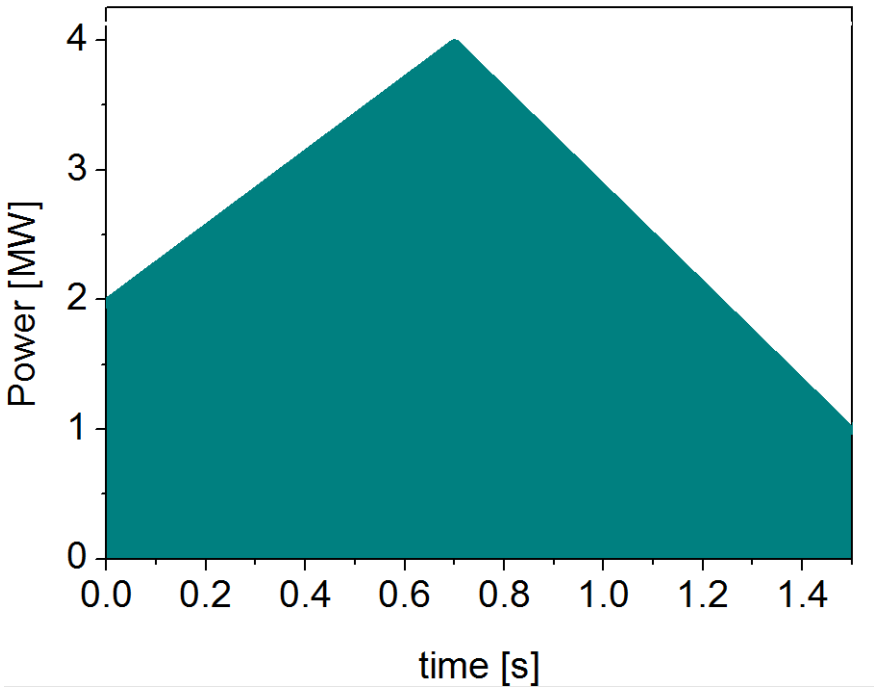
3.6.2.4.8.3 Power control

You also can activate the power control for the IMP5HCD sources.

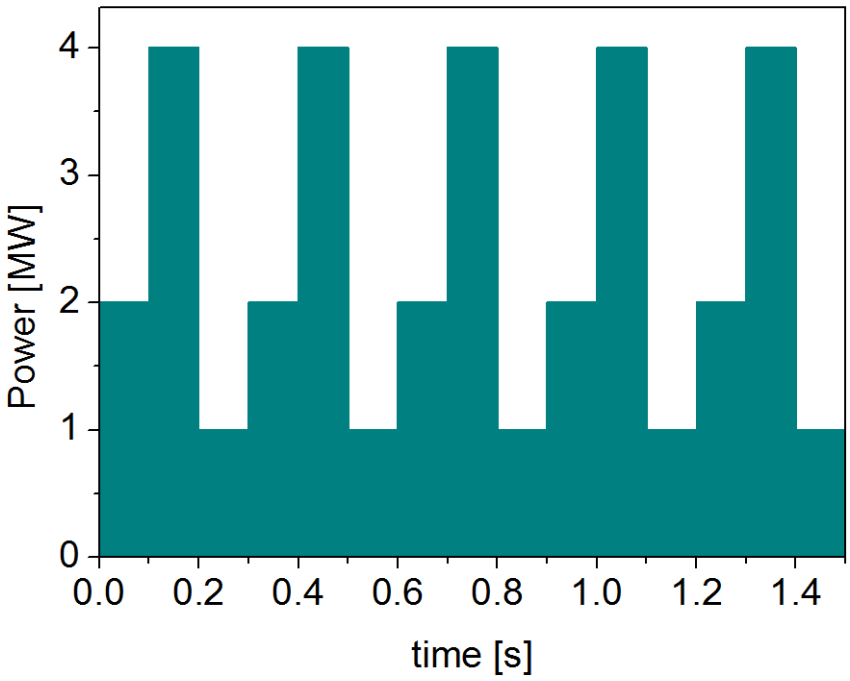
If the POWER_CONTROL is not *OFF*, there are two modes of operation:*specific* and *frequency*



For *specific* you should specify the time sequence separated by commas and the corresponding power sequence (where first power level corresponds to the first time, second to second and etc.). Linear interpolation will be done between the sequence points. For example: if you give the power sequence = 2e6,4e6,1e6 and times = 0.0, 0.7, 1.5 (s) the delivered power would be:



For *frequency* you should specify the power levels sequence separated by commas, start and end time of the power control and the frequency of switching between these levels. For example: if you give the power sequence = 2e6,4e6,1e6 and frequency = 10 (Hz) tstart =0.0 (s) tend = 1.5 (s) the delivered power would be:



3.6.2.4.8.4 Total power

Profiles of the total source for each channel are obtained from the the individual contributions (Data Base, Gaussian, Neutrals, Impurity and HCD) as a summ of all activated sources multiplied with coefficients specified on the interface of the composite actor.

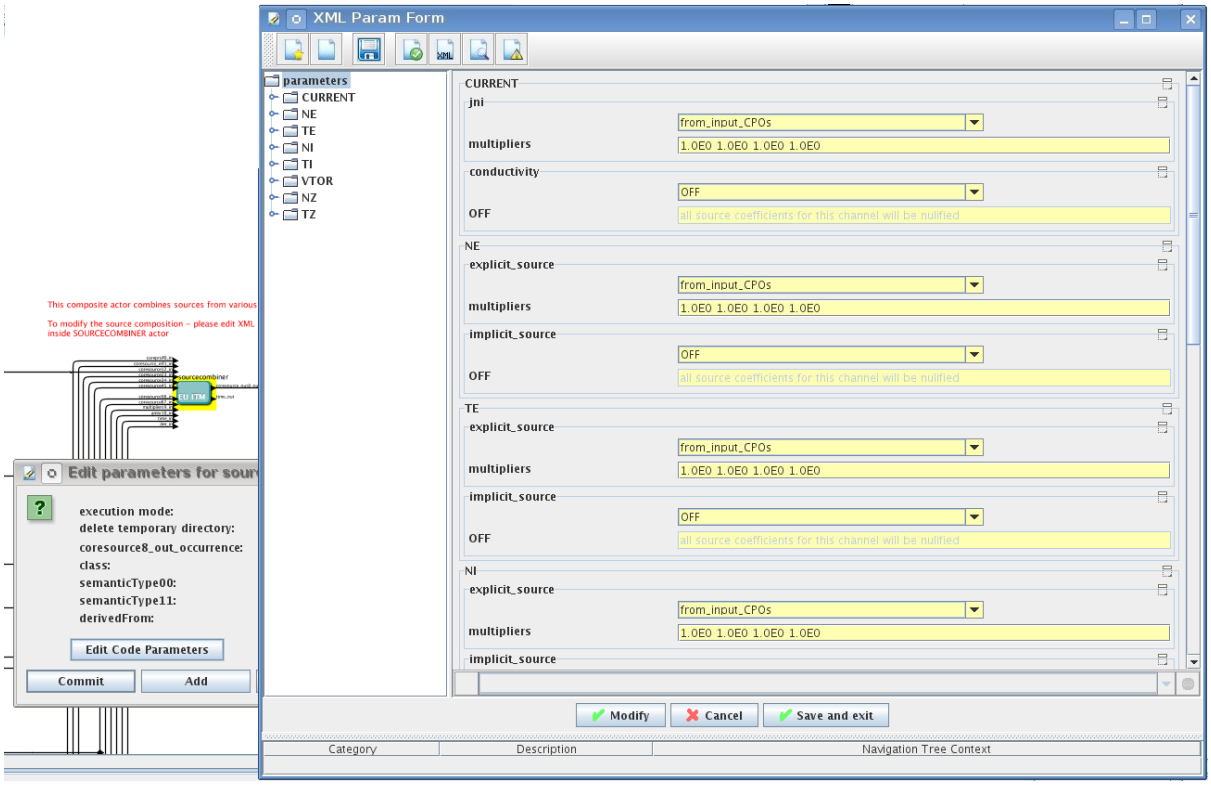
```
S_tot = S_DS*DSM + S_GS*GSM + S_Neu*NeuSM + S_IMP*IMPSM + S_HCD*HCDSM
```

!!! Note, that contributions to all transport equations will be multiplied with the same value. For example: if ImpuritySourceMultiplier=3.0, then contibutions from impurity to Se, Sz and Qe will be multiplied with 3.0

The fine tuning of of sources can be done through editing the XML code parameters of the source combiner actor:

- right click on the box 'SOURCES and IMPURITY'
- select 'Open actor' to edit settings
- right click on the box 'Source Combiner'
- select 'Open actor' to edit settings
- right click on the box 'sourcecombiner'
- select 'Configure actor'
- click 'Edit Code Parameters'
- If you like the sources to the particular equation being activated - select *from_input_CPOs*; if you select *OFF* contributions from all sources to this channel will be nullified. For active channels you can adjust multipliers for combining contributions from different source modules (array of five space separated values is expected):
  - first position - Data Base sources;

  - second position – Gaussian sources;
  - third position – HCD sources;
  - fourth position – Neutral sources;
  - fifth position – Impurity sources.
- save and exit
- Commit



3.6.2.4.9 Instantaneous events

At the moment, user can swith ON and OFF two types of events: PELLET and SAWTOOTH

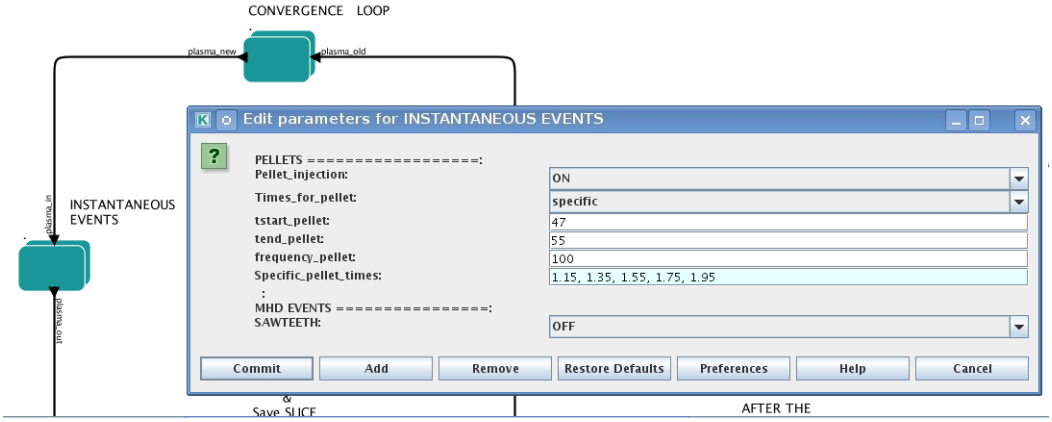
3.6.2.4.9.1 Pellet

At the top level of the workflow you can configure times for pellet injection

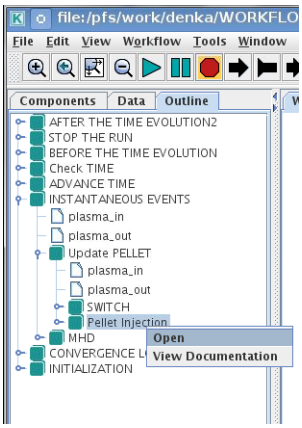
- right click on the box ‘INSTANTANEOUS EVENTS’
- select ‘Configure actor’ to edit settings
- Select Pellet_injection ‘ON’ if you like to use pellet in your simulation
- Select mode of operation: ‘specific’ - pellets will be shut at specific times, you also need to specify array ‘times_pellet’
  - ‘specific’ - pellets will be shut at exact times specified in array ‘times_pellet’

  - 'frequency' – pellets will be shut from 'tstart_pellet' until 'tend_pellet' with a 'frequency_pellet'

- 'frequency' – pellets will be shut from 'tstart_pellet' until 'tend_pellet' with a 'frequency_pellet'
- Commit



Parameters of individual pellet need to be configured through the icode_parameters of the PELLET actor. To access it go to 'Outline' on the right upper corner and open the following:



- right click on the actor 'PELLET'
- select 'Configure actor'
- click 'Edit Code Parameters'
- edit parameters and click 'save and exit'
- Commit

amn – atomic mass number: array of elements separated by space (1:nelements) [-]

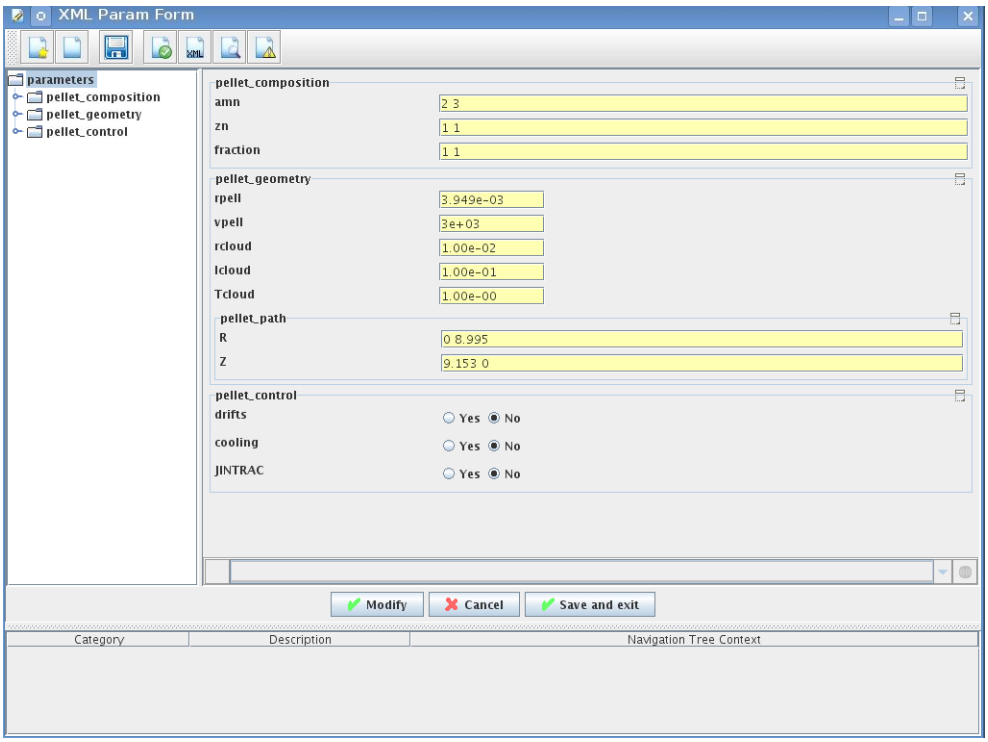
zn – nuclear charge: array of elements separated by space (1:nelements) [-]

fraction – fraction of each element in the pellet, based on the number of atoms: array of elements separated by space (1:nelements) [-]

rpell – radius of the pellet [m]

vpell – velocity of the pellet [m/s]

rcloud – radius of the pellet cloud [m], radial extension of the cloud = 2*rp0



lcloud – length of the pellet cloud along the field line [m]

Tcloud – temperature of the pellet cloud [eV]

Pellet path is specified by two points, for which R and Z coordinated should be specified

R – R coordinates of the pivot and second points of the pellet path, separated by space [m]

Z – Z coordinates of the pivot and second points of the pellet path, separated by space [m]

Control switches allow to activate:

- drifts - YES - will activate radial displacement of deposition profile, same for all path points
- cooling - YES - will activate cooling of the other side of the plasma due to parallel heat transport (essential for large pellets, which might cross the same flux surface twice)
- JINTRAC - YES - will provide temperature reduction consistent with the model used in JETTO

3.6.2.4.9.2 MHD

At the top level of the workflow you can switch ON/OFF possible MHD events

- right click on the box 'INSTANTANEOUS EVENTS'
- select 'Configure actor' to edit settings
- Select SAWTOOTH 'ON' if you like to use them in your simulation
- Commit

3.6.2.4.10 Visualization during the run

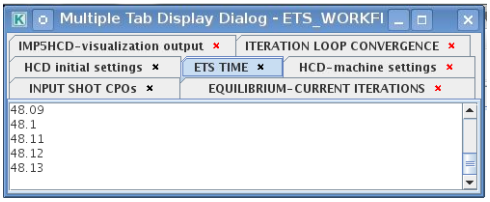
There is a number tools visualizing the ETS run.

3.6.2.4.10.1 Multiple Tab Display

The display appeares automaticaly when the ETS workflow is launched. It displays diagnostic text messages from the workflow on following topics:

- Input data statement
- Iterations to check the initial convergence between EQUILIBRIUM and CURRENT
- Time evolution
- Convergence of iteratinos within the time step
- IMP5HCD settings
- Power used by IMP5HCD actors durung the run

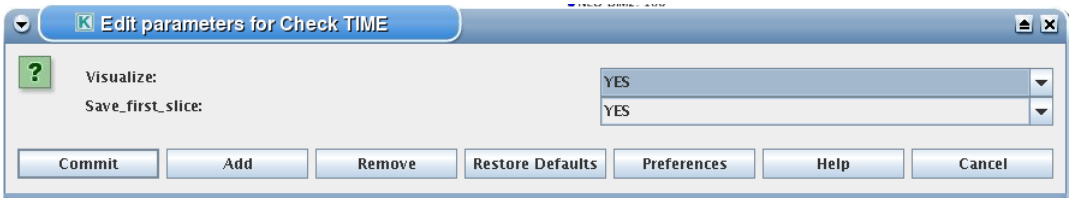
Also the error messages from execution of the workflow will be displayed here.



3.6.2.4.10.2 Python Visualization Display

You can activate the graphical visualization of your run evolution:

- right click on the box 'Check Time & Save Slice'
- select 'Configure actor'
- select visualisation 'YES' or 'NO'
- Commit



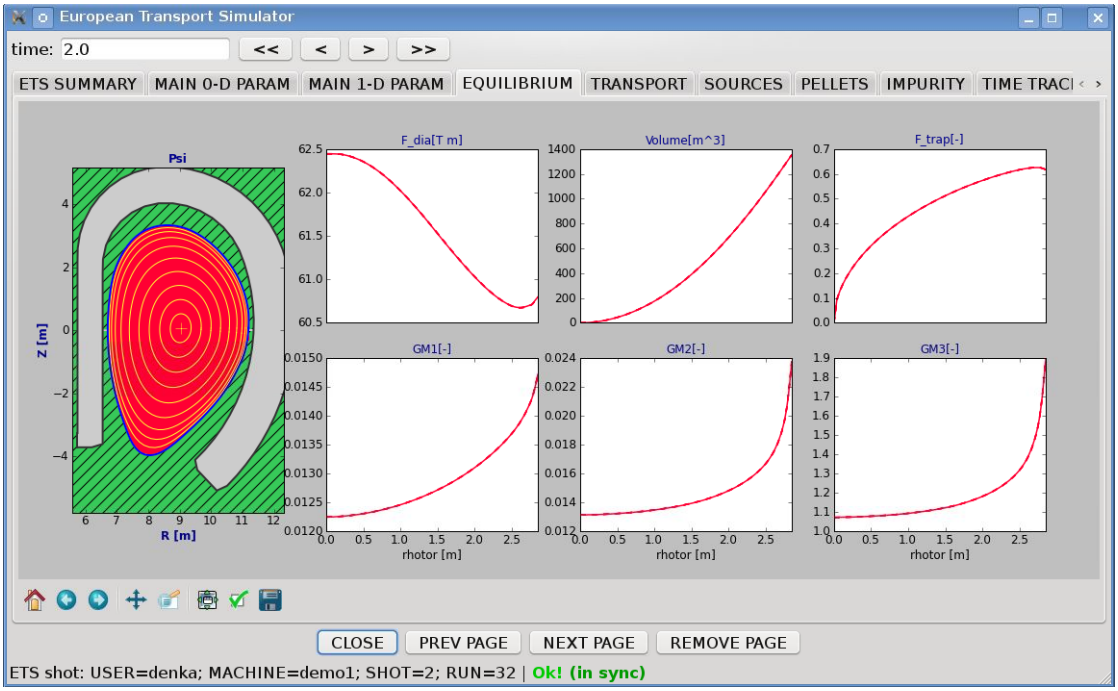
Then evolution of main discharge parameters will be shown in this window:

3.6.3 ETS_C

The ETS workflow (IMP3-ACT1) is used for 1-D transport simulation of a tokamak core plasma.

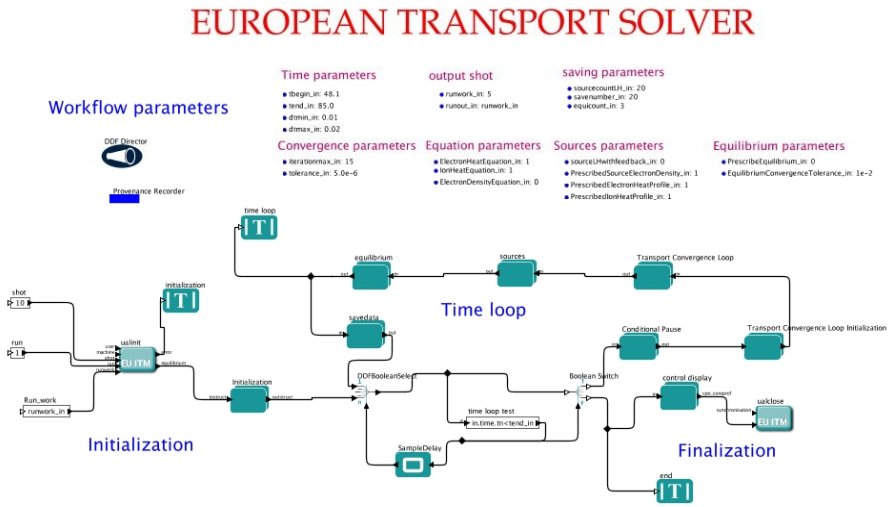
ETS workflow in KEPLER:

- uses as actors and composite actors from other IMPs, thus for the most recent versions of them please check with relevant project
- complex, but clearly structured workflow, which offers user friendly interface for configuring the simulation



- **allows for easy modifications (connecting new modules, or reconnecting** the parts of the workflow) through the easy graphical interface
- provides users with all updates through the version control system
- still actively developing tool

Contact persons: Vincent Basiuk , Philippe Huynh



3.6.4 ETS Status

Package Name / Physics Module	ETS-A	ETS-C
EQUILIBRIUM		
fixed boundary:		
BDSEQ	Ready for use	
EMEQ	Ready for use	
SPIDER	Ready for use	
SPIDER_IMP12	Ready for use	
CHEASE	Ready for use	validate
HELENA	Ready for use	
HELENA21		work in 4.09a problem when it doesn't find any equilibrium crash
free boundary:		
CEDRES++	In progress/tests are planned for Nov.2014	validate (static mode, TBD evolution mode)
CREATE-NL		
FIXFREE		
EQFAST		work in 4.09a
FREEBIE		validate
MHD		
NTM	Ready for use	validate
SAWTEETH	Implemented/Tested/ release date:Nov.2014	
Linear Stability Chain	Stand alone tests/implementation in ETS and release:2015	
TRANSPORT		
analytical & interpretative:		

Continued on next page

Table 3.3 – continued from previous page

Package Name / *Physics Module*	ETS-A	ETS-C
From DATA BASE (interpretative)	Ready for use	
Edge Transport Barried (analytical)	Ready for use	
anomalous:		
ETAIGB	Ready for use	
BOHM-GYROBOHM	Ready for use	validate, + effect of rotation
GLF23	Implemented/Tested/ release date:Nov.2014	to be tested (GLF23 installed in previous gateway not validated)
WEILAND	Implemented/Tested/ release date:Nov.2014	
REU-IM	Implemented/Tested/ release date:Nov.2014	
EWDM	Implemented/Tested/ release date:Nov.2014	
TGLF	In progress/Some initial tests	

Continued on next page

Table 3.3 – continued from previous page

Package Name / Physics Module	ETS-A	ETS-C
KIAUTO		installed (transport model based on scaling law)
neoclassical:		
NEOS	Ready for use	
NEOWES	Ready for use	
NEOART	Ready for use (probably not suggested as being too oscilatory)	
NCLASS	In progress	validate with composition (to be upgrade with compositions
NCLASS/FORCEBALL		installed (gives the radial electric field)
HEAT,PARTICLE SOURCES & CURRENT DRIVE		
analytical & interpretative:		
From DATA BASE (interpretative)	Ready for use	
Gaussian	Ready for use	

Continued on next page

Table 3.3 – continued from previous page

Package Name / Physics Module	ETS-A	ETS-C
impurity and particles:		
IMPURITY	Ready for use	
NEUTRALS	Ready for use	
PELLET	Ready for use	
ZNEUTRES		installed (simple module of CRONOS for neutral source terms)
ZRECYCLE		edge boundary for electron density
ECRH		
GRAY	Ready for use	Installed
TORAY-FOM		In preparation
TRAVIS	Tested	In preparation
TORBEAM		In preparation
ICRH		
TORIC	In progress	In preparation
ICDEP		Installed
FPSIM		Installed
NBI		
NEMO	Ready for use	Installed
BBNBI	Ready for use	In preparation
NBISIM	Ready for use	Installed
ASCOT	Ready for use	
RISK	Ready for use	In preparation
LH		
nuclear sources		
nuclearsim	Ready for use	Installed
CONTROLS		
NBI power control	Ready for use	
ECRH power control	Ready for use	

Continued on next page

Table 3.3 – continued from previous page

Package Name / *Physics Module*	ETS-A	ETS-C
ICRH power control	Ready for use	
Pellet frequency control	Ready for use	
COUPLING TO EDGE		
SOLPS	Tested at Fortran level	
DOCUMENTATION and MANUALS		
Physics Description	Description of the ETS	
Numerics Description	Form of the standardize equations	
Manuals	- ETS workflows in KEPLER - ETS source in Fortran	

CHAPTER

FOUR

EDGE TRANSPORT SIMULATOR

The goal of this work is to adopt the edge code like SOLPS-B2 to be used within the EU-IM platform.

4.1 IMP3 General Grid Description and Grid Service Library

4.1.1 Resources

- GForge project page
- Linking to library: general , specific
- A tutorial talk. Note: some slides might be out of date, please refer to the documentation.

4.1.2 Documentation

- 4.09a Resources: Sources, Fortran Examples

  Documentation:

  - Release v1.2: Fortran 90 , Python , ualconnector ,
- 4.10a Resources: Sources, Fortran Examples

  Documentation:

  - Release v1.2: Fortran 90 , Python , ualconnector ,

4.1.3 Outdated documentation

This section collects information and documentation related to the general grid description.

- Some presentations:
  - A tutorial talk from 2011 ,
  - General Meeting 2011: Short overview talk and detailed presentation
- Instructions how to get a copy of the Grid Service Library
- Documentation for the EU-IM Grid Service Library: Fortran 90 , Python
- A short manual for ualconnector and VisIt

Some examples are included in the Grid Service Library distribution.

4.1.3.1 Example grids

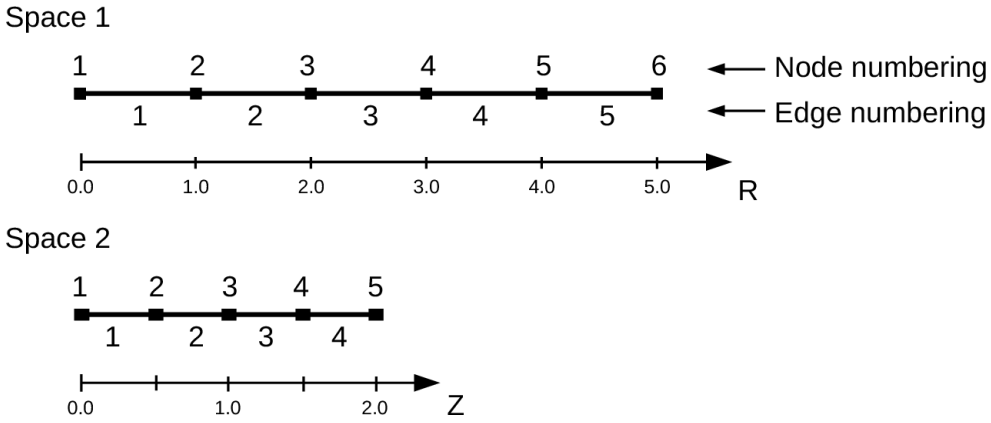
4.1.3.1.1 Example grid details

This section describes a number of example grids and gives some examples for specific constructs (object lists, subgrids).

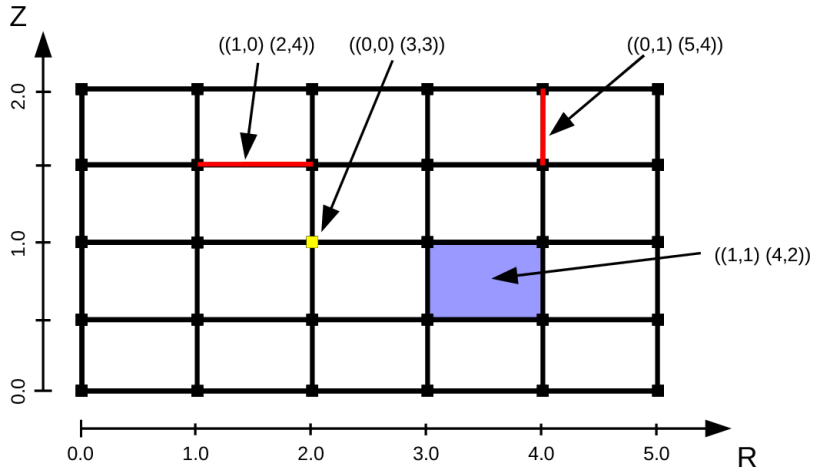
4.1.3.1.1.1 Example Grid #1: 2d structured R,Z grid

Note: the grids shown here are used in the unit tests of the grid service library implementation, i.e. the automated testing framework.

A 2d grid in (R,Z) constructed by combining two structured one-dimensional spaces. The spaces are defined as follows, they define nodes and edges as subobjects.



The whole grid then looks like this (attention, slightly differing scales in R and Z):



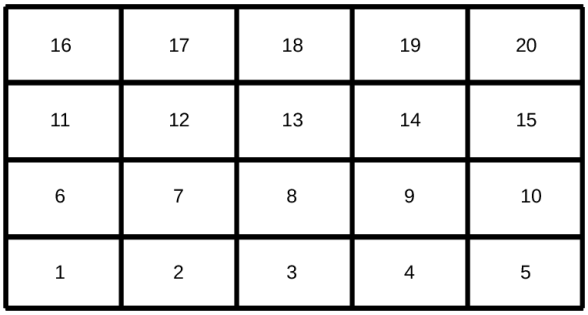
A couple of examples for object descriptor are given. Some explanations:

((1,1) (4,2)) = a 2d object (2d cell or face), implicitly created by combining the 1d object (edge) no. 4 from space 1 and the 1d object no. 2 from space 2. ((1,0) (2,4)) = a 1d object (edge), implicitly created by combining 1d object (edge) from space 1 with the 0d object (node) no. 4 from space 2. ((0,0) (2,2)) = a 0d object (node), implicitly created by combining 0d objects (nodes) no. 2 from space 1 and no. 2 from space 2.

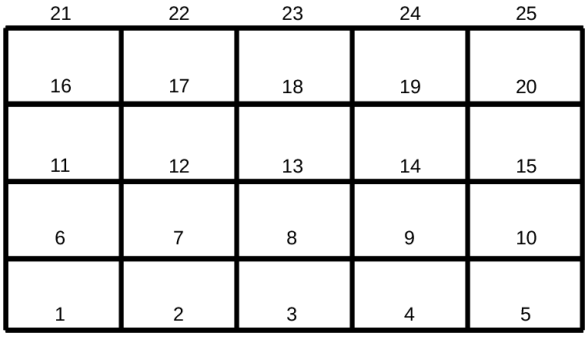
4.1.3.1.1.2 Object classes

This section shows the different object classes present in the grid. The implicit numbering of the objects in a class is obtained by iterating over all subobjects defining the objects, lowest space first.

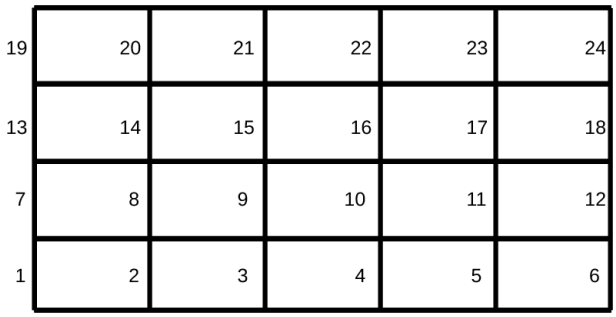
Object class (1,1): 2d cells/faces. They have the following implicit numbering:



Object class (1,0): 1d edges, aligned along the R axis ("r-aligned"). They have the following implicit numbering:



Object class (0,1): 1d edges, aligned along the Z axis ("z-aligned"). They have the following implicit numbering:

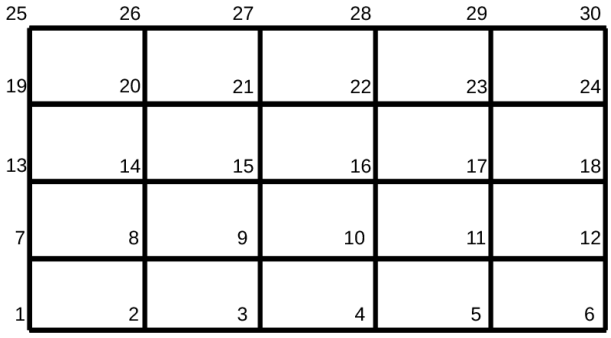


Object class (0,0): 0d nodes. They have the following implicit numbering:

4.1.3.1.1.3 Example 2: B2 grid

4.1.3.1.2 Object list examples

Some examples for object lists, to explain the concept and show the notation. All examples refer to the 2d structured R,Z example grid #1 given above. Object descriptor A single object (= and object descriptor), for object with object class (1,1), object index (4,2).




```
((1,1) (4,2))
```

Explicit object lists An explicit object list is simply an enumeration of object descriptors. The ordering of the objects is given directly by their position in the list. Note that by definition, all objects in the list must be of the same class (An implementation of an explicit object list should enforce this. If you need lists of objects with differing class, have a look at subgrids).

An explicit list of 2d cells (faces), listing the four corner cells of the grid in the order bottom-left, bottom-right, top-left, top-right:

```
(((1,1) (1,1)),
((1,1) (5,1)),
((1,1) (1,4)),
((1,1) (5,4)))
```

Implicit object lists Implicit object lists use the implicit order of (sub)objects to form an efficient representation of (possibly large) sets of objects. They thus avoid explicit enumeration of individual objects as done in the explicit objects lists. The following examples demonstrate the implicit list notation. Note: the implicit list notation is used in the Python implementation of the grid service library in exactly the form given here.

Selecting all indices An implicit object list of all r-aligned edges:

```
((1,0) (0,0))
```

Object and subobject indices in the grid description start counting from 1, i.e. object no. 1 is the first object. The index 0 is special and denotes an undefined index. In this notation, it denotes all possible indices.

An implicit object list of the (z-aligned) boundary edges on the left boundary of the grid:

```
((0,1) (1,0))
```

The first entry of the index tuple denotes the first node in the r-space, the second entry denotes all edges in the z space. The implicit list denotes a total of 4 1d edges. Their implicit numbering is again given by iterating over all defining objects, lowest space first. The list therefore expands to

```
((0,1) (1,1))
((0,1) (1,2))
((0,1) (1,3))
((0,1) (1,4))
```

Selecting explicit lists of indices An implicit object list of the (z-aligned) right and left boundary edges:

```
((0,1) ([1,6],0))
```

The first entry of the index tuple denotes a list of nodes in the r-space, more specifically the first and the last (=6th) node. The second entry denotes again all edges in the z space. The implicit list then denotes a total of 8 1d edges in the following order:

```
((0,1) (1,1))
((0,1) (6,1))
((0,1) (1,2))
((0,1) (6,2))
((0,1) (1,3))
((0,1) (6,3))
((0,1) (1,4))
((0,1) (6,4))
```

Selecting ranges of indices An implicit object list of all 2d cells, except the cells on the left and right boundary.

```
((1,1) ((2,4),0))
```

The first entry of the index tuple denotes a range of edges in the r-space, more specifically the edges 2 to 4. The second entry of the index tuple denotes all four edges in the z-space. The implicit list then denotes a total of 12 2d cells in the following order:

```
((1,1) (2,1))
((1,1) (3,1))
((1,1) (4,1))
((1,1) (2,2))
((1,1) (3,2))
((1,1) (4,2))
((1,1) (2,3))
((1,1) (3,3))
((1,1) (4,3))
((1,1) (2,4))
((1,1) (3,4))
((1,1) (4,4))
```

All implementations of the grid service library define the constant GRID_UNDEFINED=0 to specify an undefined index. Use of GRID_UNDEFINED instead of 0 is advised to increase the readability of the code. The following notations are therefore equivalent ((1,0) (0,0)) = ((1,0) (GRID_UNDEFINED,GRID_UNDEFINED)) ((0,1) (1,0)) = ((0,1) (1,GRID_UNDEFINED))

4.1.3.1.3 Subgrid examples

A subgrid is an ordered list of grid objects of a common dimension. The difference to object lists is that they can contain objects of different object classes.

The subgrid concept is central to storing data on grids. To store data, first a subgrid has to be defined. The objects in the grid have a fixed order, which then allows to unambiguously store the data associated with the objects in vectors.

Technically, a subgrid is an ordered list of object lists, of which every individual list is either explicit or implicit. The ordering of the objects in the subgrid is then directly given by the ordering of the object lists and the ordering of the grid objects therein.

Subgrid example The following subgrid consists of all boundary edges of the 2d R,Z example grid #1, given as four implicit object lists.

```
((1,0) (0,1))    ! bottom edges
((0,1) (6,0))    ! right edges
((1,0) (0,5))    ! top edges
((0,1) (1,0))    ! left edges
```

Explicitly listing the objects in the order given by the subgrid gives:

```
1:  ((1,0) (1,1))    ! bottom edges
2:  ((1,0) (2,1))
3:  ((1,0) (3,1))
4:  ((1,0) (4,1))
5:  ((1,0) (5,1))
6:  ((0,1) (6,1))    ! right edges
7:  ((0,1) (6,2))
8:  ((0,1) (6,3))
9:  ((0,1) (6,4))
10: ((1,0) (1,5))    ! top edges
11: ((1,0) (2,5))
12: ((1,0) (3,5))
13: ((1,0) (4,5))
14: ((1,0) (5,5))
15: ((0,1) (1,1))    ! left edges
16: ((0,1) (1,2))
17: ((0,1) (1,3))
18: ((0,1) (1,4))
```

The number at the beginning of each line is the *local index* of the object, where local means locally in the subgrid. Note that, again, counting starts at 1.

4.1.3.2 Grid service library

4.1.3.2.1 Using the grid service library

4.1.3.2.1.1 Setting up the environment

The grid service library requires the EU-IM data structure version 4.09a (or later). Before using it you have to make sure your environment is set up properly. The following section assumes you are using csh or tcsh on the Gateway.

First, your environment variables have to be set up properly. To check them do

```
echo $TOKAMAKNAME
```

It should return

```
test
```

Also do

```
echo $DATAVERSION
```

It should return

```
4.09a
```

(or some higher version number). If either of them returns something different, run

```
source $EU-IMSCRIPTDIR/EU-IMv1 kepler test 4.09a > /dev/null
```

and check the variables again.

Second, you have to ensure your data tree is set up properly. Do

```
ls ~/public/itmdb/itm_trees/$TOKAMAKNAME/$DATAVERSION/mdsplus/0/
```

If you get something like "No such file or directory", you have to set up the tree first by running

```
$EU-IMSCRIPTDIR/create_user_itm_dir $TOKAMAKNAME $DATAVERSION
```

and then do the previous check again.

4.1.3.2.1.2 Checking out and testing the grid service library

To be able to get the code of the grid service library, you have to be a member of the EU-IM General Grid description (itmggd) project (you can apply for this here).

Once you are a member, you can check out the code by

```
svn co https://gforge6.eufus.eu/svn/itmggd itm-grid
```

Then you can run the unit tests for the grid service library by

```
cd itm-grid
source setup.csh
```

This will setup environment variables (especially OBJECTCODE) and aliases. Then do

```
testgrid setup
```

This will set up the build system for the individual languages. It will also build and execute a Fortran program that writes a simple 2d example grid stored in an edge CPO into shot 1, run 1.

To actually run the tests do

```
testgrid all
```

This will go through the implementations in the different languages (F90, Python, . . .) and run unit tests for every on of them. If all goes well, it should end with the message

```
Test all implementations: OK
```

If this is not the case, something is broken and must be fixed.

4.1.3.2.2 Example applications (outdated)

Note: this is a bit outdated. Have a look here.

4.1.3.2.2.1 Plotting 3d wall geometry with VisIt (temporary solution, not required any more)

This example plots a 3d wall representation stored in the edge CPO (in the future, this information will be stored in the wall CPO). The example data used here is generated by a preprocessing tool which is part of the ASCOT code.

1. Check out the grid service library (See above. You don't necessarily have to run the tests)
2. Change to the python/ directory and setup the environment:

```
cd itm-grid/python/; source setup.csh
```

3. Edit the file itm/examples/write_xdmf.py to use the right shot number
4. Run it (still in the python/ directory of the service library) with

```
   python26 itm/examples/write_xdmf.py

This will create two files: wall.xmf and wall.h5
```

5. Start visit with

```
   visit23

and open the wall.xmf file. Then select Plot->Mesh->Triangle and
click on the "Draw" button.
```

4.1.3.2.2.2 Using UALConnector to visualize CPOs using the general grid description

UALConnector allows you to bring data directly from the UAL into VisIt.

1. Check out the grid service library (See above. You don't necessarily have to run the tests)
2. Run UALConnector. Examples:

```
./itm-grid/ualconnector -s 9001,1,1.0 -c edge -u klingshi -t test -v 4.09a
```

```
./itm-grid/ualconnector -s 15,1,1.0 -c edge -u klingshi -t test -v 4.09a
```

3. When finished, close VisIt and terminate the UALConnector by typing 'quit'.

You don't even have to check out the service library. UALConnector is made available at

```
~klingshi/bin/itm-grid/ualconnector
```

, i.e.

```
~klingshi/bin/itm-grid/ualconnector -s 9001,1,1.0 -c edge -u klingshi -t test -v 4.09a
```

```
~klingshi/bin/itm-grid/ualconnector -s 15,1,1.0 -c edge -u klingshi -t test -v 4.09a
```

4.1.3.3 IMP3 General Grid Description and Grid Service Library - Tutorial

4.1.3.3.1 Setup your environment

```
echo $DATAVERSION
echo $TOKAMAKNAME
```

should give "4.09a" and "test". If not, run

```
source $EU-IMSCRIPTDIR/EU-IMv1 kepler test 4.09a > /dev/null
```

To copy the tutorial files:

```
cp -r ~klingshi/bin/itm-grid ~/public
```

Switch to the right version of the PGI compiler:

```
module unload openmpi/1.3.2/pgi-8.0 compilers/pgi/8.0
module load compilers/pgi/10.2 openmpi/1.4.3/pgi-10.2
```

To set up the environment:

```
cd $HOME/public/itm-grid/f90
source setup.csh
```

4.1.3.3.2 Compile & run examples

2d structured grid write example Source file is at:

```
src/examples/itm_grid_example1_2dstructured_servicelibrary.f90
```

Compile:

```
make depend
make $OBJECTCODE/itm_grid_example1_2dstructured_servicelibrary.exe
```

Run:

```
$OBJECTCODE/itm_grid_example1_2dstructured_servicelibrary.exe
```

2d structured grid read example Source file is at:

```
src/examples/itm_grid_example1_2dstructured_read.f90
```

Compile:

```
make $OBJECTCODE/itm_grid_example1_2dstructured_read.exe
```

Run:

```
$OBJECTCODE/itm_grid_example1_2dstructured_read.exe
```

4.1.3.3.3 Visualize

To visualize the data written by the example program

```
~klingshi/bin/itm-grid/ualconnector -s 9001,1,0.0 -c edge
```

To visualize a more complex dataset

```
~klingshi/bin/itm-grid/ualconnector -s 17151,899,1000.0 -c edge -u klingshi -t aug
```

Combining data from two CPOs:

```
~klingshi/bin/itm-grid/ualconnector -s 17151,898,1000.0 -c edge -s 17151,899,1000.0 -c edge -
↪u klingshi -t aug
```

CHAPTER

FIVE

EQUILIBRIUM AND MHD STABILITY WORKFLOW (EQSTABIL)

5.1 Workflow rationale

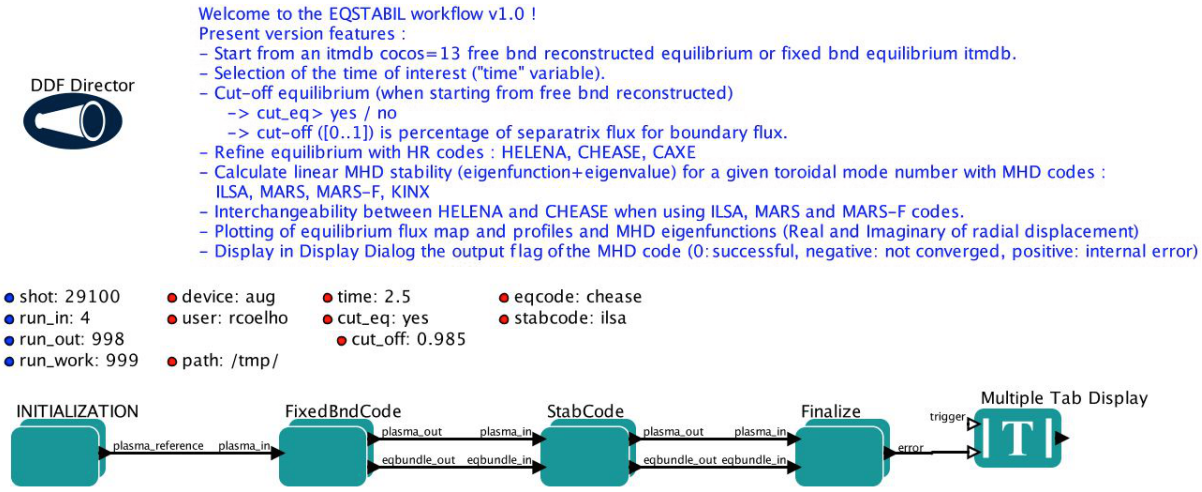
The EQSTABIL workflow is a Kepler workflow aimed at performing linear MHD stability analysis of tokamak plasma equilibria for a single or multiple toroidal mode numbers when executed. The high resolution equilibrium actors consider axisymmetric toroidal static plasmas with isotropic pressure and the linear MHD stability models stem from single fluid ideal/resistive MHD with compressibility.

The workflow is meant for straightforward stability calculations of any plasma scenario, reading from a pre-existent WPCD database shot/run/time entry. Therefore,

- It does not read from experimental databases storing locally processed plasma equilibrium e.g. PPFs (JET) or shotfiles (AUG). These can be fetched using exp2itm (ask the workflow developer for assistance).
- It is not meant for parametric studies in a single workflow execution e.g. process several time slices or scan over resistive wall position or number of poloidal harmonics. Dedicated runs for such cases are necessary, storing each run on a dedicated output shot/run_out database entry. The workflow may be subject to upgrades/revisions to accomodate new features that facilitate/enhance user experience so stay tuned for News and Recent activity.

5.2 Workflow organization & design

The top level layout of the workflow is shown below.



The workflow is organised in four sequential steps :

5.2.1 Initialization

Composite actor used to initialize the workflow. It reads from the ITM database that is specified by local variables (user, device, shot, run_in) and for the closest time sample to local variable *time* . If the user reads the input data from some other user database, the output data will however be written on his/her own database with shot/run_out id.

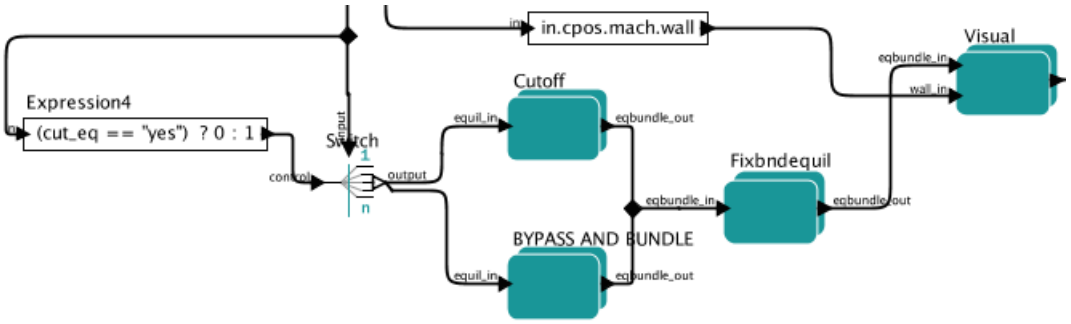
-> The workflow local variable *device* **must** be the same as the environment variable TOKAMAK-NAME. In case the two do not match the workflow stops execution. The user must close the workflow, ITMv1 with the correct device name and run the workflow.

-> Validity checks (void/not void) are made on the input equilibrium and coreprof CPOs (MARSGW actor can use coreprof for density profile). If the equilibrium CPO is not considered valid the workflow stops. If the coreprof CPO is not considered valid, the workflow continues to run but the user can still have the option to stop it before executing the chosen MHD code.

At the exit of the Composite actor, a Plasma_reference bundle (list of Kepler variables, mimicking the ETS bundle) is returned. This facilitates the future coupling of the workflow to the ETS.

5.2.2 FixedBndCode

Composite actor that prepares/calculates the equilibrium to be passed later to the MHD stability codes. This composite actor is composed of 3 main steps:



5.2.2.1 Redefining the plasma boundary (Cutoff)

This is deemed necessary when the input equilibrium (reconstructed/predictive equilibrium) as a separatrix as plasma boundary since at this moment none of the flux coordinates based equilibrium codes handles/returns plasmas with a separatrix.

If the input equilibrium does not contain a Psi(R,Z) equilibrium mapping the cut-off is not possible and thus the workflow execution will be stopped.

Redefining the plasma boundary is done by setting cut_eq: yes and places the new plasma boundary at a flux surface corresponding to cut_off (in percentage) of the input boundary flux.

The plasma profiles are also cut-off accordingly. An equilibrium bundle exits the actor containing occurrence=1 for the cut-off input equilibrium and occurrence=2 for the original equilibrium. If cut_off:no then both occurrences contain the original equilibrium.

A plot of the original + cut_off equilibrium summary is shown. Closing the plot window leads to the second stage.

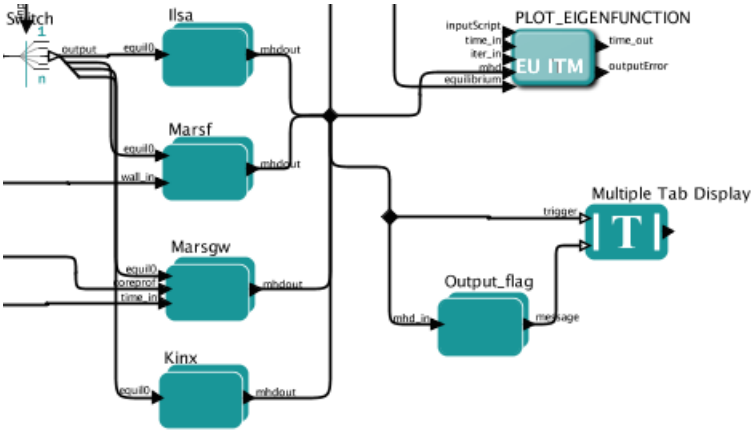
5.2.2.2 Calculation of Equilibrium (Fixbndequil)

Calculation of the high resolution equilibrium with 3 possible codes (CAXE, CHEASE, HELENA). The cut-off equilibrium (or original one) is passed to the equilibrium codes. The output HR equilibrium is added to the equilibrium bundle such that in the end one has occurrence=0 for the HR, occurrence=1 for the cut-off input equilibrium (or the original if not cut_off is requested) and occurrence=2 for the original equilibrium.

5.2.2.3 Visualization (Visual)

Visualization part. This part plots the (R,Z) flux map of the HR equilibrium and the most relevant profiles. The figures are saved automatically on closing the windows at the *path* indicated in the top level accordingly Kepler variable.

5.2.3 StabCode



Composite actor for the MHD stability calculation using 4 possible linear MHD stability codes (ILSA, KINX, MARS, MARS-F). After execution of the stability code is completed, plotting of the radial component of the displacement vector eigenfunction in the plasma domain is shown (real and imaginary parts). In case multiple toroidal mode numbers are set (ILSA or KINX), one plot window per each toroidal eigenmode is returned. A Copy in EPS format of each window is stored on the path defined by Kepler variable *path*

The Multiple Tab display window will also display the output flag of the code execution i.e. if the output is valid and the result can be used or not. The plasma bundle, on exit, is updated with the MHD cpo from the stability code.

5.2.4 Finalize

Composite actor to wrap up the final plasma bundle, with the equilibrium CPO containing 3 occurrences and one occurrence of the MHD CPO.

N.B. Only a single time slice of equilibrium and MHD CPOs is written, the remaining plasma bundle CPOs are written “as is” (whatever time slices).

5.3 Actors involved

Name	Location	Description
Check_Device	INITIALIZATION	Checks if the *device* Kepler variable coincides with the environment variable TOKAMAKNAME. If not the run stops.
SELECT_TIME_CORE/EQ	INITIALIZATION	Selects time slice of CPOs matching/closest to the requested time in *time* Kepler variable
Check Coreprof/Equil Time and Flag	INITIALIZATION	Checks the output_flag of the input CPOs to know if they are valid and prints the actual time stamp retrived from both CPOs (if time = -1 and output_flag is negative then the CPO is not valid). If the equilibrium is considered invalid a message in displayed on the Multi Tab Display window and workflow execution is stopped. If the coreprof is considered invalid a message is displayed on the Multi Tab Display window but the workflow will continue since some of the MHD codes handle plasma density internally as code parameter and their execution is not affected.
Cutoff	FixedBndCode	Performs the cut-off of the input equilibrium if requested and

5.4 Installing the workflow

The following links detail all required information to get a local installation of the workflow on the Gateway

4.10a3

4.10b8 (work in progress)

5.5 Setting up Workflow and Actor parameters

5.5.1 Setting workflow parameters

The workflow has basic settings in order to work.

- **shot** : the shot number on the user database (or from another user) where to read the reference equilibrium from (shot/run_in pair)
- **run_in** : the run number where the reference equilibrium is (shot/run_in pair)
- **run_work** : placeholder run for the temporary Kepler CPOs
- **run_out** : run number where the final results of the run will be stored (user running the workflow/shot/run_out). Since the input equilibrium can be a reconstruction that goes beyond the separatrix, 3 occurrences of the equilibrium are saved (original eq., cut equilibrium inside separatrix and corresponding high resolution equilibrium).
- **user** : username. Reading from someone else database is possible but the run_out will naturally be written to personal database only.
- **device** : device database where the input reference data is. MUST BE the same as env variable TOKAMAKNAME
- **time** : time slice (in equilibrium CPO) to be analysed in case the input shot/run_in contains many time slices.
- **path** : temporary folder where to dump the plots generated. Also used to store output files (used by HELENA/ILSA only)
- **cut_eq** :
  - yes : cut the input equilibrium (necessary if high resolution equilibrium code cannot handle separatrix plasma equilibria)
  - no : input equilibrium is used “as is”.
- **cut_off** : float]0,1], specifies the percentage of the separatrix flux that will define the poloidal flux of the new plasma boundary.
- **eqcode** : chease/caxe/helena. The equilibrium code to be used
- **stabcode** : ilsa/kinx/marsgw/marsf. The MHD stability code to be used

The user can always prevent the workflow from proceeding to the calculation of the high resolution equilibrium after the cut-off stage by Pressing the STOP button in Kepler GUI before closing the plot window with the summary of the equilibrium.

5.5.2 Setting actor parameters

Actor parameters are set on the actors themselves (not passed by the workflow). To access the actors codeparam the easiest route is to :

1. Click on “Outline” Tab (below the “Pause” button)
2. Type the name of the actor and press “Search” (or Enter)
3. On the final item in the chain of the actor composite, right click and press “Configure”. A pop-up panel appears
4. Click on “Edit Code Parameters” and a new window appears
5. Edit the code parameters and Press “Save & Exit”
6. Press “Commit” and setting is completed

5.6 Test cases and self-oriented training

Several test cases are available for testing, corresponding to different applications/examples. The itmdb files are found on the software release folder under */tutorial*

Case	Path	Original source	Description
1	/tutorial/case1	gvlad/test/180/300	Test equilibrium of elongated JET-like plasma, unstable to internal n=1 mode
2	/tutorial/case2	diy/test/1/2	Test equilibrium of circular plasma, unstable to global n=1 mode
3	/tutorial/case3	rcoelho/aug/29100/5	AUG equilibrium without separatrix, unstable to internal/global n=1 mode
4	/tutorial/case4	rcoelho/jet/77877/2	JET equilibrium without separatrix, unstable to internal n=1 mode
5	/tutorial/case5	rcoelho/aug/29100/4	Same equilibrium of Case 3 but from full (R,Z) CLISTE reconstruction.

Guided Tutorial on EQSTABIL

5.7 News and Recent activity

under construction

CHAPTER

SIX

TURBULENT FLUX QUANTITIES IN TRANSPORT MODELS

6.1 Overview

In conventional transport modelling, all quantities appearing in the equations are 1-D, in some radial coordinate (poloidal flux, normalised radius, etc). In general any monotonic radial coordinate is acceptable. In the TF-EU-IM, the toroidal flux radius is standard. All we need from the radial coordinate is the transformation to get to \(V, \) the volume enclosed by the flux surface, which is fundamental to the governing equations, which are conservation laws.

What we have to do is to take a measured result, which is a time-averaged fluctuation-based transport flux and turn it into 1-D quantities suitable to modelling. This is done using the flux surface average, explained in conventions. The transport equations themselves constitute a mean field approximation to the 3-D conservation laws. For the fundamentals encountered in transport modelling see R Hazeltine and J Meiss, Plasma Confinement (Addison-Wesley, 1992) chapter 8. For the special properties of transport driven by small-scale pressure driven ExB microturbulence see B Scott, "The character of transport caused by ExB drift turbulence," Phys Plasmas 10 (2003) 963-976.

For ambipolarity we follow the rules for dynamical alignment, which follows the physics of how electron fluctuations determine the ExB velocity fluctuations, which then advect all species. Magnetic flutter nonlinearities act independently of this, but in our modelling they are used solely for heat fluxes since the averaged particle transport due to magnetic flutter and the current cancels, leaving the parallel ion velocity which we neglect for this purpose. The reference for dynamical alignment is B Scott, "Dynamical alignment in three species tokamak edge turbulence," Phys Plasmas 12 (2005) 082305.

Note: there are now auxiliary actors provided for this purpose: IMP4DV, which does the D/V conversion and enforces ambipolarity assuming absence of impurities, and IMP4imp, which subsequently enforces ambipolarity for the set of main ion and impurity species. The IMP4DV actor should be invoked directly after the transport model actor in the workflow chain, if the model produces only fluxes or if the coefficients have to be modified with the flux given. Ambipolarity is done using IMP4imp if the coreimpurity CPO is used in the workflow. These auxiliary actors are described on the *auxiliary actors page.*

6.2 Particle Flux as an Example

The mean field equation governing particle balance is the transport equation for electrons,

$$\frac{\partial}{\partial t}\langle n\rangle + \langle \vec{\nabla}\cdot \tilde{n}\vec{\tilde{v}}_E\rangle = S$$

in which the tilde symbol over the n and v denotes fluctuating quantities and we neglect all transport processes except ExB eddy diffusion. The ExB velocity is given by

$$\vec{v}_E = \frac{c}{B^2}\vec{B}\times\vec{\nabla}\phi$$

where \(\phi \) is the electrostatic potential.

The angle brackets denote the flux surface average, and we will use the property that the flux surface average of a divergence of a vector is the volume derivative of the flux surface average of a contravariant volume component of the vector, in this case

$$\langle \vec{\nabla} \cdot \vec{\Gamma} \rangle = \frac{\partial}{\partial V} \langle \Gamma^V \rangle$$

where \(\Gamma \) is the particle flux whose flux-surface averaged volume component is

$$\langle \Gamma^V \rangle = \langle \widetilde{n} \widetilde{v}_E^V \rangle$$

This is converted to expression in terms of the radial coordinate \(\rho \) using the fact that both \(V \) and \(\rho \) are flux quantities whose gradients are parallel to each other. We have

$$\frac{\partial}{\partial V} = \frac{1}{V'_\rho} \frac{\partial}{\partial \rho} \qquad \Gamma^\rho = \frac{1}{V'_\rho} \Gamma^V \qquad V'_\rho = \frac{\partial V}{\partial \rho} \qquad g^{VV} = (V'_\rho)^2 g^{\rho\rho}$$

so we can write the transport equation as

$$\frac{\partial n}{\partial t} + \frac{1}{V'_\rho} \frac{\partial}{\partial \rho} V'_\rho \langle \Gamma^\rho \rangle = S,$$

where we have replaced \(\langle n \rangle \) with \(n \) following the assumptions of the 1-D version of mean field transport theory.

With all quantities now expressed in terms of flux quantities, we are free to characterise the transport flux \(\langle \Gamma^\rho \rangle \) in an arbitrary way, so long as only flux quantities appear. The flux expansion within the flux surface as well as expansion or contraction of surfaces of constant \(\rho \) is treated using the metric coefficient \(g^{ \rho \rho} \) which is dimensionless. This way we can characterise transport in terms of an effective diffusivity and an effective frictional slip velocity which are given in SI units. By convention both of these are done solely via \(g^{ \rho \rho} \) for convenience, also reflecting that the effective velocity is actually marking off-diagonal diffusive elements. Our convention for this follows the ETS code and is given by

$$\langle \Gamma^\rho \rangle = \langle g^{\rho\rho} \rangle \left(n V_{\text{eff}} - D_{\text{eff}} \frac{\partial n}{\partial \rho} \right)$$

So despite the special spatial distribution of any particular transport process (ie, the underlying instability or nonlinear free energy access), the flux-surface averaged flux itself and its expression in terms of diffusion and frictional slip are identical characterisations.

6.3 Metric Coefficients

Transport modellers want the Ds and Vs as physical quantities in SI units. In general the fluxes are (magnetic) flux surface averaged quantities, which implies the existence of metric elements in the conversion. In our case we need \(\langle g^{\rho \rho} \rangle \) where \(\rho \) is the toroidal flux radius in meters, so the metric elements are dimensionless. In the equilibrium CPO, this is gm3 under equilibrium%profiles_1d in the structure.

Note this is different from the ASTRA code which casts the Vs as proper velocities, i.e., with one factor of grad-rho given by \(\langle \sqrt{g^{\rho \rho}} \rangle \) which is gm7 under equilibrium%profiles_1d in the structure. The units are the same and the informational content is the same, but this difference has to be taken into account in any transport modelling and benchmarking.

6.4 Heat Fluxes

The heat flux is treated in a similar way, with transport equation

$$\frac{3}{2}\frac{\partial p_e}{\partial t} + \frac{1}{V'_\rho}\frac{\partial}{\partial \rho}V'_\rho\langle q_e^\rho\rangle = Q_e + \sum_{\text{ions}} T_{ei},$$

for electrons, with \(T_{ei} \) giving the species transfer and \(Q_e \) the source. For ExB transport the heat flux has a advective (also called convective) and a conductive piece given by

$$q_E = q_{E\text{cond}} + (3/2)T\Gamma_E$$

which appears with a 3/2 due to the Poynting cancellation. For magnetic flutter transport the advective piece appears with the usual factor,

$$q_m = q_{m\text{cond}} + (5/2)T\Gamma_m$$

Here the forms are given for each species and \(E \) and \(m \) refer to the ExB eddy and magnetic flutter channels, respectively. For reasons given below we are neglecting the magnetic flutter piece \(\Gamma_m \) for the time being, and then the flutter piece merely adds to the heat diffusivity.

The forms of these due to the fluctuations are then

$$\langle q^\rho\rangle = (3/2)\langle \widetilde{p}\widetilde{v}_E^\rho\rangle + \langle \widetilde{q}_\parallel \widetilde{b}^\rho\rangle$$

which breaks into advective and conductive pieces according to linearisation of the pressure fluctuations

$$\langle q^\rho_{\text{cond}}\rangle = (3/2)n\langle \widetilde{T}\widetilde{v}_E^\rho\rangle + \langle \widetilde{q}_\parallel \widetilde{b}^\rho\rangle \qquad \langle q^\rho_{\text{adv}}\rangle = (3/2)T\Gamma = (3/2)T\langle \widetilde{n}\widetilde{v}_E^\rho\rangle$$

hence the density fluctuation piece is accounted for by the particle flux. Neglect of the magnetic flutter advective piece (and particle flux) is the same as neglect of the \({\widetilde u_\parallel} {\widetilde b^ \rho} \) nonlinearity (in the delivery of the results, not in the turbulence computations themselves).

The total conductive flux is then represented by

$$\langle q^\rho_{\text{cond}}\rangle = \langle g^{\rho\rho}\rangle \left(nTY_{\text{eff}} - n\chi_{\text{eff}}\frac{\partial T}{\partial \rho}\right)$$

with \(\chi \) and \(Y \) giving the heat diffusion and frictional slip pieces for each species, respectively (these are in diff_eff and vconv_eff in the CPO for each quantity).

Operationally, the turbulence module communicates the diff_eff and vconv_eff due to each transport channel for each species to the transport solver, and the metric coefficients are used by both modules. The two modules can be on arbitrarily different grids, which communicate through standard interpolation. This despite the fact that transport at the micro-level is angle dependent (in general, it can be 3-D in the time average if the sources are 3-D). The effective transport is 1-D so long as parallel sound transit within the flux surface remains fast compared to the local transport time. This breaks down anyway in the edge, so the fact that the volume is a problematic coordinate and the flux surface average is a problematic operation on open field lines doesn't enter.

6.5 Ds and Vs from Turbulence Codes to Transport Solvers

To serve the results from turbulence codes to transport solvers, we have to turn the fluxes (results) into diffusivities and effective velocities (coefficients, Ds and Vs for short), which represent more information than is at hand. Transport solvers must work with Ds and Vs because they use implicit schemes.

The matrix must be diagonally dominant; hence one cannot simply use the Vs. Fluxes which are zero and/or negative should be given with positive diffusivities for the solvers to work. We need a set of rules to provide this.

Considering the particle and heat transport fluxes for a given species, we convert the gradient in to a logarithmic derivative and express the flux in terms of a specific flux, which has units of velocity,

$$F = \frac{1}{n}\langle g^{\rho\rho}\rangle^{-1}\langle\Gamma^{\rho}\rangle = V_{\text{eff}} - D_{\text{eff}}\frac{\partial \log n}{\partial \rho}$$
$$G = \frac{1}{nT}\langle g^{\rho\rho}\rangle^{-1}\langle q^{\rho}_{\text{cond}}\rangle = Y_{\text{eff}} - \chi_{\text{eff}}\frac{\partial \log T}{\partial \rho}$$

wherein the conductive part of the heat flux (without the \(3 \Gamma / 2 \) enters.

The choice of what to do with the Ds and Vs is somewhat arbitrary. The needs of implicit transport solvers is for a positive D regardless of the value or sign of either flux. We decide this by putting a limit on the effective Prandtl number or its inverse: the larger specific flux is taken to be entirely diffusive, with the effective velocity set to zero. Furthermore, to address cases with very small or negative gradients, we use proxy variables for the scale lengths to calculate the provisional diffusivities before using the Prandtl number limitation to turn these into actual diffusivities. Finally, the rest of the flux is asigned to the effective velocity, so that the D and V formula reflects the actual specific flux.

The Prandtl number limitation is expressed as follows. If the smaller specific flux is within a factor of 5 of the larger, then both are purely diffusive and the effective velocities are both zero. If not, then the D ratio is set to 5, with the result that the smaller D, having been corrected, is accompanied by the corresponding V, which is now nonzero. The specific flux with the larger D will be returned with a V which is zero.

The rationale is that the turbulent mixing by the ExB velocity affects all processes, but that linear forcing can shift the average phase shift of the fluctuations such that the effective flux can be small or negative. The simplest example is adiabatic electrons, for which the ion heat flux is robust but the particle flux is zero. In most situations the specific heat flux will be the larger, and hence the familiar situation is that of a D and V for the particle flux but a D (the chi) only for the conductive heat flux.

The full algorithm starting with the specific fluxes appears as

$$L_n^{-1} = \max\left(\frac{1}{R}, \left|\frac{\partial \log n}{\partial \rho}\right|\right) \qquad L_T^{-1} = \max\left(\frac{1}{R}, \left|\frac{\partial \log T}{\partial \rho}\right|\right)$$

$$D' = |F|\, L_n \qquad \chi' = |G|\, L_T$$

$$D = \max\left(D', \frac{1}{5}\chi'\right) \qquad \chi = \max\left(\chi', \frac{1}{5}D'\right)$$

$$V = \left(F + D\frac{\partial \log n}{\partial \rho}\right) \qquad Y = \left(G + \chi\frac{\partial \log T}{\partial \rho}\right)$$

and all four elements are set. Note that the channels are done in parallel except for the Prandtl correction, in which the Max's are taken sequentially. For the provisional diffusivities, absolute values are used to ensure positive values which are needed by transport solvers.

Note how in the end the actual gradients are used. If the gradients are moderate then their actual values are used, and if the Prandtl correction is not invoked, then both channels are diagonal. In any case the full relation is used to get the effective velocities (V and Y) so having set the rules to handle the arbitrariness

of the diffusivities (D and chi) to guarantee reasonable diagonal dominance in a transport solver, the D's and V's agree with the fluxes themselves.

If there are more than two specific fluxes per species to consider, then we treat each scale length separately as above and use N-way maxima in the Prandtl correction for the N channels.

6.6 Ambipolarity

There remains the issue of ambipolarity of the D and V for particle flux. For a pure singly charged plasma the ion and electron Ds and Vs should be equal. Even if the turbulence model is gyrokinetic or gyrofluid, in which case the gyrocenter charge density is not zero but is equal to the generalised vorticity (polarisation), the quantities given to a transport solver should follow the rules for a fluid representation. However, transport modelling usually applies ambipolarity rules to the electrons after computing the ions, while the action of turbulence is actually the other way around: Dynamical alignment refers to the process by which (1) electron parallel dynamics controls the electrostatic fluctuations, then (2) the resulting ExB velocity advects all species equally. So we correct the particle fluxes by assuming the electrons determine the D according to the above procedure and then (1) the fluctuations in the flux-inducing part of the spectrum for the logarithmic densities are the same, and (2) the D's are the same. Then the V's are solved for again, by taking

$$D_z = D_e = D \qquad V_z = V_e + D\frac{\partial \log b_z}{\partial \rho} \qquad b_z = n_z/n_e$$

This is better than the transport modelling convention but will give them the same information in a different way, and they will compute ambipolar particle fluxes (radial transport of charge is zero).

6.7 Statistical Character

Turbulence has a statistical character, so convergence to a mean is not monotonic and when within one std dev of the mean there is no further convergence. The diffusivity for ExB turbulence is comparable to

$$D_E = \langle(\widetilde{v}_E)^2\rangle / \langle(\varpi)^2\rangle^{1/2} \qquad \varpi_E = \frac{c}{B}\nabla_\perp^2\widetilde{\phi}$$

where \(\varpi_E \) is the ExB vorticity fluctuation, and these angle brackets denote the ensemble average. To get an ensemble average over a statistical quantity in practice, one must do some sort of finite-time running averaging.

For transport modelling, the transport coefficients derived from a turbulence code should always be given in terms of *running exponential averages.*

CHAPTER

SEVEN

RUNNING EXPONENTIAL AVERAGE

7.1 Overview

In conventional transport modelling, turbulent fluxes are modelled in terms of processes which are diffusive in the local relaxation sense, with the average flux given by a diffusion coefficient and an effective pinch velocity. The equations are of dominantly parabolic character, which means in practice that an iterate will move monotonically towards the solution in parameter space.

This is not the case for turbulence. Convergence is statistical, which is something different than a diffusive relaxation. If turbulence is stationary, it is meant only that the mean of a distribution of iterates is stationary, not the iterates themselves. The standard deviation can be significant, of order unity compared to the mean, of any distribution of iterates.

This makes for a noisy signal if the output of a turbulence code is used for transport coefficients in a workflow. A sound way to overcome the attendant problems is to use a moving average. Even an average over a moving window can be as noisy as the original signal, however. What works better is a weighted average over recent past values. A method to get this is called a running exponential average, which is essentially the same thing as a convolution integral over an exponential memory decay times the past signal. It turns out to be very easy to obtain this without saving past values.

The original reference for the following is S W Roberts, "Control Chart Tests Based on Geometric Moving Averages," Technometrics 1 (1959) 239-250, cited by all the good WWW resources, including the Wikipedia page on Moving Averages and the NIST Statistical Handbook online.

7.2 Definition

Consider a process \(p (\vec u) \) which is a functional of dependent variables \(\vec u \). Measure \(p \) at discrete time intervals \(t_n, \) with values \(p_n=p(t_n) \) and interval length \(\tau=t_n-t_{n-1} \). The moving exponential average \(A_n=A(p_n) \) on the \(n \hbox{-th} \) interval is defined as

$$A_n = \epsilon p_n + (1-\epsilon) A_{n-1} \qquad \text{with} \qquad \epsilon = \alpha\tau$$

in which the small parameter \(\epsilon \) is given in terms of the interval \(\tau \) and an inverse time constant \(\alpha. \)

In the first instance \(p \) is measured there is no \(A \) so the first value of \(A \) is simply set to \(p \) since it can be assumed that the initial state for \(p \) has persisted for infinite previous time up to the initial time point.

7.3 Differential Equation

The equivalent differential equation is found by forming the relevant finite difference,

$$A_n - A_{n-1} = \epsilon(p_n - A_{n-1})$$

which we can also cast as

$$(1-\epsilon)(A_n - A_{n-1}) = \epsilon(p_n - A_n)$$

Taking the limit \(\tau \to 0 \) is the same as taking \(\epsilon \to 0 \) so both of these expressions become equivalent to

$$\frac{\partial A}{\partial t} = \alpha(p - A)$$

whose solution is given below.

7.4 Equivalence to Past-Time Convolution Integral

The solution of the above differential equation is given by the method of undetermined coefficients,

$$\frac{\partial A}{\partial t} + \alpha A = \alpha p$$

$$e^{-\alpha t}\frac{\partial}{\partial t}\left(e^{\alpha t}A\right) = \alpha p$$

$$\frac{\partial}{\partial t}\left(e^{\alpha t}A\right) = \alpha p e^{\alpha t}$$

We may integrate this over all past time, to find

$$A(t) = \int_{-\infty}^{t} \alpha dt' p(t') e^{-\alpha(t-t')}$$

This is a convolution integral over the kernel \(e^{-\alpha(t-t')} \) and the signal \(p(t') \). The time constant \(\alpha^{-1} \) is just the memory decay time, while if \(p \) is constant then the integral yields unity times \(p \). This is the same as the normalisation with the \((1-\epsilon) \) factor in the average formula above, which is needed since the interval is of finite size.

Hence the running exponential average is operationally the same as a memory decay integral over past time. The elegant feature is the need to keep only the current value of \(A \), as it already contains all that is needed of the past time evolution of \(p \).

7.5 notes

Some properties of the running exponential average and how to choose its main time-memory parameter:

- The \((1-\epsilon) \) factor is needed for normalisation
- if \(p=\hbox{constant} \) then \(A=p \) for all \(t \)

- the integral with \(\alpha dt' \) yields unity
- the \(\epsilon \) and \((1-\epsilon) \) factors add to unity
- therefore set the first value of \(A \) to the first value of \(p \)
- in choosing the memory decay time \(\alpha^{-1} \dots \)
- one should have \(\alpha \tau_{cor} \ll 1 \)
- best results are for \(\alpha \tau_{sat} \sim 1 \)
- some trial/error required; edge turbulence likes \(\alpha^{-1}=200 L_ \parallel / c_s \)

In these expressions \(\tau_{{cor}} \) and \(\tau_{{sat}} \) are the correlation and saturation times of the turbulence, respectively.

CHAPTER

EIGHT

CODES

8.1 IMASviz

The IMASViz code is used for IMAS visualisation.

8.2 FC2K

FC2K is a tool for wrapping a Fortran or C++ source code into a Kepler actor. Before using it, your physics code should be EU-IM-compliant (i.e. use CPOs as input/output).

After running the EU-IMv1 script (to properly set up the environment variables), FC2K can be run simply by typing fc2k in the Linux command line.

`fc2k` was developed by ISIP in Java/Pytho. The program source is stored in the Gforge subversion repository fc2k. To check out a subversion working copy of the repository, storing it in subdirectory *fc2k*, do

```
svn co https://gforge6.eufus.eu/svn/fc2k
```

Executables etc are in $SWEU-IMDIR/ihm/fc2k/.

There are a few tools for managing actors, in $EU-IMSCRIPTDIR/ (put it in your $PATH):

- rmactor : remove an actor from your Kepler version ($KEPLER).
- extract_actor : export an actor from $KEPLER.
- import_actor

```
cd $KEPLER
ant buildkarlib
```

8.2.1 How to turn a C++ code into a Kepler actor

This document is based on material provided by Yann Frauel and describes how to make your C++ code EU-IM compliant and how to turn it into a Kepler actor.

8.2.1.1 Adapt your C++ function

You must include the header file UALClasses.h:

```
#include "UALClasses.h"
```

The function arguments that are arrays or strings must be declared as pointers, as usual. All other arguments must be passed by reference (i.e. they must be declared with an ampersand):

```
void mycppfunction(double * vector, char * string, int & scalar)
```

The function arguments that are CPOs must be declared with types ItmNs::Itm::cpo_type or ItmNs::Itm::cpo_typeArray. The first form is for time-independent CPOs or a single slice of a time-dependent CPO. The latter is for a complete time-dependent CPO. Note that in all cases, the CPO is considered as a single object, not an array, so it must be passed by reference as mentioned above:

```
void mycppfunction(
ItmNs::Itm::limiter & lim,
ItmNs::Itm::coreimpur & cor,
ItmNs::Itm::ironmodelArray & iron)
```

The syntax is identical for input and output arguments. For output CPOs, do not forget to use the usual methods to assign strings and allocate arrays:

```
lim.datainfo.dataprovider.assign("test_limiter");
iron.array.resize(3);
iron.array(j).desc_iron.geom_iron.npoints.resize(3);
```

Otherwise, the content of CPOs is accessed as usual:

```
cout << lim.datainfo.dataprovider << endl;
cout << iron.array(j).desc_iron.geom_iron.npoints(i);
```

8.2.1.2 How to use code parameters

The code parameters are passed as the last argument with ItmNs::codeparam_t& type:

```
void mycppfunction(..., ItmNs::codeparam_t & codeparam)
```

Each field of the param structure is a vector of 132-byte strings, not necessarily terminated by 0-character! (This does not follow C/C++ standards and should be changed in the future.)

8.2.1.3 Compile your function as a library

You need to include the header directories for the UAL and Blitz:

```
-I$(UAL)/include -I$(UAL)/lowlevel -I$(UAL)/cppinterface/ -I/afs/efda-
itm.eu/gf/project/switm/blitz/blitz-0.9/include/
```

Same for linking:

```
-L$(UAL)/lib -lUALCPPInterface -lUALLowLevel -L/afs/efda-
itm.eu/gf/project/switm/blitz/blitz-0.9/lib -lblitz
```

Additionally, you must compile with the -fPIC option.

8.2.1.4 Full example

We want to generate an actor that has three different types of actors as inputs and three different types of actors as output. Additionally, we have an integer as input/output, a vector of doubles as output and a string as output. We also want to use code parameters. Content of mycppfunction.cpp:

```
#include "UALClasses.h"

typedef struct {
      char **parameters;
      char **default_param;
      char **schema;
} param;

void mycppfunction(
      ItmNs::Itm::summary  SUM,
      EU-IMNS::EU-IM::ANTENNAS & ANT,
      EU-IMNS::EU-IM::EQUILIBRIUMARRAY & EQ,
      INT & X,
      EU-IMNS::EU-IM::LIMITER & LIM,
      EU-IMNS::EU-IM::COREIMPUR & COR,
      EU-IMNS::EU-IM::IRONMODELARRAY & IRON,
      DOUBLE * Y,
      CHAR * STR,
PARAM & CODEPARAM)
{

      /* DISPLAY FIRST LINE OF PARAMETERS */
      COUT &LT< codeparam.parameters[0] << endl;
      cout << codeparam.default_param[0] << endl;
      cout << codeparam.schema[0] << endl;
      /* display content of inputs */
      cout << "x=" << x << endl;
      cout << sum.time << endl;
      cout << sum.datainfo.dataprovider << endl;
      cout << ant.datainfo.dataprovider << endl;
      cout << eq.array(0).datainfo.dataprovider << endl;
      for (int k=0; k<3; k++) {
           for (int i=0; i<4; i++) {
                cout << eq.array(k).profiles_1d.psi(i)<< " ";
           }
           cout << endl;
      }
      /* fill limiter CPO */
      lim.datainfo.dataprovider.assign("test_limiter");
      lim.position.r.resize(5);     // allocate vector
      for (int i=0; i<5; i++) {
            lim.position.r(i)=(i+1);
      }
      /* fill coreimpur CPO */
      cor.datainfo.dataprovider.assign("test_coreimpur");
      cor.flag.resize(3);          // allocate vector
      for (int i=0; i<3; i++) {
            cor.flag(i)=(i+1)*10;
      }
      cor.time=0; // don't forget to fill time for time-dependent CPOs
      /* fill ironmodel CPO */
      iron.array.resize(3);        // allocate slices
      for (int j=0; j<3; j++) {
            char s[255];
            sprintf(s,"test_ironmodel%d",j);
            iron.array(j).datainfo.dataprovider.assign(s); // allocate vector
            iron.array(j).desc_iron.geom_iron.npoints.resize(3);
            for (int i=0; i<3; i++) {
                  iron.array(j).desc_iron.geom_iron.npoints(i)=j*i;
            }
            iron.array(j).time=j;       // fill time for time-dependent CPOs
      }
      /* assign value to non CPO outputs */
      x=5;
```

(continues on next page)

(continued from previous page)

```
      for (int i=0; i<10; i++) {
            y[i]=i;
      }
      strcpy(str,"This is a test string");
}
```

Content of Makefile:

```
CXXFLAGS=-g -fPIC -I$(UAL)/include -I$(UAL)/lowlevel -I$(UAL)/cppinterface/
-I$SWEU-IMDIR/blitz/blitz-0.9/include/
LDFLAGS=-L$(UAL)/lib -lUALCPPInterface -lUALLowLevel -L/afs/efda-
itm.eu/gf/project/switm/blitz/blitz-0.9/lib -lblitz
libmycppfunction.a: mycppfunction.o
      ar -rvs libmycppfunction.a mycppfunction.o
mycppfunction.o: mycppfunction.cpp
clean:
      rm mycppfunction.o libmycppfunction.a
```

8.2.1.5 How to fill the FC2K window

First tab (Argument):

- set number of input and output arguments (combined)
- select type of arguments from drop-down menu
- tick if argument is a single time slice
- tick if argument is array (not for pointers)
- if necessary define size of arrays
- tick if argument is input argument
- tick if argument is output argument (multiple ticks possible)

The fields Kepler, Ptolemy, and UAL are automatically filled with the values which you set by running the `EU-IMv1 script`.

Second tab (HasReturn):

- specify return parameters (type, array, size)

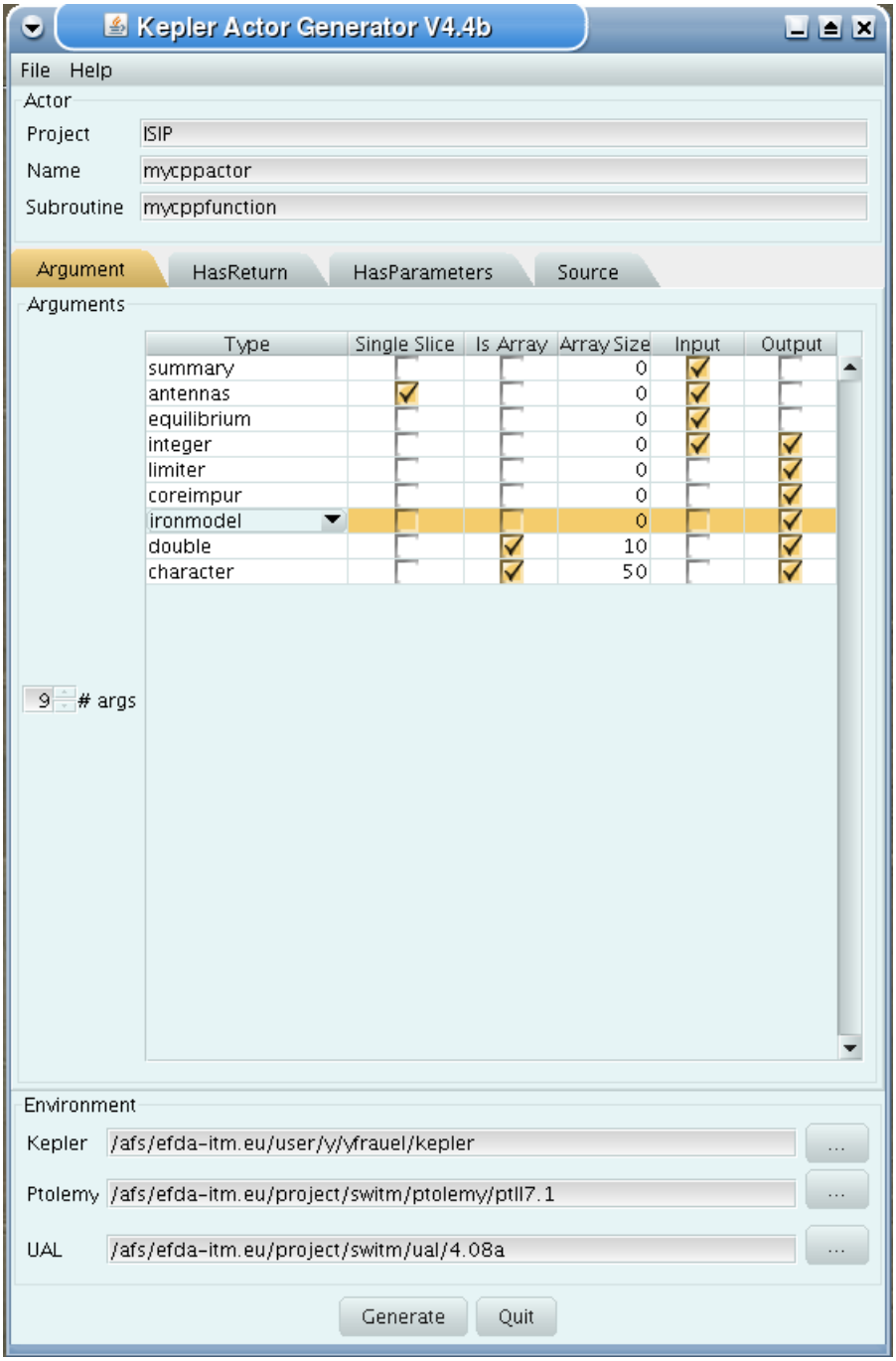
Third tab (HasParameters):

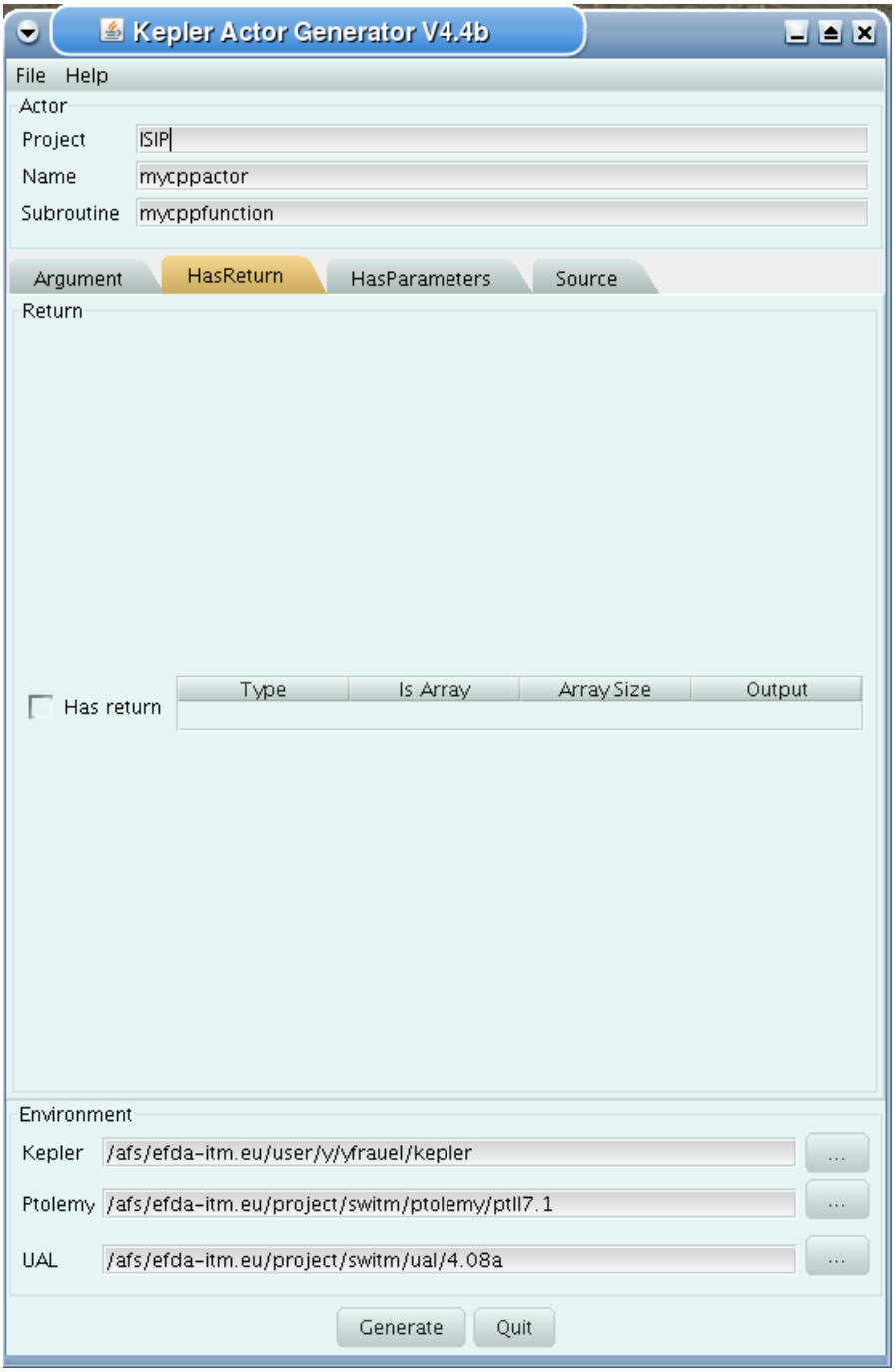
- tick if subroutine uses code specific parameters
- specify (or browse for) XML code parameter input file
- specify (or browse for) XML default code parameter file
- specify (or browse for) W3C XML schema file (XSD)

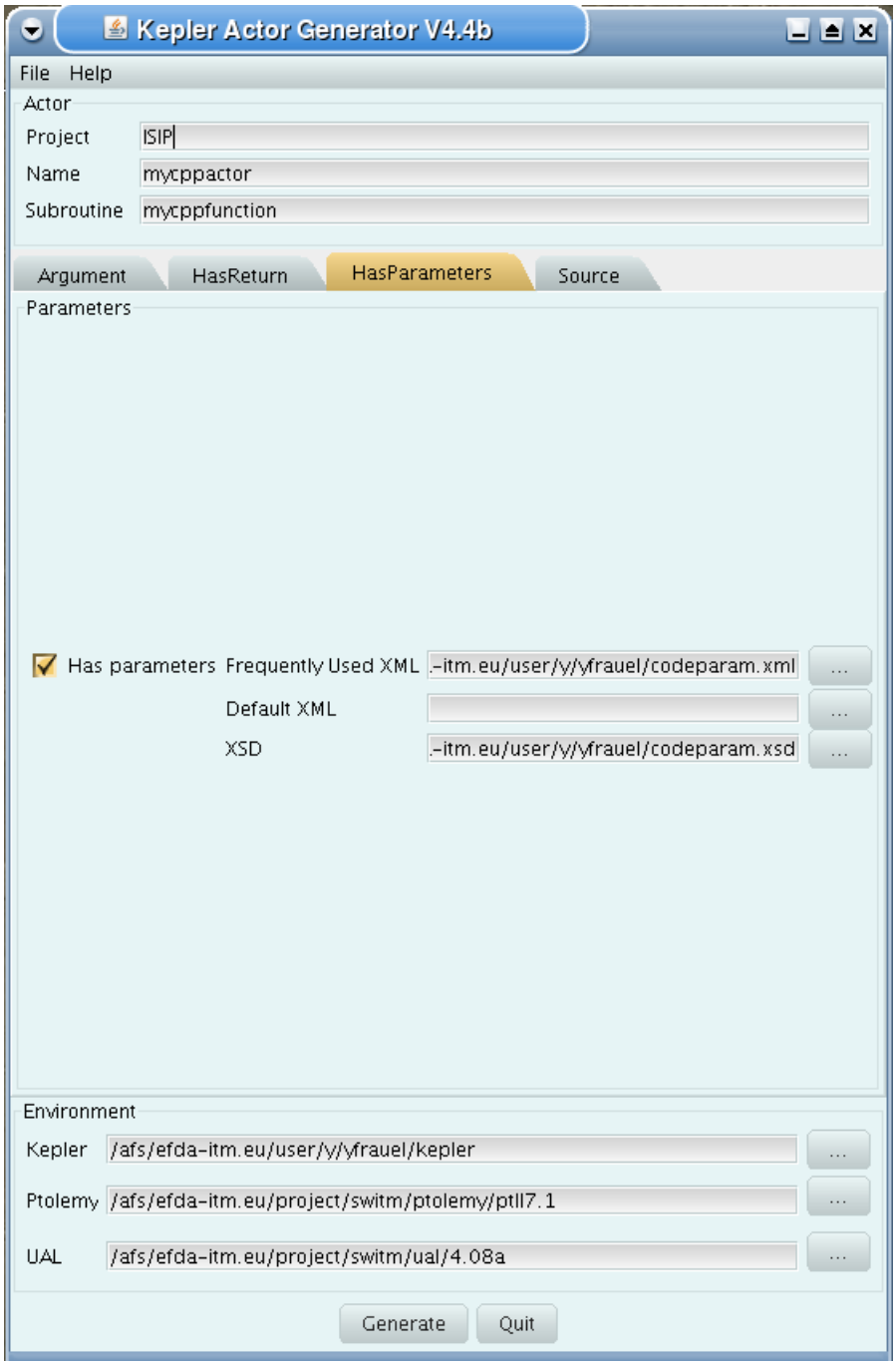
For information on code specific parameters, please see *How to handle code specific parameters*.

Fourth tab (Source):

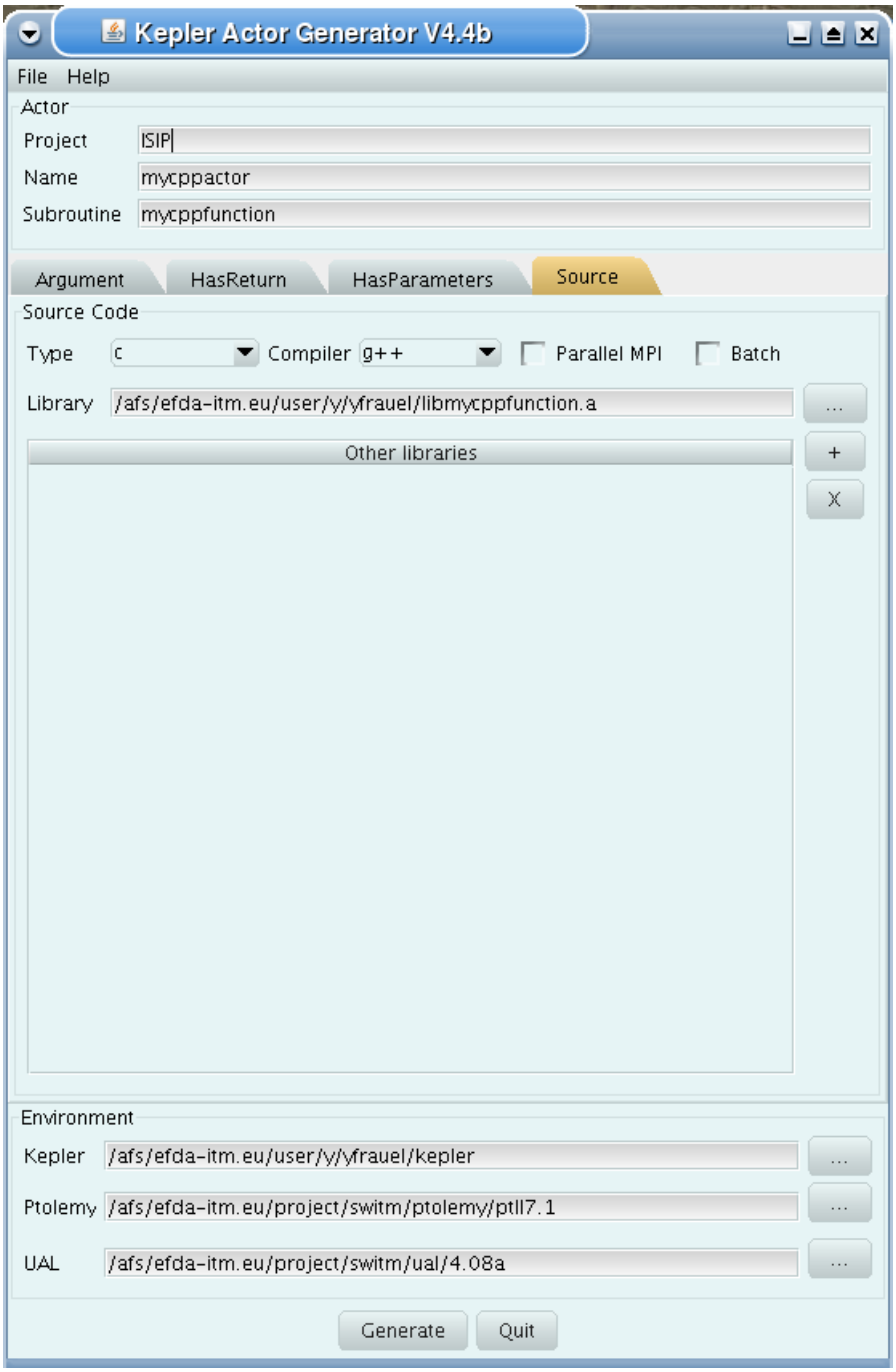
- specify programming language of source code
- select appropriate compiler
- tick Parallel MPI if code module is using MPI


Kepler Actor Generator V4.4b
File Help
Actor
Project ISIP
Name mycppactor
Subroutine mycppfunction
Argument HasReturn HasParameters Source
Arguments
Type Single Slice Is Array Array Size Input Output
summary 0
antennas 0
equilibrium 0
integer 0
limiter 0
coreimpur 0
ironmodel 0
double 10
character 50
9 # args
Environment
Kepler /afs/efda-itm.eu/user/y/yfrauel/kepler
Ptolemy /afs/efda-itm.eu/project/switm/ptolemy/ptII7.1
UAL /afs/efda-itm.eu/project/switm/ual/4.08a
Generate Quit


Kepler Actor Generator V4.4b
File
Help
Actor
Project
ISIP
Name
mycppactor
Subroutine
mycppfunction
Argument
HasReturn
HasParameters
Source
Return
Has return
Type
Is Array
Array Size
Output
Environment
Kepler
/afs/efda-itm.eu/user/y/yfrauel/kepler
Ptolemy
/afs/efda-itm.eu/project/switm/ptolemy/ptII7.1
UAL
/afs/efda-itm.eu/project/switm/ual/4.08a
Generate
Quit


Kepler Actor Generator V4.4b
File Help
Actor
Project ISIP
Name mycppactor
Subroutine mycppfunction
Argument HasReturn HasParameters Source
Parameters
Has parameters Frequently Used XML .-itm.eu/user/y/yfrauel/codeparam.xml
Default XML
XSD .-itm.eu/user/y/yfrauel/codeparam.xsd
Environment
Kepler /afs/efda-itm.eu/user/y/yfrauel/kepler
Ptolemy /afs/efda-itm.eu/project/switm/ptolemy/ptII7.1
UAL /afs/efda-itm.eu/project/switm/ual/4.08a
Generate Quit

- tick Batch if code module shall be run in batch mode rather than interactively when running Kepler workflows
- specify (or browse for) library file containing the code module
- specify (or browse for) other libraries required by the code module



8.3 Plasma equilibrium and MHD (IMP12) list of codes

The following list lists the codes and modules which are part of EU-IM-TF tasks and their responsible officers. A link takes you to the status page for each code.

A number of IMP12 codes have projects on gforge.

Update the code status here.

8.3.1 Free boundary equilibrium codes

CEDRES++, S. Brémond, CEA (code status, gforge)

CLISTE, P. Mc Carthy, DCU (code status)

CREATE-NL, M. Mattei, ENEA Frascati (code status)

EFIT++, L. Appel, CCFE (code status)

EQUAL, W. Zwingmann, EC (code status, gforge, *actor*)

EQUINOX, B. Faugeras, CEA (code status, gforge)

FIXFREE, E. Giovannozzi, ENEA Frascati (code status)

8.3.2 Fixed boundary equilibrium codes

CAXE, S. Medvedev, EPFL (code status)

CHEASE, O. Sauter, EPFL (code status, gforge)

HELENA, C. Konz, IPP (code status, *actor*)

8.3.3 Linear MHD stability codes

KINX, S. Medvedev, EPFL (code status)

ILSA, C. Konz, IPP (code status, *actor*)

MARS, G. Vlad, ENEA Frascati (code status, gforge)

MARS-F, D. Yadykin, Chalmers (code status, gforge)

8.3.4 Equilibrium codes with flow

FLOW, R. Paccagnella, ENEA RFX (code status)

8.3.5 3D Equilibrium Codes

8.3.6 Sawtooth Crash Modules

SAWTEETH, O. Sauter, CRPP (code status, gforge)

8.3.7 ELM Modules

8.3.8 RWM Modules

8.3.9 NTM Modules

8.3.10 3D MHD Codes

JOREK, G. Huysmans, CEA (code status)

8.3.11 Error Field Modules

8.3.12 2D MHD Codes

8.3.13 Disruption Modules

8.3.14 Numerical Tools

PROGEN, C. Konz, IPP (code status, *actor*)

JALPHA, C. Konz, IPP (code status, *actor*)

8.4 Heating, current drive (H&CD) and fast particles (IMP5) list of codes

The following list lists the codes and modules which are part of EU-IM-TF tasks and their responsible officers.

A number of IMP5 codes have projects on gforge.

Update the code status here.

8.4.1 Electron heating codes

8.4.1.1 EC wave codes

- TORAY-FOM, E. Westerhof, FOM (code status, codeparam)
- TORBEAM, E. Poli, IPP-Garching (code status)
- GRAY, L. Figini, ENEA-CNR (code status, gforge, codeparam)
- TRAVIS, N. B. Marushchenko, IPP-Greifswald (code status, gforge)

8.4.1.2 LH wave codes

- RAYLH, A. Cardinali, EURATOM-ENEA (code status)

8.4.1.3 Combined EC and LH wave codes

- C3PO, Y. Peysson, CEA (Cadarache) (code status)

8.4.1.4 Combined electron Fokker-Planck codes

- RELAX, E. Westerhof, FOM (code status)
- LUKE, Y. Peysson (code status, gforge)

8.4.1.5 LH coupling

- ALOHA, J. Hillairet, CEA (Cadarache) (code status, gforge)

8.4.1.6 Time domain wave codes

- FWTOR, C. Tsironis, Hellenic Association (code status, gforge)

8.4.2 Ion heating codes

8.4.2.1 Wave codes for ion cyclotron heating

- TORIC, R. Bilato, IPP-Garching (code status, gforge)
- EVE, R. Dumont, CEA (Cadarache) (code status, gforge)
- LION, O. Sauter, CRPP
- Cyrano, E. Lerche, ERM/KMS
- ICCOUP, T. Johnson, VR (gforge)

8.4.2.2 Fokker-Planck codes for ion cyclotron heating

- FPSIM, L.-G. Eriksson, EC (code status, gforge)
- SSFPQL, R. Bilato, IPP-Garching (code status)
- RFOF, T. Johnson, VR (gforge, documentation, codeparam)
- Stix_Redist, E. Lerche and D. Van Eester (gforge, codeparam)
- Stix_Disp, E. Lerche and D. Van Eester (gforge)

8.4.2.3 NBI sources for Fokker-Planck codes

- BBNBI (Beamlet-based NBI module of ASCOT), O. Asunta, TEKES (code status, gforge)
- NEMO, M. Schneider, CEA (Cadarache) (code status, gforge,
- SNBI (OAW NBI source), K. Schöpf, OAW (code status)

8.4.2.4 Nuclear sources (input for Fokker-Planck codes)

- Nuclearsim, T.Johnson, VR (gforge, codeparam)

8.4.2.5 NBI Fokker-Planck codes

- RISK, M. Schneider, CEA (Cadarache) (code status, gforge)
- NBISIM, T. Johnson, VR
- FIDIT, K. Schöpf, OAW (code status)

8.4.2.6 Advanced codes

(The following codes include either the synergy between IC and NBI heating, or include both wave field and Fokker-Planck solver)

- ASCOT, S. Sipila, TEKES (code status, gforge, codeparam)
- SPOT, M. Schneider, CEA (Cadarache) (code status, gforge)
- SELFO-light, T. Hellsten, VR (code status, gforge)

8.4.2.7 Orbit tracing codes

- SOFI, S. Sipila, TEKES (code status, gforge)
- OAW Orbit Following Monte Carlo, K. Schöpf, OAW (code status)

8.4.3 Fast particle codes

8.4.3.1 Codes for fast ion-MHD interactions

- LIGKA, P. Lauber, IPP-Garching (code status)
- MARS, G. Vlad, ENEA-Frascati (code status, gforge)
- HYMAGYC, G. Vlad, ENEA-Frascati (code status)
- HMGC, C. Di Troia, ENEA-Frascati (code status)
- LEMAN, W.A. Cooper, EPFL-CRPP (code status)

8.4.3.2 Runaway electrons

- ARENA, G. Pokol and G. Csepany (code status, gforge)

8.5 Transport list of codes (IMP3)

- ASPOEL, , (code status)
- BIT1, , (code status)
- CARRE, , (code status)
- COS, , (code status)
- EIRENE, , (code status)
- EIRENE2, , (code status)
- EMC3-EIRENE, , (code status)
- ERO, , (code status)
- ETS, , (code status)
- METIS4EU-IM, , (code status)
- SOLPS, , (code status)
- SOLPS6, , (code status)

CHAPTER

NINE

CONVENTIONS

9.1 Standard Machine Names

The following machine names are suggested:

- aug
- ftu
- iter
- jet
- mast
- tcv
- tore_supra
- west

9.2 Physics Conventions

The EU-IM-TF has agreed on a variety of conventions to facilitate the integration of the code modules across EFDA. In the following the most important conventions are explained in detail to remove confusion and avoid ambiguity. For more physical detail than that represented here see F Hinton and R Hazeltine, Rev Mod Phys 48 (1976) 239-308, or R Hazeltine and J Meiss, Plasma Confinement (Addison-Wesley, 1992).

9.2.1 Coordinate System

There are generally two choices for defining a right-handed coordinate system in a toroidal geometry with the following coordinates:

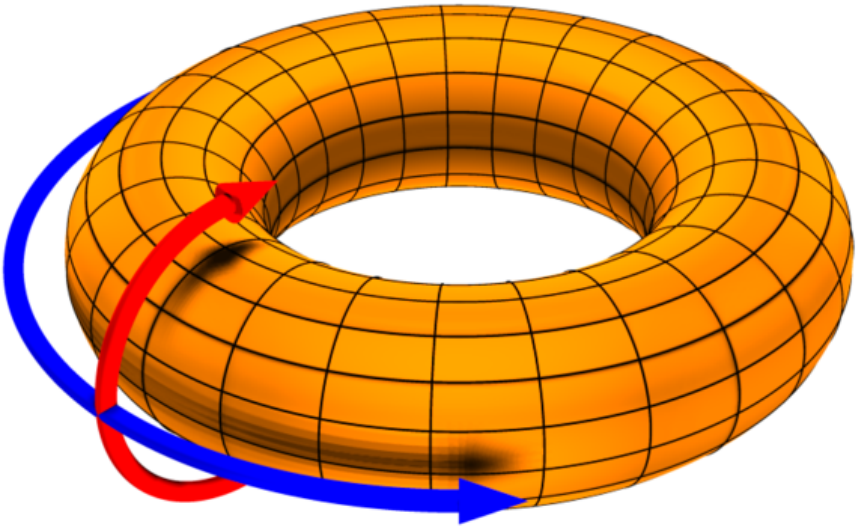
- major radius R
- vertical heights Z
- toroidal angle \(\phi \)

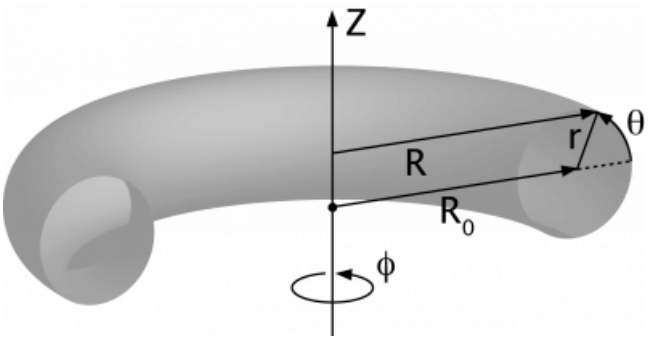
Remaining consistent with ITER, the EU-IM-TF has chosen to adopt the right-handed system

$$(R, \phi, Z)$$

i.e. R is to the right, Z is upwards, and \(\phi \) points into the plane on the right-hand side of the torus (i.e. mathematically positive). Looking from above, the toroidal angle is counter-clockwise, i.e. mathematically positive.

The following figures demonstrate the orientation of the toroidal angle \(\phi \) and the poloidal angle \(\theta \):





source:

http://www-fusion.ciemat.es/fusionwiki/index.php/Toroidal_coordinates

http://en.wikipedia.org/wiki/Toroidal_and_poloidal

9.2.2 Representation of the Magnetic Field and Current

Generally, the magnetic field is described in terms of two scalar fields as it is divergence free. If the field is also axisymmetric then MHD equilibrium demands these are functions of each other. In the

EU-IM-TF the relevant quantities are \(F_{{\rm dia}}\) and \(\Psi \) and the representation is

$$\mathbf{B} = F_{\rm dia}\nabla\phi + (2\pi)^{-1}\nabla\Psi \times \nabla\phi$$

where the factor of \(2 \pi \) is to have \(\Psi \) one and the same with the poloidal flux in Webers (see below).

The current given by Ampere's law is

$$\mu_0\mathbf{J} = \nabla F_{\rm dia} \times \nabla\phi - (2\pi)^{-1}(R^2\nabla \cdot R^{-2}\nabla\Psi)\nabla\phi$$

The respective covariant toroidal components are useful forms:

$$B_\phi = F_{\rm dia} \qquad\qquad \mu_0 J_\phi = -(2\pi)^{-1}(R^2\nabla \cdot R^{-2}\nabla\Psi)$$

where the latter is often expressed in terms of the "delta-star" operator, \(\Delta^* = R^2 \nabla \cdot R^{-2} \nabla \). These are not the toroidal field and current but the toroidal field and current multiplied by \(R \) respectively. The total plasma current \(I_p \) is the integral of \(J_{\phi} / R \) over the poloidal cross section (usually, but not always, over the closed flux surface region only).

9.2.3 Poloidal and Toroidal Fluxes

The toroidal flux \(\Phi \) is the integral of \(B_{\phi} / R \) over the region enclosed by the flux surface. Due to axisymmetry it is also a volume integral

$$\Phi = \oint d^3V(2\pi R^2)^{-1}F_{\rm dia}$$

All volume integrals are understood as integration over the region enclosed by the flux surface. They are therefore flux quantities (pure functions of \(\Psi \)). The units of \(\Phi \) are volt-seconds, or Webers (Wb).

The poloidal flux is \(\Psi \) due to the construction of \(\bf B \). The factor of \(2 \pi \) ensures this is not Wb per radian (the more usual quantity \(\psi \) used as a covariant toroidal component of the magnetic potential is in Wb/radian; the factor of \(2 \pi \) results from integration over one angular circuit). Note that the poloidal flux \(\Psi \) and its equivalent per radian \(\psi \) are often used equivalently in the literature.

9.2.4 Safety Factor

The magnetic pitch parameter is defined in terms of the flux components:

$$q \equiv d\Phi d\Psi$$

which is a flux quantity. This definition is the same as saying the magnetic pitch is given as the number of toroidal cycle a magnetic field line traverses per unit poloidal cycle. It is also called the local safety factor for MHD stability reasons (here, "local" means local to a given flux surface). Equivalent relations often seen depend on the definition of coordinates. These are given for straight field line coordinates, below.

9.2.5 Signs

With the above definition of the toroidal coordinate system and the magnetic field, the following sign relationships ensue (where increasing and decreasing refer to going from the magnetic axis to the separatrix on the outboard midplane):

Table 9.1: Sign Relations

\(B_{tor}\)	\(I_{p}\)	\(\Psi \)	\(\Phi \)	safety factor \(q \)
positive	positive	decreasing	increasing	negative
positive	negative	increasing	increasing	positive
negative	positive	decreasing	decreasing	positive
negative	negative	increasing	decreasing	negative

9.2.6 COCOS - toroidal coordinate conventions

16 different fundamental coordinate conventions (COCOS) has been identified for toroidal systems. These are described by O. Sauter and S. Yu. Medvedev, Computer Phys. Commun. 184 (2013) 293, and summarized in the figure below.

Table 1
(a) Coordinate conventions for each *COCOS* index. $COCOS \leq 8$ refers to ψ divided by (2π) and thus with $e_{Bp} = 0$ while $COCOS \geq 11$ refers to full poloidal flux with $e_{Bp} = 1$. Otherwise $COCOS = i$ and $COCOS = 10 + i$ have the same coordinate conventions. The cylindrical (with the related $\sigma_{R\phi Z}$ value) and poloidal (with $\sigma_{\rho\theta\varphi}$) right-handed coordinate systems are given as well. (b) The indications in this subtable (last three columns) are assuming I_p and B_0 positive in the related coordinate system, that is in the direction of the related φ.

COCOS	e_{Bp}	σ_{Bp}	Cylind, $\sigma_{R\phi Z}$	Poloid, $\sigma_{\rho\theta\varphi}$	φ from top	θ from front
1/11	0/1	+1	$(R, \varphi, Z), +1$	$(\rho, \theta, \varphi), +1$	Cnt-clockwise	Clockwise
2/12	0/1	+1	$(R, Z, \varphi), -1$	$(\rho, \theta, \varphi), +1$	Clockwise	Cnt-clockwise
3/13	0/1	−1	$(R, \varphi, Z), +1$	$(\rho, \varphi, \theta), -1$	Cnt-clockwise	Cnt-clockwise
4/14	0/1	−1	$(R, Z, \varphi), -1$	$(\rho, \varphi, \theta), -1$	Clockwise	Clockwise
5/15	0/1	+1	$(R, \varphi, Z), +1$	$(\rho, \varphi, \theta), -1$	Cnt-clockwise	Cnt-clockwise
6/16	0/1	+1	$(R, Z, \varphi), -1$	$(\rho, \varphi, \theta), -1$	Clockwise	Clockwise
7/17	0/1	−1	$(R, \varphi, Z), +1$	$(\rho, \theta, \varphi), +1$	Cnt-clockwise	Clockwise
8/18	0/1	−1	$(R, Z, \varphi), -1$	$(\rho, \theta, \varphi), +1$	Clockwise	Cnt-clockwise

(a)

ψ_{ref}	sign(q)	sign $\left(\frac{dp}{d\psi}\right)$
Increasing	+1	−1
Increasing	+1	−1
Decreasing	−1	+1
Decreasing	−1	+1
Increasing	−1	−1
Increasing	−1	−1
Decreasing	+1	+1
Decreasing	+1	+1

(b)

The current EU-IM convention (described above) is number 13, while the ITER convention is 11.

9.2.6.1 Determining the COCOS number

9.2.6.2 Equilibrium COCOS transformation library and actor

A Fortran library has been developed for transforming the equilibrium cpo between different COCOS. The source is found in

```
https://gforge6.eufus.eu/svn/numerical_tools/tags/COCOStransform_v1_1
```

and the actor is

```
https://gforge6.eufus.eu/svn/kepleractors/tags/4.09a/imp12/COCOStransformequil.tar
```

(also available from: ~sauter/public/ACTORS/4.09a)

Inputs:

- Equilibrium_in : input cpo
- COCOS_in : COCOS of the input equilibrium (if the COCOS is not stored in Equilibrium_in)
- COCOS_out : Requested COCOS for the Equilibrium_out

V. CHECKING THE CONSISTENCY OF EQUILIBRIUM QUANTITIES/ASSUMPTION WITH A *COCOS* INDEX

Let us obtain conditions of consistency of an input equilibrium with a specific *COCOS* index, generalizing Eq. (22). For this, it is easier to use Eq. (21) and to note that since with the CHEASE normalization we have I_p and B_0 positive, we should have I_p and F positive, ψ_{chease} increasing, $dp/d\psi_{chease}$ negative and q positive (from Table I, $COCOS = 2$ line). Thus, using Eq. (21) we should have for any *cocos* equilibrium:

$$\begin{aligned}
\sigma_{Ip} &= sign(I_p),\\
\sigma_{B_0} &= sign(B_0),\\
sign(F_{cocos}) &= \sigma_{B_0},\\
sign(\Phi_{cocos}) &= \sigma_{B_0},\\
sign\big[\psi_{cocos}(\text{edge}) - \psi_{\text{cocos}}(\text{axis})\big] &= \sigma_{Ip}\,\sigma_{Bp,cocos},\\
sign(\frac{dp}{d\psi}\Big|_{cocos}) &= -\,\sigma_{Ip}\,\sigma_{Bp,cocos},\\
sign(j_{cocos}) &= \sigma_{Ip},\\
sign(q_{cocos}) &= \sigma_{Ip}\,\sigma_{B_0}\,\sigma_{\rho\theta\varphi},
\end{aligned} \tag{23}$$

with $\sigma_{Bp,cocos}, \sigma_{\rho\theta\varphi}$ given in Table I for the related *cocos* value. Note that the sign of $dp/d\psi$ being $-\sigma_{Ip}\sigma_{Bp,cocos}$ should be understood as the "main" $sign(dp/d\psi)$ following the fact that pressure is usually much larger on axis than at the edge. To be more precise one could replace this relation by $sign\big(\sum_0^{\text{edge}} \frac{dp}{d\psi}\,\Delta\psi\big) = -1$.

It should be noted that Eq. (23) can also be used to determine the *COCOS* used in a code or set of equations. Usually, one starts by checking if ψ is increasing or decreasing from magnetic axis to the edge. Then, depending on $sign(I_P)$, one can obtain the value of $\sigma_{Bp,cocos}$. Another way is if $\mathbf{B}_p \sim \nabla\varphi \times \nabla\psi$, thus $\sigma_{Bp,cocos} = +1$ or $\mathbf{B}_p \sim \nabla\psi \times \nabla\varphi$, yielding $\sigma_{Bp,cocos} = -1$. Then one can check with the sign of $dp/d\psi$. The next step is to determine $\sigma_{R\varphi Z}$, either from the comparison of the sign of I_p and B_0 with the effective direction of I_p and B_0 if it is known, or by comparing the definition of B_R, for example, with Eqs. (12) and (13) and taking into account the value of σ_{Bp}. Then, the effective sign of q gives the value of $\sigma_{\rho\theta\varphi}$. Finally, e_{Bp} is obtained from the factor 2π appearing either in the definition of $\mathbf{B}_p$, Eq. (1), giving $e_{Bp} = 1$ or in the definition of q, Eq. (9), yielding $e_{Bp} = 0$. Note that if a

- Ipsign_out : Requested sign for output Ip; -9 if just wants IP_in transformed to new equilibrium, +1 or -1 if a specific sign in output is desired
- B0sign_out : Requested sign for output B0

Output:

- Equilibrium_out : Output cpo

9.2.7 The Flux Surface Average

In general, the flux surface average is the operation which annihilates the magnetic derivative \({\bf B} \cdot \nabla \) and acts as an identity operator on any flux quantity. It can be proved that this results in a volume derivative of a volume integral (alternatively one starts with the latter property and then proves the former, as the above Ciemat reference does). The flux surface average of a scalar and divergence of a vector are given by

$$\langle G \rangle = \frac{\partial}{\partial V} \oint d^3V G \qquad\qquad \langle \nabla \cdot \mathbf{G} \rangle = \frac{\partial}{\partial V} \langle \mathbf{G} \cdot \nabla V \rangle$$

where \({\bf G} \cdot \nabla V \) is the contravariant volume component of the vector \({\bf G} \). It follows that the flux surface average is an angle average weighted by the volume element \(\sqrt{g} \)

$$\langle G \rangle = \oint d\phi \oint d\theta \sqrt{g} G \Big/ \oint d\phi \oint d\theta \sqrt{g}$$

for any choice of toroidal and poloidal angle as well as radial coordinates, where \(g \) is the determinant of the covariant metric tensor components in those coordinates. Note in general \(G \) is not an axisymmetric quantity so the integration is actually over both angles.

For more detail see the above references.

9.2.8 The Toroidal Flux Radius as the Radial Coordinate

The EU-IM-TF has decided to use the toroidal flux radius \(\rho_{{ \rm tor}} \) defined by

$$\Phi = \pi B_0 \rho_{\rm tor}^2$$

where \(B_0 \) is the reference (vacuum) magnetic field value. Note that \(\rho_{{ \rm tor}} \) is a positive quantity which has units of meters. For several applications the volume radius \(\rho_{{ \rm vol}} \) is also used. It is a normalised radius going from 0 to 1 and is defined as

$$V = V_{\rm LCFS} \rho_{\rm vol}^2$$

where LCFS refers to the last closed flux surface. Both should be defined in the equilibrium CPO (as well as \({ \tt volume} \equiv V \) itself).

9.2.9 Toroidal and Parallel Current

These are not equivalent, despite the often-seen experimental practice of considering them so. The toroidal current given in Amperes depends on some convention applied to \(J_{\phi} \) given above, which is not a flux quantity. The EU-IM-TF has decided on this definition of the toroidal current as a flux quantity:

$$\texttt{jphi} \equiv \langle J^{\phi} \rangle / \langle 1/R \rangle$$

This uses the contravariant toroidal component of \(\bf J \) which is a pure divergence

$$J^\phi = \mathbf{J} \cdot \nabla\phi = J_\phi / R^2 = -\nabla \cdot (2\pi\mu_0 R^2)^{-1} \nabla\Psi$$

Hence the flux surface average invokes the often-used quantity \(\langle g^{\rho \rho}/R^2 \rangle \) in the form

$$\langle J^\phi \rangle = -(2\pi\mu_0)^{-1} \frac{1}{V'_\rho} \frac{\partial}{\partial\rho} V'_\rho \langle g^{\rho\rho}/R^2 \rangle \frac{\partial\Psi}{\partial\rho}$$

Here, \(V'_\rho \equiv \partial V / \partial \rho_{\rm tor}\) explicitly using the toroidal flux radius as the radial coordinate.

The parallel current is different from this due to the finiteness of the poloidal current and magnetic field. Generally the correction is \(O(\epsilon^2 / q^2) \) which is usually a few percent (but not in a spherical tokamak). Using the representations for \(\bf B \) and \(\bf J \) given above we find

$$\mathbf{J} \cdot \mathbf{B} = -(2\pi\mu_0)^{-1} F_{\rm dia}^2 \nabla \cdot \frac{1}{F_{\rm dia} R^2} \nabla\Psi$$

Since \(F_{\rm dia} \) is a flux quantity the flux surface average behaves as for \({\tt jphi} \) and we use a factor of \(B_0 \) to provide the correct units, yielding

$$\texttt{jparallel} \equiv -(2\pi\mu_0 B_0)^{-1} \frac{F_{\rm dia}^2}{V'_\rho} \frac{\partial}{\partial\rho} \frac{V'_\rho}{F_{\rm dia}} \langle g^{\rho\rho}/R^2 \rangle \frac{\partial\Psi}{\partial\rho}$$

This form has been chosen due to the natural use of the flux surface average \(\langle{\bf J} \cdot{ \bf B} \rangle \) in neoclassical theory and the magnetic flux diffusion equation (see the Hinton and Hazeltine reference above).

9.2.10 Straight Field Line Coordinates

A variety of modules in the EU-IM-TF use straight field line coordinate systems to represent the closed flux surface region. To guarantee consistency with the definition of the poloidal flux and the magnetic field representation given above, a standard definition of the coordinate volume element follows. This is the same sense as the usage of the term "Jacobian" in the CPOs (note many papers use the inverse volume element as the "Jacobian" by contrast). Here, "straight field line coordinates" refers to the use of the right-handed coordinate system \((\Psi, \theta, \zeta) \) with the poloidal flux \(\Psi \), the straight field line angle \(\theta \), and the toroidal angle \(\zeta = - \phi \). Therefore, \(\theta \) has the same orientation as the poloidal angle \(\theta \) in toroidal coordinates, while the toroidal angle \(\zeta \) is in the opposite direction of \(\phi \). This is standard usage generally in terms of "flux coordinates" (see Hazeltine and Meiss, above).

Note here that while the toroidal angle is the geometric one in the orientation sense of flux coordinates, the poloidal angle is not geometric. This results from the demand that the field lines be straight in the coordinate plane \((\theta, \zeta) \). The definition of this property is given by the specification of the ratio of contravariant components of the magnetic field as a flux quantity, which is one and the same with the pitch parameter ("local safety factor"):

$$q = q(\Psi) = -B^\zeta / B^\theta = B^\phi / B^\theta$$

where the minus sign appears by consistency with the primary definition in terms of the flux components as given above. This represents a magnetic differential equation for the poloidal angle:

$$B^\theta = B^\phi / q = F_{\rm dia} / q R^2$$

Due to the choice of "natural" coordinates (with \(\Psi \), not \(\rho_{{\rm tor}} \)) this relation is close to the definition of the volume element \(\sqrt{g} \) and, equivalently, the Jacobian \(J \)

$$J \equiv \sqrt{g} \qquad J^{-1} = \nabla\Psi \cdot \nabla\theta \times \nabla\zeta = \nabla\Psi \times \nabla\phi \cdot \nabla\theta$$

Note the ordering of \({\bf \nabla} \Psi \) and \({\bf \nabla} \phi \).

The components of the magnetic field are then

$$\begin{aligned} B^\theta &= \mathbf{B} \cdot \nabla\theta = (2\pi)^{-1} \nabla\Psi \times \nabla\phi \cdot \nabla\theta = (2\pi J)^{-1} \\ B^\zeta &= \mathbf{B} \cdot \nabla\zeta = -B_\phi / R^2 = -F_{\rm dia}/R^2 \\ B^\Psi &= \mathbf{B} \cdot \nabla\Psi = 0 \end{aligned}$$

With these relations the following relationship between the Jacobian and pitch parameter ("local safety factor") holds

$$J = (2\pi)^{-1} q R^2 / F_{\rm dia}$$

This is the quantity labelled \({\tt jacobian} \) in the equilibrium CPO.

9.2.11 Plasma Betas

Out of the many definitions of plasma betas, the EU-IM has agreed to adhere to the following definitions: Following Wesson (p. 116), the poloidal beta is defined as an integral over the poloidal cross section

$$\beta_{\rm p} = \frac{2\mu_0}{B_{\rm a}^2} \frac{\int_A p {\rm dS}}{\int_A {\rm dS}}$$

where \(A = A (\Psi) \) is the poloidal cross section enclosed by the flux surface \(\Psi \), \(B_{\rm a} = \frac{ \mu_{0} I}{l} \) is the flux surface averaged poloidal magnetic field, \(I = I(\Psi) \) the toroidal plasma current inside the flux surface \(\Psi \) and \(l = \oint \rm{d} l \) the length of the poloidal perimeter of flux surface \(\Psi \). This definition yields a one-dimensional profile \(\beta_{\rm p} = \beta_{\rm p} (\Psi) \) stored in profiles_1d%beta_pol in the equilibrium CPO. The overall poloidal beta \(\beta_{rm p} (\Psi = \Psi_{\rm bd}) \) is stored in global_param%beta_pol.

The toroidal beta is defined as

$$\beta_{\rm tor} = \frac{2\mu_0}{B_0^2} \frac{\int_\Omega p {\rm dV}}{\int_\Omega {\rm dV}}$$

with \(B_{0} \) the vacuum magnetic field as stored in global_param%toroid_field%b0. The integral is carried out over the entire plasma volume and the result stored in global_param%beta_tor.

The normalized plasma beta is defined as

$$\beta_{\rm N} = 100 \frac{a B_0}{10^{-6} I_{\rm p}} \beta_{\rm tor}$$

with \(I_{\rm p} \) the total plasma current (following Y.-S. Na et al., PPCF 44 (2002), 1285) and a is the minor radius. It is stored in global_param%beta_normal.

9.2.12 Internal Inductance

The definition of the internal inductance follows J.A. Romero et al., NF 50 (2010), 115002. The magnetic energy contained inside the flux surface \(\Psi \) is

$$W_{\mathrm{mag}} = \frac{1}{2\mu_0}\int_{\Omega} B_{\mathrm{p}}^2 \mathrm{dV}$$

where \(B_{\rm p} \) is the poloidal component of the magnetic field. The (unnormalized) internal inductance is then defined as

$$L_{\mathrm{i}} = \frac{2W_{\mathrm{mag}}}{I^2}$$

where \(I = I(\Psi) \) is the toroidal plasma current enclosed by the flux surface \(\Psi \). The normalized internal inductance, as stored in profiles_1d%li is defined as

$$l_{\mathrm{i}} = \frac{2L_{\mathrm{i}}}{\mu_0 \bar{R}}$$

with the surface averaged major radius

$$barR = \frac{\int_A R\mathrm{dS}}{\int_A \mathrm{dS}} = \frac{V(\Psi)}{2\pi A(\Psi)}$$

The overall internal inductance \(l_{\rm i} (\Psi = \Psi_{\rm bd}) \) is stored in global_param%li.

9.2.13 Poloidal Angle Dimension in Equilibrium CPO

The following entries in the equilibrium CPO are defined along the poloidal dimension (as dim2 in the case of a flux surface equilibrium, i.e. radial coordinate psi in dim1 and poloidal angle in dim2):

```
coord_sys%jacobian(:,:)
coord_sys%g_11(:,:)
coord_sys%g_12(:,:)
coord_sys%g_13(:,:)
coord_sys%g_22(:,:)
coord_sys%g_23(:,:)
coord_sys%g_33(:,:)
profiles_2d%position
profiles_2d%grid
profiles_2d%psi_grid(:,:)
profiles_2d%jphi_grid(:,:)
profiles_2d%jpar_grid(:,:)
profiles_2d%br(:,:)
profiles_2d%bz(:,:)
profiles_2d%bphi(:,:)
```

The EU-IM-TF has decided not to repeat the first poloidal point (with poloidal angle \(\theta = 0\)), which is identical to \(\theta = 2 \pi\). This option was chosen to facilitate Fourier transforms along the poloidal direction. To that purpose it is required that the dimension dim2 be equidistant in the poloidal angle \(\theta \) (going from \(\theta = 0 \) to \(\theta = (ndim2-1)/ndim2*2 \pi \) where ndim2 is the number of poloidal grid points), whatever the choice of this angle is.

9.3 Numerical and computational conventions

9.3.1 Standardized Variable Types

To ensure that physics modules produce identical results on various computer architectures and to avoid issues with double precision versus single precision interfaces, the EU-IM-TF has agreed on a set of

standardized variable types. It is recommended that these types be used throughout all EU-IM modules, but at least for the interface definitions. The Fortran90 module defining the type standards itm_types.f90 is hosted by the project itmshared . To check out the relevant files please do

```
svn checkout https://gforge6.eufus.eu/svn/itmshared/trunk/src/itm_types target_dir
```

For Fortran90, the following standard types have been defined

```
INTEGER,  PARAMETER :: EU-IM_I1 = SELECTED_INT_KIND (2)        ! Integer*1
INTEGER,  PARAMETER :: EU-IM_I2 = SELECTED_INT_KIND (4)        ! Integer*2
INTEGER,  PARAMETER :: EU-IM_I4 = SELECTED_INT_KIND (9)        ! Integer*4
INTEGER,  PARAMETER :: EU-IM_I8 = SELECTED_INT_KIND (18)       ! Integer*8
INTEGER,  PARAMETER :: R4 = SELECTED_REAL_KIND (6, 37)   ! Real*4
INTEGER,  PARAMETER :: R8 = SELECTED_REAL_KIND (15, 300) ! Real*8
```

To implement these types in your code, please add the following line to your modules

```
use itm_types
```

Compiled versions of the module can be found in

```
$EU-IMLIBDIR/itmtypes/lib/$OBJECTCODE
```

where the following values of OBJECTCODE are supported

```
amd64_g95_0.92
amd64_gfortran_4.7
amd64_intel_12
amd64_pgi_10
```

(More information about the EU-IM libraries.)

9.3.2 Standardized Physical Constants

To avoid discrepancies in simulations from using different definitions of the physical constants, the EU-IM-TF has agreed upon a set of standardized physical constants (all in SI units except for temperatures) based on the NIST recommendations . It is recommended that these constant be used throughout all EU-IM modules. The Fortran90 module defining the standardized physical constants itm_constants.f90 is hosted by the project itmshared . To check out the relevant files please do

```
svn checkout https://gforge6.eufus.eu/svn/itmshared/trunk/src/itm_constants target_dir
```

Compiled versions of the module can be found in

```
$EU-IMLIBDIR/itmconstants/lib/$OBJECTCODE
```

where the following values of OBJECTCODE are supported

```
amd64_g95_0.92
amd64_gfortran_4.7
amd64_intel_12
amd64_pgi_10
```

The C equivalent (“itm_constants.h”) can be found in

```
$EU-IMLIBDIR/itmconstants/include/
```

and the Python in

```
$EU-IMLIBDIR/itmconstants/lib/python2.6/
```

A Java version is available but has not yet been released — contact ISIP if you are interested. (More information about the EU-IM libraries .) Thefollowing constants are available: itm_constants.xml All constants are double precision floats (R8).

9.3.3 Invalid Data Base Entries

The EU-IM data base does not allow for setting data base entries directly to invalid in case they should not be set. Since the Universal Access Layer (UAL) always pulls out complete CPOs, i.e. complete data structures, of which not all fields may be filled, the problem arose of how to identify those fields which have not been filled. In the case of arrays, this is simply done by not associating the corresponding pointer. In the case of scalars, however, unique values for floats and integers had to be defined to identify empty fields. These values identify invalid data base entries and can be tested through comparison. The values for invalid data base entries in Fortran90 are defined below:

```
INTEGER,  PARAMETER :: itm_int_invalid = -999999999
REAL(R8), PARAMETER :: itm_r8_invalid = -9.0D40
```

They have been found to be safely out of any physical range for the affected fields such that no accidental confusion with real values may occur. The Fortran90 module defining these values itm_types.f90 is hosted by the project itmshared . To check out the relevant files please do

```
svn checkout https://gforge6.eufus.eu/svn/itmshared/trunk/src/itm_types target_dir
```

The module also includes three functions of type boolean itm_is_valid_int4 , itm_is_valid_int8 , and itm_is_valid_real8 which are overloaded under the interface itm_is_valid to check whether a data base entry has been filled. Example:

```
if (itm_is_valid(equilibrium%global_param%i_plasma)) then
  write(*, *) 'Plasma current Ip = ', equilibrium%global_param%i_plasma
end if
```

9.3.4 Enumerated datatypes/Identifiers

This section concerns how to specify the origin of data in certain types of CPOs. The specification is performed using the datatype identifier. The following specifies the conventions of the allowed enumerated datatypes.

- cocos_identifier.xml
- coordinate_identifier.xml
- coredelta_identifier.xml
- coreneutral_identifier.xml
- coresource_identifier.xml
- coretransp_identifier.xml
- distsource_identifier.xml
- fast_particle_origin_identifier.xml
- fast_thermal_filter_identifier.xml

- fokker_planck_source_identifier.xml
- pellet_shape_identifier.xml
- species_reference_identifier.xml
- wall_identifier.xml
- wave_identifier.xml

Compiled versions of the modules can be found in

```
$EU-IMLIBDIR/itmconstants/lib/$OBJECTCODE
```

where the following values of OBJECTCODE are supported

```
amd64_g95_0.92
amd64_gfortran_4.7
amd64_intel_12
amd64_pgi_10
```

The C equivalent can be found in

```
$EU-IMLIBDIR/itmconstants/include/
```

and the Python in

```
$EU-IMLIBDIR/itmconstants/lib/python2.6/
```

A Java version is available but has not yet been released — contact the CPT if you are interested. (More information about the EU-IM libraries .)

9.3.4.1 Example: How to fill coresource/values/sourceid

When filling in an enumerated datatype, like coresource/values/sourceid, it is recomended to use the parameters and functions built into the fortran modules associated with each such datatype. These modules are available as part of the UAL package. As an examples we may include the coresource_identifier:

```
use coresource_identifier, only: fusion, get_type_name, get_type_description__ind
```

Here the value of the integer-parameter fusion is the Flag for fusion reactions in the *coresource_identifier* structure (i.e. fusion=5). Once we know the Flag we may get the Id using the function Id=get_type_name(Flag) and the Description using the function Description=get_type_description__ind(Flag). These function are available for every datatype.

Below you have an example of how to use these functions:

```
program coresource_example use euitm_schemas, only: type_coresource use
  coresource_identifier, only: fusion, get_type_name,
  get_type_description__ind use write_structures, only: open_write_file,
  write_cpo, close_write_file use deallocate_structures, only:
  deallocate_cpo implicit none

  type (type_coresource) :: coresource
  integer :: idx, i

  character*128 :: filename
  integer :: shot, run

  data filename / &
       & 'coresource.cpo' &
```

(continues on next page)

(continued from previous page)

```
      & /

  allocate(coresource%values(1))
  allocate(coresource%values(1)%sourceid%id(1))
  allocate(coresource%values(1)%sourceid%description(1))
  coresource%values(1)%sourceid%flag = fusion
  coresource%values(1)%sourceid%id = get_type_name(fusion)
  coresource%values(1)%sourceid%description =
  get_type_description__ind(fusion)

  call open_write_file(1, filename)
  call write_cpo(coresource, 'coresource')
  call close_write_file

  call deallocate_cpo(coresource)

end program coresource_example
```

This example program, and similar examples for other enumerated datatypes, are available in:

```
https://gforge6.eufus.eu/svn/itmshared/trunk/src/itm_constants/examples
```

9.3.5 Grid Types in Equilibrium CPO

Equilibria may be represented in a variety of different ways depending on which EU-IM module has calculated them and which module shall use them. To avoid ambiguity and to allow modules to check which type of equilibrium is stored in the equilibrium CPO, a unique grid identifier is stored in profiles_2d%grid_type. The grid identified currently consists of 4 strings (at 132 chars) with the following structure (array indices in Fortran notation):

Position	Content
grid_type(1)	integer identifier for grid type
grid_type(2)	string identifier for grid type
grid_type(3)	integer identifier for poloidal angle
grid_type(4)	string identifier for poloidal angle

9.3.5.1 Grid Type Identifier

The currently allowed values (integer and string) for the identifier of the grid type are listed below:

Integer Values	String Value	Description
1	rectangular	Regular grid in \((R, Z) \).'EFIT-like grid'
2	inverse	Regular grid in \((\Psi, \theta)\).'flux surface grid'.
3	irregular	Irregular grid. All fields in profiles_2d are given as (ndim1, 1) degenerate 2D matrices, i.e. as lists of vertices (for triangles or quadrilaterals).

9.3.5.1.1 Poloidal Angle Identifier

The currently allowed values (integer and string) for the identifier of the poloidal angle are listed below:

Integer Values	String Value	Description
1	straight field line	straight field line angle \(\theta \) as defined in Straight Field Line Coordinates
2	equal arc	Poloidal angle \(\theta \) defined by equal arc lengths along flux surfaces
3	polar	Poloidal angle \(\theta \) in toroidal coordinates as defined in Coordinate System

9.3.6 Standardized EU-EU-IM Plasma Bundle

The EU-IM has agreed on a standardized way to bundle CPOs and control parameters inside KEPLER.

Field names			*Type*	*Description*
time			real	The synthetic time of the simulation, or for time-dependent workflows; the end of the present time step. For example, consider a time dependent workflows, where physics quantities are update one after the other. Thus, while the physics quantities are updated the various fields below (e.g. the CPOs) may be describe at different time points. In such workflows the this "time"-field describe the time at the end of the present time step. Units: (s)
CONTROL	tau		real	time-step (s)
	tau_out		real	time interval for saving output (s)
	ETS	amix	real	mixing factor
		amix_tr	real	mixing factor for profiles

Continued on next page

Table 9.2 – continued from previous page

Field names			*Type*	*Description*
		sigma_source	integer	option for origin of plasma electrical conductivity: 0: plasma collisions; 1: transport module; 2: source module
		solver_type	integer	choice of numerical solver
		conv_rec	real	required fractional convergence
CPOS	MHD	equilibrium	cpo	see type and fortran descriptions
		toroidfield	cpo	see type and fortran descriptions
		mhd	cpo	see type and fortran descriptions
		sawteeth	cpo	see type and fortran descriptions
	CORE	coreprof	cpo	see type and fortran descriptions
		coretransp	cpo	see type and fortran descriptions
		coresource	cpo	see type and fortran descriptions
		coreimpur	cpo	see type and fortran descriptions
		coreneutral	cpo	see type and fortran descriptions
		corefast	cpo	see type and fortran descriptions
		coredelta	cpo	see type and fortran descriptions
		compositionc	cpo	see type and fortran descriptions
		neoclassic	cpo	see type and fortran descriptions
	EDGE	edge	cpo	see type and fortran descriptions
	HCD	waves	cpo	see type and fortran descriptions
		distsource	cpo	see type and fortran descriptions

Continued on next page

Table 9.2 – continued from previous page

Field names			*Type*	*Description*
		distribution	cpo	see type and fortran descriptions
	MACH	vessel	cpo	see type and fortran descriptions
		wall	cpo	see type and fortran descriptions
		nbi	cpo	see type and fortran descriptions
		antennas	cpo	see type and fortran descriptions
		ironmodel	cpo	see type and fortran descriptions
		pfsystems	cpo	see type and fortran descriptions
	DIAG	fusiondiag	cpo	see type and fortran descriptions
		scenario	cpo	see type and fortran descriptions
	EVENTS	pellets	cpo	see type and fortran descriptions
PCS	input		pcs_in	Diagnostics input signals to the plasma control system (see comple-type definition below)
	reference		pcs_ref	Reference signals for the plasma control system (see comple-type definition below)
	output		pcs_out	Output signals from plasma control system (see comple-type definition below)

The complex-types used in the PCS.

Field names	*Type*	*Description*
pcs_in (under development)		*Diagnostics for plasma control*
pcs.inputs.plasma_variables	type_plasma_variables	Plasma variables
pcs.inputs.plant_variables	type_plant_variables	Plant variables
pcs_ref (under development)		*Reference signals for plasma control*
pcs.reference.plant_variables	type_plant _variables	Plant variables
pcs.reference.plant_configuration	type_plant _configuration	Plant configuration
pcs_out (under development)		*Output signal for plasma control*
pcs.output.plasma_variables	type_plasma _variables	Plasma variables
pcs.output.plant_variables	type_plant _variables	Plant variables. NOTE: only for artificial control.
type_plasma_variables (under development)		*Plasma properties relevant for plasma control*
plasma.shape.ZIP	float	Z_centre*Ip (used for vertical control; definition of Z_centre can vary) [Am]
plasma.shape.gaps(:)	float	Distance between the plasma and the wall components [m]
plasma.magnetics.b_toroidal	real	Toroidal magnetic field at the magnetic axis [T]
plasma.magnetics.Ip	real	Current (A) CPO element: equilibrium().global_param.current_tot
plasma.magnetics.v_loop	real	Loop voltage (V) CPO element: coreprof().profiles1d.vloop.value
plasma.confinement.ne_line_integrated	real	

CHAPTER

TEN

KEPLER

10.1 Introduction to Kepler - basics

Kepler is a workflow engine and design platform for analyzing and modeling scientific data. Kepler provides a graphical interface and a library of pre-defined components to enable users to construct scientific workflows which can undertake a wide range of functionality. It is primarily designed to access, analyse, and visualise scientific data but can be used to construct whole programs or run pre-existing simulation codes.

Kepler builds upon the mature Ptolemy II framework, developed at the University of California, Berkeley. Kepler itself is developed and maintained by the cross-project Kepler collaboration.

The main components in a Kepler workflow are actors, which are used in a design (inherited from Ptolemy II) that separates workflow components ("actors") from workflow orchestration ("directors"), making components more easily reusable. Workflows can work at very levels of granularity, from low-level workflows (that explicitly move data around or start and monitor remote jobs, for example) to high-level workflows that interlink complex steps/actors. Actors can be reused to construct more complex actors enabling complex functionality to be encapsulated in easy to use packages. A wide range of actors are available for use and reuse.

10.1.1 Installing Kepler with Serpens add-on

You can download Kepler from the following page https://kepler-project.org/users/downloads

In order to install Kepler and Serpens related workflows you have to do following:

```
cd ~
wget http://scilla.man.poznan.pl/euforia/install/serpens-demo-workflows.zip -O serpens-demo-
 ↪workflows.zip
unzip serpens-demo-workflows.zip
```

Now you can start Kepler application and proceed to tutorial examples that can be find at the following page http://scilla.man.poznan.pl/confluence/display/euforia/01.+Introduction+to+Kepler+-+basics

10.2 Kepler IMAS actors

Imas actor	Deacription
UALSliceCollector	Store one slice from input IDS into different run. This way, it is possible to collect intermediate results during workflow execution.
UALPython	Allows to run external Python process and pass input/output data between workflow and process itself. This actor is, most commonly, used for data visualization. User can pass Python script directly to the actor.
UALMuxParam	Provides similar behavior to ualmux/UALMux, however, this actors has two additional ports: - fieldDescription - contains name of the filed that will be modified - fieldValue - contains new value of the field Main difference between ualmux/ ualmuxparam actors lays in it's ability to be used in a loop that modify different field inside IDS. You can simply provide different field name for different loop's step.
UALMux	Provides a method for putting data inside IDS inputs - inputIds/inputCpo - cpo we are going to modify - inTime - time index at which data are supposed to be updated - name of the field is specified as port name - new value of the field is passed as value sent to the port outputs: - outputIds/outputCpo - modified IDS - outTime - actual time index (depend on approximation mode)
UALInit	Initializes input pulse file, creates run work and provides ID S description for other actors inputs:

10.3 IMAS Kepler based configuration

10.3.1 Running Kepler using IMAS environment

10.3.1.1 Setting up environment

Please do not forget to set JAVA memory settings:

```
export _JAVA_OPTIONS="-Xss20m -Xms8g -Xmx8g"
```

10.3.1.2 Backing up old files

Before first configuration of Kepler, make sure to backup your old data files

```
cd ~
mv .kepler .kepler~
mv KeplerData KeplerData~
mv .ptolemyII .ptolemyII~
```

10.3.1.3 Creating place to store your personal installations of Kepler

IMAS based installations are stored inside **$HOME/kepler** directory.

Before proceeding further, make sure to create **kepler** directory

```
# create directory inside $HOME
cd ~
mkdir kepler
```

10.3.1.4 Running Kepler (default release)

In order to start Kepler you have use helper scripts that will install and configure your personal copy of Kepler

- load IMAS module

```
module load imas
module load kepler
# NOTE! It might be that you don't have Kepler copy inside your $HOME
# in that case you need to install it kepler_install_light
```

- Start Kepler

```
# run alias that will execute Kepler
kepler
```

CHAPTER

ELEVEN

EDRG

EDRG (EXPERIMENTALISTS AND DIAGNOSTICIANS RESOURCE GROUP) is one of the two projects falling under the coordination of the Leadership of the EU Task Force for Integrated Tokamak Modelling (EU-IM) and is the privileged contact of the EU-IM with the experimentalists and diagnosticians community.

11.1 Scientific Rationale and Main Objectives

The consolidation of the validated suite of simulation tools that the EU-IM aims to provide for ITER and existing experiments requires a strong interaction with the experimentalists and diagnosticians fusion community. The former are promoted by the Experimentalist and Diagnosticians Resource Group (EDRG). Acting as a contact point within the EU-IM towards the full range of experiments and some of the EFDA Topical Groups and Working Group initiatives, the EDRG group promotes the provision of a machine independent approach to modelling, to encompass realistic operational conditions and to facilitate verification and validation of the modelling codes. The groups action comprises

1. Developing a comprehensive set of Machine descriptions and experimental data mappings to access experimental databases from european devices.

2. The coordination of the overall plasma control activities to be carried within the EU-IM-TF and in liaison with other EFDA initiatives.

3. The integration of synthetic diagnostic modules to assist Verification and Validation of EU-IM modules and virtual PCS.

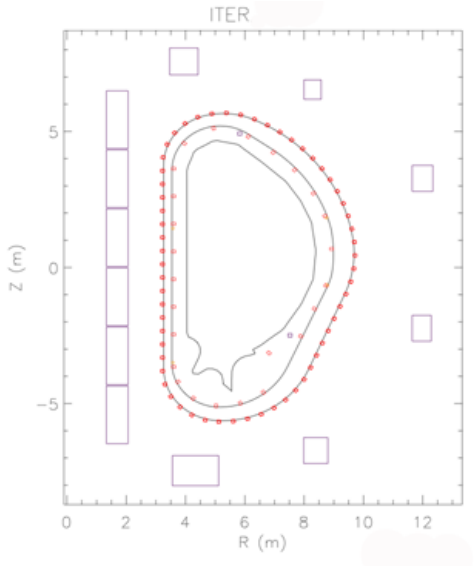
11.2 Machine Descriptions and Data Mappings

At the forefront of the EDRG activities, the consolidation of the machine descriptions and data mappings for the experimental data on the participating devices is of utmost importance.

11.2.1 Machine Descriptions

11.2.1.1 Background

The machine description (MD) of a device basically builds on the set of engineering and diagnostic settings characterising any device. This includes, for instance, the vessel/limiter description (for the moment in R-Z domain only), the PF coils and circuiting and lines of sight of diagnostics (an example for ITER is seen below evidencing the vessel+limiter+pfsystems and magdiad CPOs).



In practice, all MD information is encapsulated in a XML file that emanates from the MD tagged datastructure schemas. A MD instance of a given device is then stored into the EU-IM db as shot 0 for that device database. At data structure version 4.09a, the list of CPOs with machine description tags is indicated in the Figure below

11.2.1.2 MD content on dataversions

4.07b	4.08a	4.08b	4.09a
antennas, interfdiag, ironmodel, limiter, magdiag, msediag, pfsystems, polardiag, toroidfield, vessel	antennas, ecediag, interfdiag, ironmodel, limiter, magdiag, msediag, pfsystems, polardiag, reference, toroidfield, tsdiag, vessel	antennas, ecediag, interfdiag, ironmodel, limiter, magdiag, msediag, nbi, pfsystems, polardiag, toroidfield, tsdiag, vessel	antennas, ecediag, fusiondiag, interfdiag, ironmodel, langmuirdiag, limiter, magdiag, msediag, nbi, pfsystems, polardiag, toroidfield, tsdiag, vessel

11.2.1.3 Tutorial and specific information on some CPOs

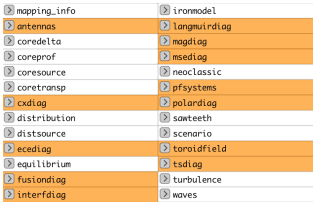
- A good introduction to the machine description concept is found in this User Guide .

- Up to data version 4.07c, the geometry of lines of sight used in some diagnostics (interfdiag and polardiag) was characterized by two angles (poloidal and toroidal). A caveat was found, leading to the adoption of a new set of angles as described in this report . The pivot points became 3 (previously 2) to encompass slightly different entry/exit positions for the reflected beams.
- From data version 4.09a, the limiter, nbi and antennas CPOs use arrays of structures data types. This enabled a more refined description of plasma facing components and the divertor region and both open and closed 'tile' elements. Each NBI and antenna unit is also a dedicated structure.
- Detailed definitions are available for Flux loop position , PFconnections , Langmuir probes and Fusion CPO .

11.2.2 Data Mappings

11.2.2.1 Background

In regard to the experimental data from a particular shot/device set, the EU-IM had to develop its own tool to retrieve the data from the experimental databases and populate the corresponding EU-IM db entry since the latter rarely adopt the same datastructure ontology and different methods/implementation for the databases might exist on different devices. A XML file contains all the mapping essentials, e.g. download method, local signal names, gains and offsets, time base and eventual interpolation option to ensure that only one time base is set for each CPO that is built from multiple local signals. A java code (exp2EU-IM developed under ISIP), with the MD and DM files as inputs, is then run to connect to the local device database, retrieve the required experimental data and populate the EU-IM db instance for that shot/device and dataversion. At data structure version 4.09a, the list of CPOs with data mappings tags is indicated in the Figure below (experimental signals are colored in orange; mappings to other CPOs, e.g. equilibrium or coreprof have been set in order to assist the retrieval of simulated data from other databases, e.g. JSP, JSPC)



11.2.2.2 DM content on dataversions

4.07b	4.08a	4.08b	4.09a
antennas, interfdiag, ironmodel, magdiag, msediag, pfsystems, polardiag, toroidfield	antennas, cxdiag, ecediag, interfdiag, ironmodel, magdiag, msediag, nbi, pfsystems, polardiag, toroidfield, tsdiag, coredelta, coreprof, coretransp, distribution, distsource, equilibrium, neoclassical, sawteeth, scenario, waves	antennas, cxdiag, ecediag, interfdiag, ironmodel, magdiag, msediag, nbi, pfsystems, polardiag, toroidfield, tsdiag, coredelta, coreprof, coretransp, distribution, distsource, equilibrium, neoclassical, sawteeth, scenario, turbulence, waves	antennas, cxdiag, ecediag, fusiondiag, interfdiag, ironmodel, langmuirdiag, magdiag, msediag, nbi, pfsystems, polardiag, toroidfield, tsdiag, coredelta, coreprof, coretransp, distribution, distsource, equilibrium, neoclassical, sawteeth, scenario, turbulence, waves

11.2.2.3 Tutorial on data mappings

- A good starting point to understand the basics of the data mapping concept and how to fill it with the device dependent referecing of the experimental data is found in this User Guide .
- A description on the data mapping concept and processing by exp2EU-IM (with usage tips) is found in this presentation .

For more updated information on the MD and DM activity please check the md_and_dm project in Gforge

CHAPTER

TWELVE

AMNS

12.1 Scientific Rationale and Main Objectives

The EU-IM has a broad need for data relating to atomic, molecular, nuclear and surface data (AMNS). In particular, AMNS data are needed in several of the EU-IM modelling projects. A consistent approach, taking into account the specific requirements of the EU-IM while maintaining the work aligned with other European efforts in this area, is therefore required. As a consequence the AMNS tasks are implemented as Tasks under the TF leadership and has the following scope:

- Coordination of the work in the four different sub areas.
- Supply of data not presently residing in easily accessible data bases.
- Identify any Intellectual Property Rights (IPR) protection needs in view of a broader collaboration with ITER partners.
- Provide software for delivery of AMNS data to EU-IM-TF codes.

12.2 EU-IM contact person

David Coster email: David.Coster@ipp.mpg.de

12.3 AMNS tasks

The AMNS work is divided into two broad areas:

- The maintenance and development of the AMNS library (and the associated AMNS CPO) to provide access to AMNS data in the various languages used by the codes within the Work Package
- The addition to the AMNS database of AMNS data needed by the codes within the Work Package

12.4 Private AMNS pages

For access to the private AMNS pages, an AMNS password is needed.

12.5 AMNS Documentation

The AMNS library is meant to be called by Work Package codes if the codes need data for Atomic, Molecular, Nuclear or Surface processes. The calling sequence is described in more detail below, but the basic idea is: (1) initialize the package; (2) request data for a particular reaction by initializing a "table" for that reaction; (3) (repeatedly) requesting data for that reaction as a fuction of plasma or other parameters; (4) finishing with the table; and (5) finishing with the AMNS library.

The actual AMNS data is provided by CPOs stored under the "amns" tokamak and will first be searched for in the user's database, and if not found there, the system will default to obtaining the data from the public AMNS database. Multiple versions of the AMNS data are possible: in 4.09a and 4.09b this was done via a mysql database; in 4.10a and later this is done by having an index block stored in shot 0, run 1 of the AMNS CPO.

Some presentations:

- Nuclear reactions (pdf), by V. Kiptily
- Simulations of the edge plasma: the role of atomic, molecular and surface physics (pdf), by D. P. Coster, S. Gori, X. Bonnin, D.Reiter, A.Kukushkin, P. Krstic, P. Strand, L.-G. Eriksson, Contributors to the EFDA TF EU-IM
- Atomic, Molecular, Surface and Nuclear (AMSN) data for the EU-IM-TF (pdf), presented by D.P. Coster (IMP3 Leader) at the ADAS workshop, based on the talk given by Lars-Goran Eriksson at the EU-IM General Meeting, 2008-09
- EU-IM AMNS Interface (pdf), by D.P. Coster

Some papers:

- **Simulations of the edge plasma: the role of atomic, molecular and surface** physics (pdf), by D.P. Coster, X. Bonnin, D. Reiter, A. Kukushkin, S. Gori, P. Krstic, P. Strand, L.-G. Eriksson and Contributors to the EFDA-TF-EU-IM

The present coding for the AMNS project is done in the gforge amnsproto project.

12.5.1 AMNS User Interface

This section discusses the user interface to the AMNS subsystem.

The AMNS library is made available via a module - available versions can be found by executing

```
module avail amns
```

The include and library locations are specified via the "pkg-config" system. To display the available package names do

```
pkg-config --list-all | grep amns
```

Doxygen information about the user interface can be found here.

The AMNS library can be called from

1. Fortran
2. C
3. Python

4. Java (in development)
5. Matlab (in development)

The various bindings for the different languages are given below, but make use of a set of standard concepts which are described first.

12.5.1.1 AMNS User Interface Data Structures

A number of data structures are used by the library interface. Some are opaque (i.e. the contents are not of relevance to the user), and some need to be set or read by the user programme.

The two opaque types are handles which are returned by the setup routines and then need to be passed to the other routines:

1. amns_handle_type, used for the database wide routines
2. amns_handle_rx_type, used for the reaction specific routines

In some language bindings these are the basis of classes.

The non-opaque types are:

1. amns_error_type, used to indicate if an error occured and, if so, what the error was
2. amns_reaction_type, used to indicate the requested reaction
3. amns_set_type, used to set an AMNS internal parameter
4. amns_query_type, used to query an AMNS internal parameter
5. amns_answer_type, used to contain the answer from an AMNS query
6. amns_version_type, used to specify the AMNS version
7. amns_reactants_type, used to specify the reactants to a reaction
8. amns_reactant_type, a sub-component of amns_reactants_type used to characterize the individual reactants

The definitions of these data types can be found at the doxygen documentation for the AMNS User routines

12.5.1.2 AMNS User Interface Data Reactions

The currently available reactions specified in reaction_typex%string in the call to EUIM_AMNS_SETUP_TABLE are

1. RC: Recombination (acd)
2. EI: Electron Impact Ionisation (scd)
3. CX: CX recombination coeffts (ccd)
4. BR: Recomb/brems power coeffts (prb)
5. LR: Line radiation (plt)
6. ZE: Effective Charge (zcd)
7. ZE2: Effective Square Charge (ycd)

8. EIP: Effective Ionisation Potential (ecd)
9. some nuclear reactions
10. ...

The actual reactions are listed in the AMNS section.

12.5.1.3 AMNS User Interface Data Queries

The currently available queries for query%string in the call to EU-IM_AMNS_QUERY is

1. version: Return the version information

The currently available queries for query%string in the call to EU-IM_AMNS_QUERY_TABLE are

1. source: source (origin) of the data
2. no_of_reactants: number of tractants involved
3. index: Not sure what this is
4. filled: whether the data table has been filled ("Filled" or "Empty")
5. reaction_type: reaction type
6. reactants: nuclear charges of reactants
7. version: information about the version
8. state_label: label for the charge state (if appropriate)
9. result_unit: units of the result
10. result_label: description of the result

12.5.1.4 AMNS User Interface Data Setting Options

The currently setting options for set%string in the call to EU-IM_AMNS_SET is

1. NONE

The currently available setting options for set%string in the call to EU-IM_AMNS_SET_TABLE is

1. nowarn: deactivate warning when extrapolating

12.5.1.5 FORTRAN AMNS User Interface

The fortran interface to the AMNS subsystem is based on a standardised set of calls to the AMNS library. The details of what lies behind these calls is the responsibility of the AMNS data providers and does not need to be understood by the users of the AMNS data.

The code modules devloped for the AMNS project are hosted in gforge as the project amnsproto.

12.5.1.5.1 AMNS User Interface: Fortran Calls

The 9 calls to the AMNS system are:

1. EU-IM_AMNS_SETUP, initialization call for the AMNS package

```
subroutine EU-IM_AMNS_SETUP(handle, version, error_status)
  optional version, error_status
  type(amns_handle_type), intent(out) :: handle
  type(amns_version_type), intent(in) :: version
  type(amns_error_type), intent(out) :: error_status
```

2. EU-IM_AMNS_QUERY, query routine for the AMNS package

```
subroutine EU-IM_AMNS_QUERY(handle,query,answer,error_status)
  optional error_status
  type(amns_handle_type), intent(in) :: handle
  type(amns_query_type), intent(in) :: query
  type(amns_answer_type), intent(out) :: answer
  type(amns_error_type), intent(out) :: error_status
```

3. EU-IM_AMNS_SET, set a parameter for the AMNS package

```
subroutine EU-IM_AMNS_SET(handle,set,error_status)
  optional error_status
  type(amns_handle_type), intent(in) :: handle
  type(amns_set_type), intent(in) :: set
  type(amns_error_type), intent(out) :: error_status
```

4. EU-IM_AMNS_FINISH, finalization call for the AMNS package

```
subroutine EU-IM_AMNS_FINISH(handle, error_status)
  optional error_status
  type(amns_handle_type), intent(inout) :: handle
  type(amns_error_type), intent(out) :: error_status
```

5. EU-IM_AMNS_SETUP_TABLE, initialization call for a particular reaction

```
subroutine EU-IM_AMNS_SETUP_TABLE(handle, reaction_type, reactant, handle_rx, error_status)
  optional error_status
  type(amns_handle_type), intent(in) :: handle
  type(amns_reaction_type), intent(in) :: reaction_type
  type(amns_reactants_type), intent(in) :: reactant
  type(amns_handle_rx_type), intent(out) :: handle_rx
  type(amns_error_type), intent(out) :: error_status
```

6. EU-IM_AMNS_QUERY_TABLE, query routine for a particular reaction

```
qsubroutine EU-IM_AMNS_QUERY_TABLE(handle_rx,query,answer,error_status)
optional error_status type(amns_handle_rx_type), intent(in) ::
handle_rx type(amns_query_type), intent(in) :: query
type(amns_answer_type), intent(out) :: answer
type(amns_error_type), intent(out) :: error_status
```

7. EU-IM_AMNS_SET_TABLE, set a parameter for a particular reaction

```
subroutine EU-IM_AMNS_SET_TABLE(handle_rx,set,error_status)
  optional error_status
  type(amns_handle_rx_type), intent(in) :: handle_rx
  type(amns_set_type), intent(in) :: set
  type(amns_error_type), intent(out) :: error_status
```

8. EU-IM_AMNS_FINISH_TABLE, finalization call for a particular reaction

```
subroutine EU-IM_AMNS_FINISH_TABLE(handle_rx, error_status)
  optional error_status
  type(amns_handle_rx_type), intent(inout) :: handle_rx
  type(amns_error_type), intent(out) :: error_status
```

9. EU-IM_AMNS_RX, get the rates associated with the input args for a particular reaction

```
interface EU-IM_AMNS_RX
  module procedure EU-IM_AMNS_RX_1, EU-IM_AMNS_RX_2, EU-IM_AMNS_RX_3
end interface

subroutine EU-IM_AMNS_RX_1(handle_rx,out,arg1,arg2,arg3,error_status)
  optional arg2,arg3,error_status
  type(amns_handle_rx_type), intent(inout) :: handle_rx
  real (kind=R8), intent(out) :: out(:)
  real (kind=R8), intent(in) :: arg1(:),arg2(:),arg3(:)
  type(amns_error_type), intent(out) :: error_status

subroutine EU-IM_AMNS_RX_2(handle_rx,out,arg1,arg2,arg3,error_status)
  optional arg2,arg3,error_status
  type(amns_handle_rx_type), intent(inout) :: handle_rx
  real (kind=R8), intent(out) :: out(:,:)
  real (kind=R8), intent(in) :: arg1(:,:),arg2(:,:),arg3(:,:)
  type(amns_error_type), intent(out) :: error_status

subroutine EU-IM_AMNS_RX_3(handle_rx,out,arg1,arg2,arg3,error_status)
  optional arg2,arg3,error_status
  type(amns_handle_rx_type), intent(inout) :: handle_rx
  real (kind=R8), intent(out) :: out(:,:,:)
  real (kind=R8), intent(in) :: arg1(:,:,:),arg2(:,:,:),arg3(:,:,:)
  type(amns_error_type), intent(out) :: error_status
```

12.5.1.5.2 AMNS User Interface Example (Fortran)

An example of the use of the code can be found in the (fortran minimal example):

```
program minimal
  use itm_types
  use amns_types
  use amns_module

  implicit none

  type (amns_handle_type) :: amns                              ! AMNS global handle
  type (amns_handle_rx_type)  :: amns_rx                       ! AMNS table handle
  type (amns_reaction_type) :: xx_rx
  type (amns_reactants_type) :: species
  real (kind=R8) :: te=100.0_R8, ne=1e20_R8, rate

  call EU-IM_AMNS_SETUP(amns)                                    ! set up the AMNS system
  allocate(species%components(4))                              ! set up reactants
  species%components = (/ amns_reactant_type(6, 1, 12, 0), &
                          amns_reactant_type(1, 0, 2, 0), &
                          amns_reactant_type(6, 0, 12, 1), &
                          amns_reactant_type(1, 1, 2, 1) /)
  xx_rx%string='CX'                                            ! set up reaction
  call EU-IM_AMNS_SETUP_TABLE(amns, xx_rx, species, amns_rx)     ! set up table
  call EU-IM_AMNS_RX(amns_rx, rate, te, ne)                      ! get results
  write(*,*) 'Rate = ', rate
  call EU-IM_AMNS_FINISH_TABLE(amns_rx)                          ! finish with table
  call EU-IM_AMNS_FINISH(amns)                                   ! finish with amns

end program minimal
```

12.5.1.5.3 AMNS User Interface Example Fortran Makefile

An example Makefile demonstrating the use of the AMNS routines:

```
obj/minimal: src/minimal.f90
        ifort -g -o $@ $< ${shell eval-pkg-config --cflags --libs \
      amns-amd64_intel_12 itmtypes-amd64_intel_12 ual-amd64_intel_12}
```

Other examples can be found (here):

12.5.2 C AMNS User Interface

The C interface to the AMNS subsystem is based on a standardised set of calls to the AMNS library. The details of what lies behind these calls is the responsibility of the AMNS data providers and does not need to be understood by the users of the AMNS data.

The code modules devloped for the AMNS project are hosted in gforge as the project amnsproto.

12.5.2.1 AMNS User Interface: C Calls

The 9 calls to the AMNS system are:

1. EU-IM_AMNS_SETUP, initialization call for the AMNS package

```
void EU-IM_AMNS_C_SETUP(void **handle_out, amns_error_type *error_status);
```

2. EU-IM_AMNS_QUERY, query routine for the AMNS package

```
void EU-IM_AMNS_C_QUERY(void *handle_in, amns_query_type *query,
                        amns_answer_type *answer, amns_error_type *error_status)
```

3. EU-IM_AMNS_SET, set a parameter for the AMNS package

```
void EU-IM_AMNS_C_SET(void *handle_in, amns_set_type *set, amns_error_type *error_status);
```

4. EU-IM_AMNS_FINISH, finalization call for the AMNS package

```
void EU-IM_AMNS_C_FINISH(void **handle_inout, amns_error_type *error_status);
```

5. EU-IM_AMNS_SETUP_TABLE, initialization call for a particular reaction

```
void EU-IM_AMNS_C_SETUP_TABLE(void *handle_in, amns_reaction_type *reaction_type,
                              void *reactant_handle_in, void **handle_rx_out,
                              amns_error_type *error_status);
```

6. EU-IM_AMNS_QUERY_TABLE, query routine for a particular reaction

```
void EU-IM_AMNS_C_QUERY_TABLE(void *handle_rx_in, amns_query_type *query,
                              amns_answer_type *answer, amns_error_type *error_status);
```

7. EU-IM_AMNS_SET_TABLE, set a parameter for a particular reaction

```
void EU-IM_AMNS_C_SET_TABLE(void *handle_rx_in, amns_set_type *set,
                            amns_error_type *error_status);
```

8. EU-IM_AMNS_FINISH_TABLE, finalization call for a particular reaction

```
void EU-IM_AMNS_C_FINISH_TABLE(void **handle_rx_inout, amns_error_type *error_status);
```

9. EU-IM_AMNS_RX, get the rates associated with the input args for a particular reaction

```
void EU-IM_AMNS_C_RX_0_A(void *handle_rx_in, double *out,
                         double arg1, amns_error_type *error_status);
void EU-IM_AMNS_C_RX_0_B(void *handle_rx_in, double *out,
                         double arg1, double arg2, amns_error_type *error_status);
void EU-IM_AMNS_C_RX_0_C(void *handle_rx_in, double *out,
                         double arg1, double arg2, double arg3, amns_error_type *error_s
tatus);

void EU-IM_AMNS_C_RX_1_A(void *handle_rx_in, int nx, double *out,
                         double *arg1, amns_error_type *error_status);
void EU-IM_AMNS_C_RX_1_B(void *handle_rx_in, int nx, double *out,
                         double *arg1, double *arg2, amns_error_type *error_status);
void EU-IM_AMNS_C_RX_1_C(void *handle_rx_in, int nx, double *out,
                         double *arg1, double *arg2, double *arg3, amns_error_ty
pe *error_status);

void EU-IM_AMNS_C_RX_2_A(void *handle_rx_in, int nx, int ny,
                         double *out, double *arg1, amns_error_type *error_status);
void EU-IM_AMNS_C_RX_2_B(void *handle_rx_in, int nx, int ny,
                         double *out, double *arg1, double *arg2, amns_error_type *error_
 ↪status);
void EU-IM_AMNS_C_RX_2_C(void *handle_rx_in, int nx, int ny,
                         double *out, double *arg1, double *arg2, double *arg3, amns_error_type␣
 ↪*error_status);

void EU-IM_AMNS_C_RX_3_A(void *handle_rx_in, int nx, int ny, int nz,
                         double *out, double *arg1, amns_error_type *error_status);
void EU-IM_AMNS_C_RX_3_B(void *handle_rx_in, int nx, int ny, int nz,
                         double *out, double *arg1, double *arg2, amns_error_type *error_
 ↪status);
void EU-IM_AMNS_C_RX_3_C(void *handle_rx_in, int nx, int ny, int nz,
                         double *out, double *arg1, double *arg2, double *arg3, amns_error_type␣
 ↪*error_status);
```

In addition, service routines are provided for dealing with reactants:

```
void EU-IM_AMNS_C_SETUP_REACTANTS(void **reactants_handle_out, char string_in[reaction_
 ↪length],
                                  int index_in, int n_react
ants);
void EU-IM_AMNS_C_SET_REACTANT(void *reactants_handle_in, int reactant_index, amns_reactant_
 ↪type *reactant_in);
void EU-IM_AMNS_C_GET_REACTANT(void *reactants_handle_in, int reactant_index, amns_reactant_
 ↪type *reactant_out);
void EU-IM_AMNS_C_FINISH_REACTANTS(void **reactants_handle_inout);
```

12.5.2.2 AMNS User Interface Example (C)

An example of the use of the code can be found in the (c minimal example):

```
#include "amns_interface.h"

int main(int argc, char *argv[])
{
  void* amns_handle = NULL;
  amns_c_error_type error_stat = DEFAULT_AMNS_C_ERROR_TYPE;
  void* reactants_handle = NULL;
  amns_c_reactant_type species1  = {.ZN=6, .ZA=1, .MI=12, .LR=0};
  amns_c_reactant_type species2  = {.ZN=1, .ZA=0, .MI=2 , .LR=0};
  amns_c_reactant_type species3  = {.ZN=6, .ZA=0, .MI=12, .LR=1};
  amns_c_reactant_type species4  = {.ZN=1, .ZA=1, .MI=2 , .LR=1};
  amns_c_reaction_type xx_rx  = {.string = "CX"};
```

(continues on next page)

(continued from previous page)

```
  void* amns_cx_handle;
  double rate;

  EU-IM_AMNS_CC_SETUP(AMNS_HANDLE, &ERROR_STAT)
  printf("error = %s: %s\n", error_stat.flag ? "true" : "false", error_stat.string);
  EU-IM_AMNS_CC_SETUP_REACTANTS(REACTANTS_HANDLE, "", 0, 4)
  EU-IM_AMNS_CC_SET_REACTANT(reactants_handle, 1, SPECIES1)
  EU-IM_AMNS_CC_SET_REACTANT(reactants_handle, 2, SPECIES2)
  EU-IM_AMNS_CC_SET_REACTANT(reactants_handle, 3, SPECIES3)
  EU-IM_AMNS_CC_SET_REACTANT(reactants_handle, 4, SPECIES4)
  EU-IM_AMNS_CC_SETUP_TABLE(amns_handle, XX_RX, REACTANTS_HANDLE, &AMNS_CX_HANDLE, &ERROR_
↪STAT)
  printf("error = %s: %s\n", error_stat.flag ? "true" : "false", error_stat.string);
  EU-IM_AMNS_CC_RX_0_B(amns_cx_handle, RATE, 100.0, 1E20, &ERROR_STAT)
  printf("error = %s: %s\n", error_stat.flag ? "true" : "false", error_stat.string);
  printf("rate=%e\n", rate);
  EU-IM_AMNS_CC_FINISH_TABLE(AMNS_CX_HANDLE, &ERROR_STAT)
  printf("error = %s: %s\n", error_stat.flag ? "true" : "false", error_stat.string);
  EU-IM_AMNS_CC_FINISH_REACTANTS(REACTANTS_HANDLE)
  EU-IM_AMNS_CC_FINISH(AMNS_HANDLE, &ERROR_STAT)
  printf("error = %s: %s\n", error_stat.flag ? "true" : "false", error_stat.string);
  return 0;
}
```

12.5.2.3 AMNS User Interface Example C Makefile

An example Makefile demonstrating the use of the AMNS routines:

```
obj/minimal: src/minimal.c
        gcc -g -o $@ $< ${shell eval-pkg-config --cflags --libs\
          amns-ifort itmconstants ual-amd64_intel_12}
```

Other examples can be found (here):

12.5.3 Python AMNS User Interface

The Python interface to the AMNS subsystem is based on a standardised set of calls to the AMNS library. The details of what lies behind these calls is the responsibility of the AMNS data providers and does not need to be understood by the users of the AMNS data.

The code modules devloped for the AMNS project are hosted in gforge as the project amnsproto.

12.5.3.1 AMNS User Interface: Python Calls

The Python interface creates

1. Amns (class)
    1. finalize (method)
    2. get_table (method)
    3. query (method)
    4. set (method)
2. Table (class)
    1. data (method)

2. finalize (method)
3. query (method)
4. set (method)

3. Reactants (class)
    1. add (method)
    2. test (method)
    3. value (method)

12.5.3.2 AMNS User Interface Example (Python)

An example of the use of the code can be found in the (python minimal example):

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-
import amns
import numpy as np

amnsdb = amns.Amns()
r = amns.Reactants()
r.add(6,1,12)
r.add(1,0,2)
r.add(6,0,12,lr=1)
r.add(1,1,2,lr=1)
table = amnsdb.get_table("CX", r)
print "table.no_of_reactants", table.no_of_reactants
print table.data(np.array([100.0]), np.array([1e20]))
amnsdb.finalize()
```

Other examples can be found (here):

12.5.4 AMNS CPO

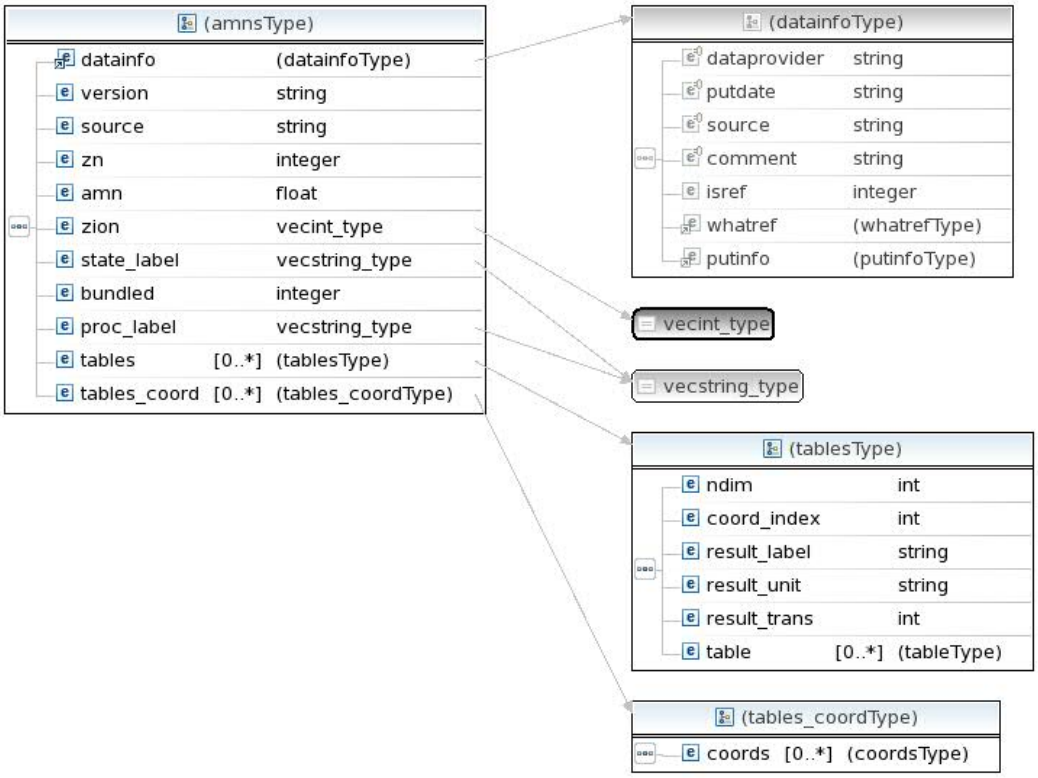
The current (4.08b) data structure for AMNS data in the standard tree view can be browsed here (Browse)

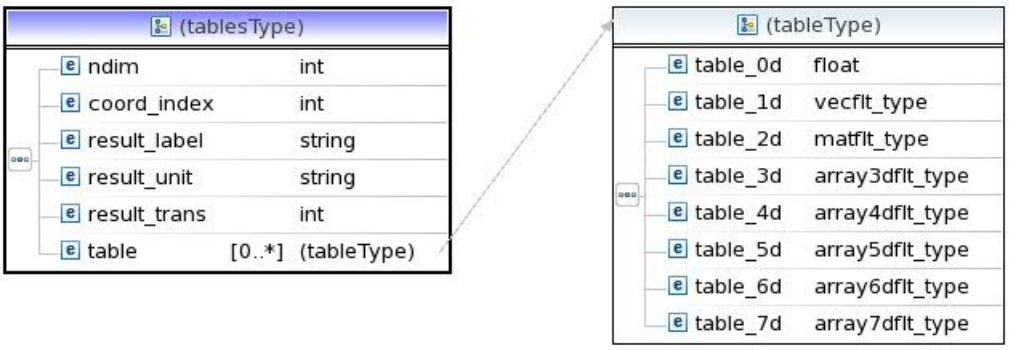
We are currently considering a revision of the AMNS data structure that makes use of arrays-of-structures (not available earlier)

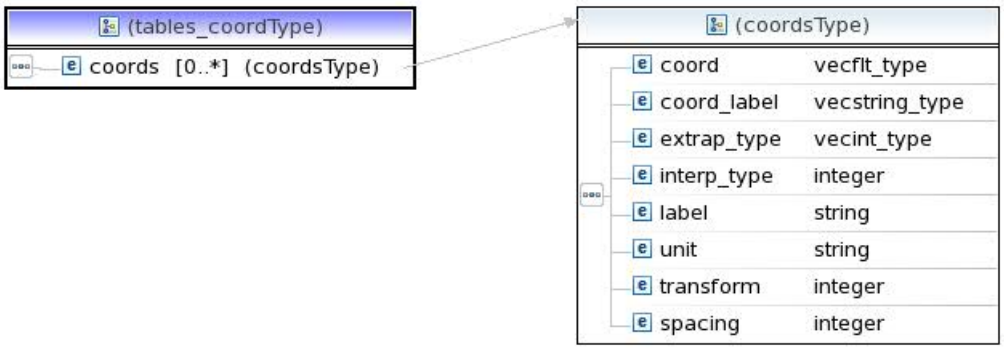
At the top level we would have

with the definition of tables

and the tables of coordinates


(amnsType)
datainfo (datainfoType)
version string
source string
zn integer
amn float
zion vecint_type
state_label vecstring_type
bundled integer
proc_label vecstring_type
tables [0..*] (tablesType)
tables_coord [0..*] (tables_coordType)
(datainfoType)
dataprovider string
putdate string
source string
comment string
isref integer
whatref (whatrefType)
putinfo (putinfoType)
vecint_type
vecstring_type
(tablesType)
ndim int
coord_index int
result_label string
result_unit string
result_trans int
table [0..*] (tableType)
(tables_coordType)
coords [0..*] (coordsType)


(tablesType)
ndim int
coord_index int
result_label string
result_unit string
result_trans int
table [0..*] (tableType)
(tableType)
table_0d float
table_1d vecflt_type
table_2d matflt_type
table_3d array3dflt_type
table_4d array4dflt_type
table_5d array5dflt_type
table_6d array6dflt_type
table_7d array7dflt_type


(tables_coordType)
coords [0..*] (coordsType)
(coordsType)
coord vecflt_type
coord_label vecstring_type
extrap_type vecint_type
interp_type integer
label string
unit string
transform integer
spacing integer

CHAPTER

THIRTEEN

USING THE WPCD WORKFLOWS

Use of the WPCD workflows is available via the EUROfusion Gateway. In order to obtain an account on the Gateway, follow the instruction at https://wiki.eufus.eu/doku.php#how_to_get_a_user_account

13.1 Indices and tables

- genindex
- modindex
- search