ISIP

December 17, 2020

Contents

1 Scope

ISIP (Infrastructure and Software Integration Project) is in charge of developing and maintaining the ITM-TF simulation infrastructure.

2 Project Timeline

The ISIP Timeline can be found here [1].

3 Contacts

Project Leader : frederic.imbeaux@cea.fr (CEA)

Deputies : gabriele.manduchi@igi.cnr.it (Consorzio RFX), hmk@ipp.mpg.de (IPP Garching)

4 News

- 22/03/2012: Data version 4.10a has been released
- 16/03/2012: Data version 4.09b has been released
- 20/05/2011: Data version 4.09a has been released
- 20/10/2010: ISE has been updated for 4.08a and 4.08b and is now fully usable
- 23/09/2010: Data version 4.08b has been released

5 User Support

Questions/problems and Feature Requests related to the ISIP tools can be posted to the General Support [2]

6 Data Structure

In a workflow, physics modules exchange physics data in the form of standardised blocks of information : the Consistent Physical Objects (CPOs). The list of CPOs as well as their inner structure defines the ITM Data Structure. All physics modules should use these standardised interfaces for I/O.

An introduction to the ITM data structure is given in the presentation Data Structures in Practice [3] by Frédéric Imbeaux.

6.1 Data Structure Releases (users)

The whole ITM platform is released by versions. To each version corresponds a definition of the Data Structure which can be found below. See the History [4] for the history / description of all releases.

Data structure 4.10b.10
Released as a test version Data structure 4.10b.10 (Browse) [5]

Type definitions for Fortran can be found here Fortran [6]

List of changes:
AMNS CPO: a few nodes have become vectors instead of scalar
ANTENNAS CPO: two nodes have been added
COREPROF CPO: a few nodes have been added
NEOCLASSIC CPO: a few nodes have been added
EQUILIBRIUM CPO: clarified definition of geom_axis
COREDELTA CPO: correction of the impurity nodes
DISTRIBUTION CPO: a few nodes have been added and a wrong type definition has been corrected

[1] https://www.efda-itm.eu/ITM/imports/isip/public/isip_timeline.pdf
[2] https://gforge6.eufus.eu/project/generalsupport
[3] https://www.efda-itm.eu/ITM/imports/isip/public/isip_ITMDataStructures-1.pdf
[4] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/isip_Phase4Versions.pdf
[5] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.10b.10/Phase4TOP.html
[6] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.10b.10/euitm_schemas.f90

Data structure 4.10b.8
Production release for 4.10b, dated August 2014
Data structure 4.10b.8 (Browse) [7](Download) [8].

Type definitions for Fortran can be found here Fortran [9]

Data structure 4.10b.3
TEST release date: 12 May 2014, this is the first version of datastructure 4.10b released for TESTING purposes. The default production version is still 4.10a.3 (see below)
Data structure 4.10b.3 (Browse) [10](Download) [11].

Data structure 4.10a.3
release date: February 2013, this is the slightly updated 4.10a version under exploitation on the new Gateway in Garching
Data structure 4.10a.3 (Browse) [12](Download) [13].

Data structure 4.10a
release date: 15/05/2012, significant updates of the datastructure + full integration of memory caching in Kepler/UAL. Last datastructure release on the Portici Gateway
Data structure 4.10a (Browse) [14](Download) [15].

Type definitions:

- Fortran [16]
- C++ [17]
- Java: The class definitions for Java can be found in `$SWITMDIR/ual/4.10a/javainterface/ualmemory/javainterfac`
- Python: The class definitions for Python can be found in `$SWITMDIR/ual/4.10a/pythoninterface`
- Matlab: The class definitions for Matlab can be found in `$SWITMDIR/ual/4.10a/matlabinterface`

Data structure 4.09b
release date: 16/03/2012, this is a transition version enabling UAL memory cache with the same datastructure as 4.09a
4.09b: This version has the same datastructure as 4.09a and features in-memory transfer for the UAL (available only for the default JNI execution mode). In-memory data transfer implied changes in UALinit and UALcollector actors.
Full Kepler release can be found at: $SWITMDIR/kepler/4.09b
If you prefer to start from your Kepler 4.09a and only update the UALinit and UALcollector actors, do this by running the script: $SWITMDIR/kepler/4.09b/script.sh (do this after ITMv1 specifying the location of your original 4.09a Kepler). Note that this transformation is not backward compatible.

Data structure 4.09a
release date: 20/05/2011

[7] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.10b.8/Phase4top.html
[8] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.10b.8/Phase4.10b.8_HTML.zip
[9] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.10b.8/euitm_schemas.f90
[10] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.10b.3/Phase4TOP.html
[11] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.10b.3/Phase4.10b.3_HTML.tar
[12] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.10a.3/Phase4top.html
[13] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.10a.3/Phase4.10a.3_HTML.zip
[14] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.10a/Phase4top.html
[15] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.10a/Phase4.10a_HTML.zip
[16] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.10a/euitm_schemas.f90
[17] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.10a/UALClasses.h

Data structure 4.09a (Browse) [18](Download) [19].

Type definitions:

- Fortran [20]
- C++ [21]
- Java: The class definitions for Java can be found in `$SWITMDIR/ual/4.09a/javainterface/ualmemory/javainterfac`
- Python: The class definitions for Python can be found in `$SWITMDIR/ual/4.09a/pythoninterface`
- Matlab: The class definitions for Matlab can be found in `$SWITMDIR/ual/4.09a/matlabinterface`

Data structure 4.08b
release date: 23/09/2010
Data structure 4.08b (Browse) [22](Download) [23].

Type definitions:

- Fortran [24]
- C++ [25]
- Java: The class definitions for Java can be found in `$SWITMDIR/ual/4.08b/javainterface/ualmemory/javainterfac`
- Python: The class definitions for Python can be found in `$SWITMDIR/ual/4.08b/pythoninterface`
- Matlab: The class definitions for Matlab can be found in `$SWITMDIR/ual/4.08b/matlabinterface`

Data structure 4.08a
release date: 02/04/2010
Data structure 4.08a (Browse) [26](Download) [27].

Type definitions:

- Fortran [28]
- C++ [29]
- Java: The class definitions for Java can be found in `$SWITMDIR/ual/4.08a/javainterface/ualmemory/javainterfac`
- Python: The class definitions for Python can be found in `$SWITMDIR/ual/4.08a/pythoninterface`
- Matlab: The class definitions for Matlab can be found in `$SWITMDIR/ual/4.08a/matlabinterface`

Data structure 4.07c
release date: 02/04/2010

Data structure 4.07c is exactly the same as 4.07b but this tag corresponds to a new release of Kepler and associated tools.

[18] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.09a/Phase4top.html
[19] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.09a/Phase4.09a_HTML.zip
[20] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.09a/euitm_schemas.f90
[21] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.09a/UALClasses.h
[22] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.08b/Phase4top.html
[23] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.08b/Phase4.08b_HTML.zip
[24] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.08b/euitm_schemas.f90
[25] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.08b/UALClasses.h
[26] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.08a/Phase4top.html
[27] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.08a/Phase4.08a_HTML.zip
[28] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.08a/euitm_schemas.f90
[29] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.08a/UALClasses.h

Data structure 4.07b
release date: 14/09/2009
Data structure 4.07b (Browse) [30](Download) [31].

Type definitions:

- Fortran [32]
- C++ [33]
- Java: The class definitions for Java can be found in `$SWITMDIR/ual/4.07b/javainterface/ualmemory/javainterfac`
- Python: The class definitions for Python can be found in `$SWITMDIR/ual/4.07b/pythoninterface`
- Matlab: The class definitions for Matlab can be found in `$SWITMDIR/ual/4.07b/matlabinterface`

Data structure 4.07a
release date: 04/05/2009
Data structure 4.07a (Browse) [34](Download) [35].

Contains many revisions to IMP3 data structure, new CPOs [36] from IMP2 and IMP5.

Type definitions:

- Fortran [37]
- C++ [38]
- Java: The class definitions for Java can be found in `$SWITMDIR/ual/4.07a/javainterface/ualmemory/javainterfac`
- Python: The class definitions for Python can be found in `$SWITMDIR/ual/4.07a/pythoninterface`
- Matlab: The class definitions for Matlab can be found in `$SWITMDIR/ual/4.07a/matlabinterface`

Data structure 4.06d
release date: 19/09/2008
Data structure 4.06d (Browse) [39](Download) [40].

Contains the core transport + equilibrium data structure.

Type definitions:

- Fortran [41]
- C++ [42]
- Java: The class definitions for Java can be found in `$SWITMDIR/ual/4.06d/javainterface/ualmemory/javainterfac`
- Python: The class definitions for Python can be found in `$SWITMDIR/ual/4.06d/pythoninterface`
- Matlab: The class definitions for Matlab can be found in `$SWITMDIR/ual/4.06d/matlabinterface`

[30] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.07b/Phase4top.html
[31] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.07b/Phase4.07b_HTML.zip
[32] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.07b/euitm_schemas.f90
[33] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.07b/UALClasses.h
[34] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.07a/Phase4top.html
[35] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.07a/Phase4.07a_HTML.zip
[36] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_cpo
[37] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.07a/euitm_schemas.f90
[38] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.07a/UALClasses.h
[39] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.06d/Phase4top.html
[40] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.06d/Phase4.06d_HTML.zip
[41] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.06d/euitm_schemas.f90
[42] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/4.06d/UALClasses.h

6.2 Data structure XML schemas (experts)

The ITM datastructure is coded as xml schemas. This unique source is used to derive all ITM applications related to CPOs : UAL, CPO documentation, ... The data structure XML schemas are stored in a subversion repository in /afs/efda-itm.eu/isip/project/portal/gforge/storage/svnroot/datastructure.

To export version 4.08b from the repository, storing it in subdirectory *xml* , do

```
svn export https://gforge6.eufus.eu/svn/datastructure/tags/4.08b xml
```

To check out a subversion working copy of the entire repository, storing it in subdirectory *datastructure* , do

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

The instructions for writing the ITM datastructure XML schemas can be found here. [43]

---

last update: 2019-01-31 by g2dpc

7 Databases

ITM data entries are defined by the following information : (user, machine, shot, run). "user" is either "public" (public ITM database) or any Gateway [44] username, allowing the creation of private ITM databases.

In order to run a KEPLER workflow, you must first create your private database(s). A private database is created for a given machine (tokamak name) and data structure version, by the following command:

```
$ITMSCRIPTDIR/create_user_itm_dir TokamakName \
 DataVersion
```

Example (creates a tree for tokamak name test, allowed for testing purposes):

```
$ITMSCRIPTDIR/create_user_itm_dir test 4.08a
```

The database is created under ~my_username/public/itmdb/itm_trees. Since it is located in your public directory, all Gateway users can read from it.

Standard tokamak names are : asdex_upgrade, jet, mast, tore_supra, ... Nonetheless, any arbitrary machine name is allowed by the system for testing purposes (e.g. test). It is however strongly recommended to use the standard machine names when using real experimental data.

Before using KEPLER or the UAL [45] , you must specify on which database you wish to work, using the following command :

```
source $ITMSCRIPTDIR/set_itm_data_env public|user TokamakName DataVersion
```

Example (set the environment variables to work to the previously created directory):

```
source $ITMSCRIPTDIR/set_itm_data_env my_username test 4.08a
```

Note that with this command, you can read any ITM database on the Gateway, including the public database (replace my_username by "public").

The public database is located at /pfs/itmdb/itm_trees/public/. Its content is summarised in ITM Public Database [46], which contains also the location of the various files, the list of standard machine names ...

[43] https://www.efda-itm.eu/ITM/imports/isip/public/data_structure/isip_InstructionsSchemas.pdf
[44] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_gateway
[45] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_ual
[46] https://www.efda-itm.eu/ITM/imports/isip/public/isip_PublicContent.pdf

For the moment, the ITM software (KEPLER, UAL) are able to work only on one database at a time. Reading is allowed from all databases (by using the set_itm_data_env command above). However, if you wish to write also some results during the workflow, you can write only to your private database. Therefore, if you wish to use data entries from other users in your workflow, you need first to copy them in your own directory.

For example, to copy the test example from the public database (user=public, machine=test, shot=1, run=1) to your directory, type:

```
cp /pfs/itmdb/itm_trees/public/test/4.08a/mdsplus/0/euitm_10001.* \
   ~my_username/public/itmdb/itm_trees/test/4.08a/mdsplus/0/.
```

This data entry can then be used as (user=my_username, machine=test, shot=1, run=1). It is not recommended to change the shot or run number when copying data entries like this. More flexible tools for working on multiple databases simultaneously will be provided in the near future.

Machine descriptions [47] are stored in shot 0 of each tokamak. Different versions (valid e.g. for different shot ranges) can be stored, using different run numbers. To copy the Tore Supra machine description (run 1) to your local folder, type (NB you must first create your private tore_supra 4.08a database):

```
cp /pfs/itmdb/itm_trees/public/tore_supra/4.08a/mdsplus/0/euitm_001.* \
   ~my_username/public/itmdb/itm_trees/tore_supra/4.08a/mdsplus/0/.
```

To copy the Tore Supra experimental data set for shot 39736, run 1, type:

```
cp /pfs/itmdb/itm_trees/public/tore_supra/4.08a/mdsplus/0/euitm_397360001.* \
   ~my_username/public/itmdb/itm_trees/tore_supra/4.08a/mdsplus/0/.
```

Note for advanced users : an ITM data entry consists in three file, euitm_SSSSSSRRRR.* where SSSSSS is the shot number (truncated, e.g. shot=1 is SSSSSS=1) and RRRR the run number (exception : if shot = 0, SSSSSSR-RRR = (R)RRR, e.g. shot 0 run 1 is euitm_001.*). In the example above, it is assumed that one is using the default storage method MDS+ (otherwise, for HDF5, replace in the path the "mdsplus" folder name by "hdf5"), and that the run number is below 9999 (otherwise, replace the "0" folder name by "1" for run numbers between 10000 and 19999, ...).

7.1 exp2ITM

Experimental Data Overview [48]

---

last update: 2013-09-12 by dpc

8 Universal Access Layer (UAL)

8.1 Introduction

The UAL (Universal Access Layer) [49] is a multi-language library that allows exchanging Consistent Physical Objects (CPOs) between various modules, and to write to an ITM database. The documentation here is provided for rather experienced users who want to practice the UAL in their test programs. Regular KEPLER [50] 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 **ITMv1** 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.

[47] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_machine_description
[48] https://www.efda-itm.eu/ITM/imports/isip/public/isip_ExperimentalDataITM_v3.pdf
[49] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_ual
[50] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_kepler

```
source $ITMSCRIPTDIR/ITMv1 KEPLERFOLDER
MACHINENAME DATAVERSION
```

e.g.:

```
source $ITMSCRIPTDIR/ITMv1 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 ITMv1 script uses the two following scripts: to set the database environment variables (mandatory prior UAL usage), use:

```
source $ITMSCRIPTDIR/set_itm_data_env USERNAME
 MACHINENAME DATAVERSION
```

e.g.:

```
source $ITMSCRIPTDIR/set_itm_data_env myname jet 4.08a
```

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

```
source $ITMSCRIPTDIR/set_itm_env DATAVERSION
```

e.g.:

```
source $ITMSCRIPTDIR/set_itm_env 4.08a
```

UAL libraries are installed in `/afs/efda-itm.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
```

8.2 UAL User Guide

Click on the following link for the UAL User Guide [51]

---

last update: 2019-01-31 by g2dpc

9 FC2K

FC2K [52] is a tool for wrapping a Fortran or C++ source code into a Kepler [53] actor. Before using it, your physics code should be ITM-compliant (i.e. use CPOs [54] as input/output).

After running the **ITMv1 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 [55] subversion repository `fc2k` . To check out a subversion working copy of the repository, storing it in subdirectory *fc2k* , do

[51] https://www.efda-itm.eu/ITM/imports/isip/public/isip_UAL_User_Guide.pdf
[52] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_fc2k
[53] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_kepler
[54] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_cpo
[55] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_gforge

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

Executables etc are in `$SWITMDIR/ihm/fc2k/` .

There are a few tools for managing actors, in `$ITMSCRIPTDIR/` (put it in your $PATH):

- rmactor [56]: remove an actor from your Kepler version ($KEPLER).
- extract_actor [57]: export an actor from $KEPLER.
- import_actor [58]: import an actor into $KEPLER. Then update your Kepler version:

```
cd $KEPLER
ant buildkarlib
```

9.1 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 ITM compliant and how to turn it into a Kepler [59] actor [60] .

9.1.1 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 [61] must be declared with types `ItmNs::Itm::cpo_type` or `ItmNs::Itm::cpo_typeAr`
. 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);
```

[56] https://www.efda-itm.eu/ITM/html/itm_howtos.html#remove_actor
[57] https://www.efda-itm.eu/ITM/html/itm_howtos.html#export_actor
[58] https://www.efda-itm.eu/ITM/html/itm_howtos.html#import_actor
[59] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_kepler
[60] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_actor
[61] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_cpo

9.1.2 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.)

9.1.3 Compile your function as a library

You need to include the header directories for the UAL [62] and Blitz:

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

Same for linking:

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

Additionally, you must compile with the -fPIC option.

9.1.4 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,
      ItmNs::Itm::antennas & ant,
      ItmNs::Itm::equilibriumArray & eq,
      int & x,
      ItmNs::Itm::limiter & lim,
      ItmNs::Itm::coreimpur & cor,
      ItmNs::Itm::ironmodelArray & iron,
      double * y,
      char * str,
param & codeparam)
{

      /* display first line of parameters */
      cout << codeparam.parameters[0] << endl;
```

[62] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_ual

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

Content of *Makefile* :

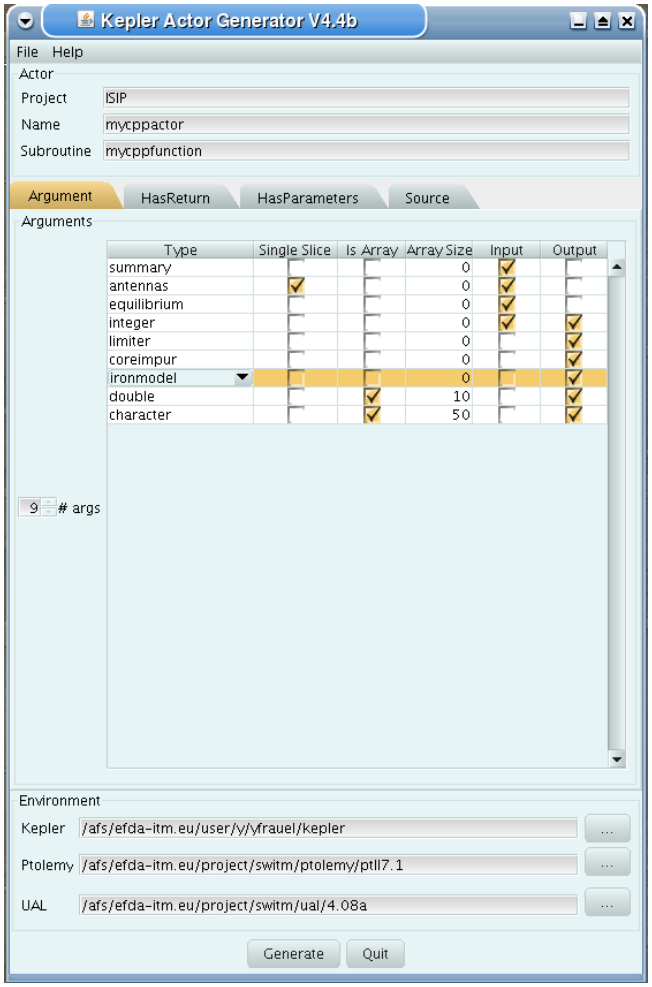
```
CXXFLAGS=-g -fPIC -I$(UAL)/include -I$(UAL)/lowlevel -I$(UAL)/cppinterface/
-I$SWITMDIR/blitz/blitz-0.9/include/
LDFLAGS=-L$(UAL)/lib -lUALCPPInterface -lUALLowLevel -L/afs/efda-
itm.eu/project/switm/blitz/blitz-0.9/lib -lblitz
libmycppfunction.a: mycppfunction.o
      ar -rvs libmycppfunction.a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

9.1.5 How to fill the FC2K window

First tab (Argument):

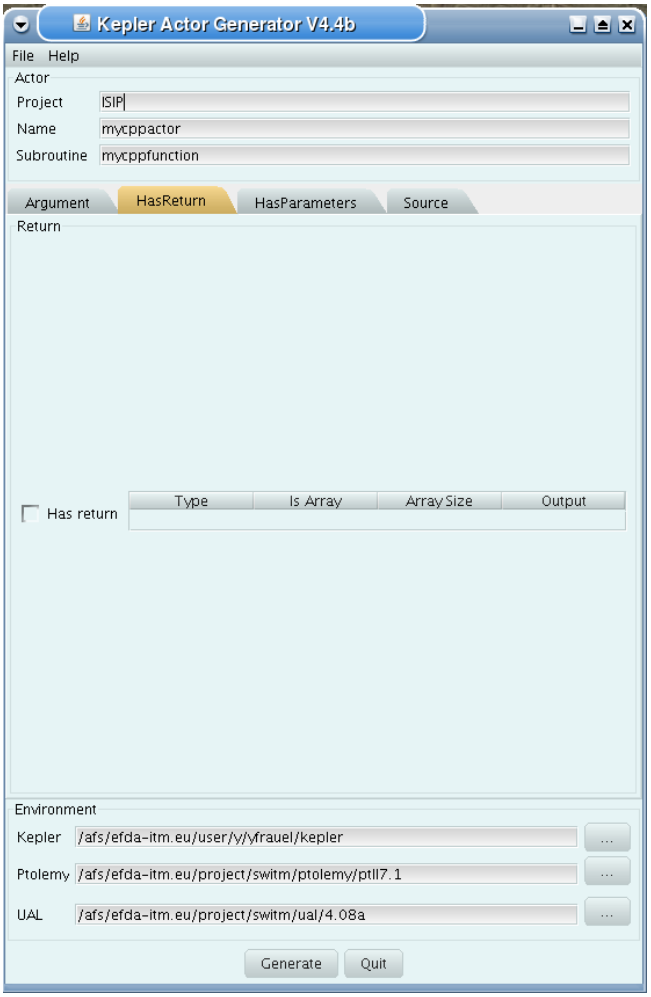
- set number of input and output arguments (combined)
- select type of arguments from drop-down menu
- tick if argument is a single time slice
- tick if argument is array (not for pointers)
- if necessary define size of arrays
- tick if argument is input argument
- tick if argument is output argument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

The fields `Kepler` , `Ptolemy` , and `UAL` are automatically filled with the values which you set by running the **ITMv1 script** .



Second tab (HasReturn):

- specify return parameters (type, array, size)

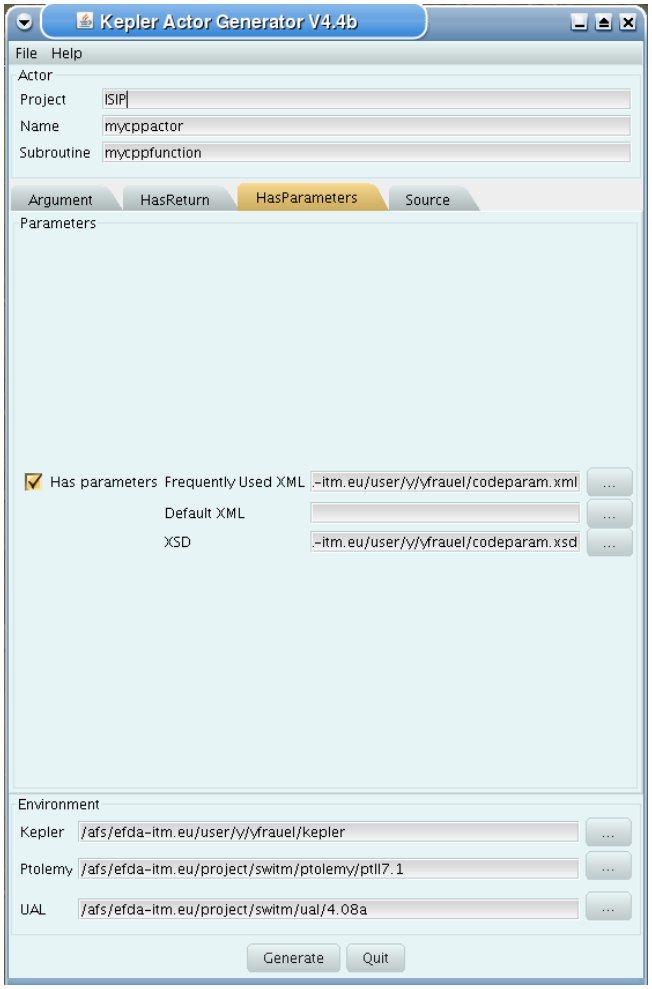


Third tab (HasParameters):

- 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)

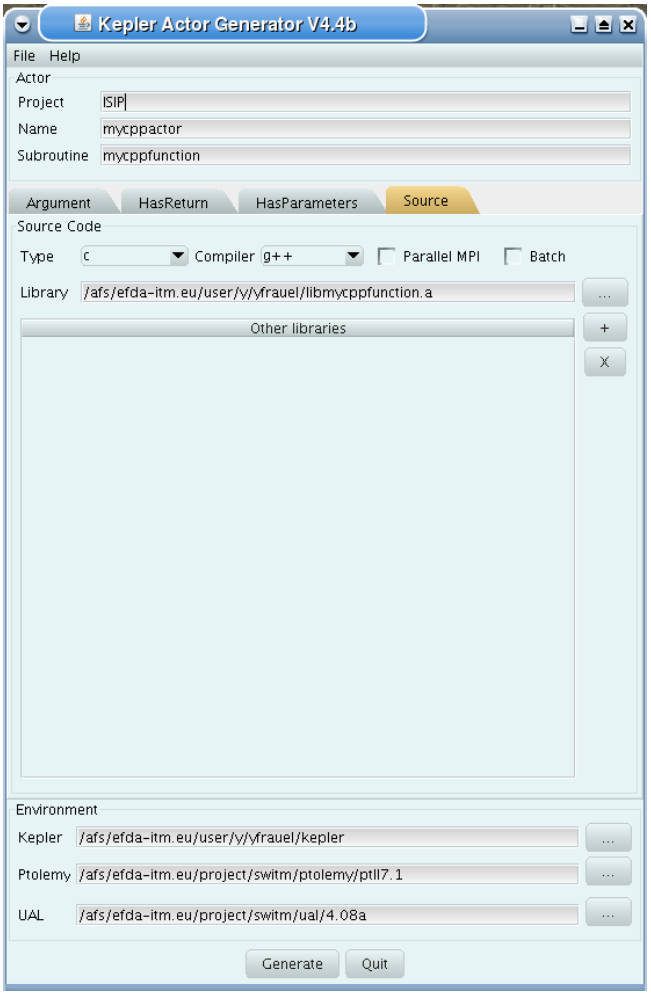
For information on code specific parameters, please see How to handle code specific parameters [63].

[63] https://www.efda-itm.eu/ITM/html/itm_code_parameters.html#itm_code_parameters



Fourth tab (Source):

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



last update: 2013-09-12 by dpc

last update: 2019-01-31 by g2dpc

10 Kepler

10.1 Setup

These are instructions for installing the Kepler [64] release for data version 4.10a

Install a private version of KEPLER for 4.10a on your account:

```
cd ~
rm -rf kepler .kepler
tar xvf $SWITMDIR/kepler/4.10a/kepler.tar
```

Set the Kepler, UAL [65] and database environment variables. Kepler will work in your private database, under the machine TokamakName (TokamakName=test is allowed for testing purposes).

```
source $ITMSCRIPTDIR/ITMv1 KeplerFolder TokamakName DataVersion
```

KeplerFolder should be your private Kepler path relative to your home directory. Simply "kepler" is recommended, unless you want to have several private versions of Kepler in your directory. Example:

[64] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_kepler
[65] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_ual

```
source $ITMSCRIPTDIR/ITMv1 kepler test 4.10a
```

If this is not run on restart, kepler will open in the default environment which is not what you want!

To have your environment set by default (**mandatory for new features such as debugging, use of batch jobs, ...**), it is recommended to set default envinronment variables by adding this command in you .cshrc file as follows:
`source $ITMSCRIPTDIR/ITMv1 KeplerFolder TokamakName DataVersion >/dev/null`

Prepare the input data (see Databases (7) for details). Create a "test" zone in your private ITM data tree if you haven't already done it.

```
$ITMSCRIPTDIR/create_user_itm_dir test 4.10a
```

This version of Kepler is provided with a few workflow examples, located in the kepler/demos directory. To use them, you need to copy the example input data set (shot 4, run 1) from the public database to your test database.

```
cp /pfs/itmdb/itm_trees/public/test/4.10a/mdsplus/0/euitm_40001.* \
   $HOME/public/itmdb/itm_trees/test/4.10a/mdsplus/0
```

Run KEPLER:

```
kepler
```

Select File > Open File and open any of the example workflows in the demos directory. Start for instance with staticexample.xml. Results will be written to your private database.

```
ls -gtr $HOME/public/itmdb/itm_trees/test/4.10a/mdsplus/0
```

10.2 Issues

1. If the kepler environment variables are not set as described above (10.1), kepler will open in the default environment which is probably not what you want. Kepler will probably complain about missing files. Check that $KEPLER points at your private version.
2. If Kepler refuses to run. it may be because you have last run it on a different gateway [66] node. Delete .kepler and try again!
3. If Kepler dies, the core dump can be enormous - 1.6GB seen with an ITM test workflow. To find core files in your home directory tree do

   ```
   find ~/ -name 'core.*'
   ```

   and to check disk quota, do

   ```
   fs listquota
   ```

4. If you have selected the debug mode of an actor (*configure actor*), Kepler will fire up `totalview` when running this actor. The process which starts `totalview` will do so in a new shell. Therefore, any `.cshrc` or `.login` file will be sourced again and environment variables (like `TOKAMAKNAME` or `UAL`) will be reset to the values defined in these files (if you included for instance a source of the ITMv1 script). This

[66] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_gateway

may result in Kepler not being able to access the `runwork` data file anymore since this was stored at the original location in the data base.
If running actors in debug mode, make sure that `TOKAMAKNAME` and `UAL` set in `.cshrc` and `.login` are what you expect them to be!

---

last update: 2013-09-12 by dpc

11 KeplerActors

The **kepleractors** project under GForge [67] is used for exchanging Kepler [68] actors [69] among ITM users.

This project and its associated SVN repository have been created related to the GForge [70] project **Keplerworkflows (16.1.2.1.12)** for enhancing traceability and reproducibility of simulations.

We describe here a complete procedure that is recommended for traceability purpose. The goal is to be able to keep track of how the actor has been generated (Gforge project of the source code, FC2K [71] parameters, ...).

For public releases a unique name shall be assigned to the actor:

All **public actors** should be generated (and thus appear in Kepler) with a name which is the concatenation of
`actor_base_name + actor_public_version` .
`actor_base_name` : this can be chosen arbitrarily, though we suggest that it corresponds to project name under Gforge, i.e. the name of the source code's project under Gforge when there is a one-to-one link between the Gforge project and the actor.
`actor_public_version` : is the number of public release for this particular actor.
`actor_name` : concatenation of actor_base_name and actor_public_version

11.1 Structure of the actor repository

The actor repository is under SVN under the Gforge project **kepleractors** .

The policy is that people use SVN/trunk for development versions, while official releases should be done under tags.

To check out the repository please do

```
svn checkout https://gforge6.eufus.eu/svn/kepleractors target_dir
```

The structure of the actor_repository directory is:

```
kepleractors/trunk/datastructure_version/IMP/physics_topic/actor_base_name/ \
actor_public_version/
```

`IMP` : is imp12 .. imp5 or isip. IMPs can add another level fo more detailed classification, e.g. fixed boundary equilibrium, free boundary equilibrium, linear MHD, etc.

It is the responsibility of the actor provider to create the appropriate directories under SVN.

[67] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_gforge
[68] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_kepler
[69] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_actor
[70] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_gforge
[71] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_fc2k

Currently the **kepleractors** repository has the following subdirectories for the UAL release versions (6.1):

```
- 4.07b
- 4.07c
- 4.08a
- 4.08b
- 4.09a
```

Each UAL [72] release version hosts the following subdirectories for the ITM projects:

```
- amns
- edrg
- isip
- imp12
- imp3
- imp4
- imp5
- ism
```

Below these the following physics topics are currently defined:

```
imp12:
 - fixed_boundary_equilibrium
 - free_boundary_equilibrium
 - linear_MHD
 - NTM
 - numerical_tools
 - RWM
 - sawtooth
```

11.2 Content of the actor repository

All files are stored at the bottom level of the tree structure. These are:

- actor TAR file generated by the `extract_actor` script (via the `put_repository` script)
- `actor_info.xml` file generated by the `put_repository` script
- FC2K parameter XML file (the FC2K parameters with which the actor has been created, obtained by selecting `save` in the FC2K menu)
- `actor_doc` file (PDF or TAR). Any useful and up-to-date documentation file (PDF recommended) should be gathered in a TAR archive with standardised name `actor_doc.tar` .

The `actor_info.xml` file is in an XML file gathering the following information (aiming at establishing a book-keeping link between the actor files and the source code, as stored in Gforge:

- Actor_name (as defined above)
- Gforge project name: name of the source code's project under Gforge
- SVN_rev: revision number of the source code in the SVN repository
- SVN_path: path of the source code in the SVN repository (e.g. `tag/v4.0`)
- FC2K_version: evaluated on the fly from `$FC2K`
- datastructure_version: evaluated on the fly from `$UAL`
- KEPLER_version: in ITM numbering

[72] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_ual

11.3 Procedure to put an actor in the actor repository

This procedure describes how to proceed to update the kepleractor repository according to the above organization.

11.3.1 Pre-requisites

- Have a copy of Kepler installed in your environment
- Run the script `ITMv1` to specify the working kepler directory (private) and set the environment variables (for FC2K and the UAL [73])
- Go into a directory in which you have write permission

11.3.2 How to

When you have generated a Kepler actor with FC2K, using the name convention as indicated above (actor_name), you can update the SVN repository :

- Prepare any `actor_doc` file (PDF or TAR)
- Run the script `put_repository`

  ```
  put_repository actor_name Gforge_project_name SVN_rev SVN_path
  ```

  The actions done by the `put_repository` script are:

  - Run the `extract_actor` script to take out the actor from the private Kepler, and generate an actor TAR file
  - Generate the `actor_info.xml` file, evaluating some information on the fly from environment variables as stated above
- Use the standard SVN commands to move in the relevant place in the SVN kepleractors repository the following files :
  - the FC2K parameter file
  - the `actor.tar` file generated by the `put_repository` script
  - the `actor_doc.tar` file
  - the `actor_info.xml` file generated by the `put_repository` script

11.4 Procedure to get an actor from the actor repository

The script `import_actor` allows to import an existing actor directly from the kepleractors SVN repository into your own distribution of Kepler.

The script search the actor repository under the Gforge project **kepleractors** (in the trunk and in the tags sections). Its content (actor TAR file, `actor_info.xml` file, and `actor_doc` file if exists) is copied in the current working directory and the actor extracted from the TAR file is added in the private copy of Kepler.

11.4.1 Pre-requisites

- Have a copy of Kepler installed in your environment
- Run the script `ITMv1` to specify the working kepler directory (private) and set the environment variables (for the UAL)
- Go into a directory in which you have write permission

[73] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_ual

11.4.2 How to import an actor from svn repository

- You do not know the location of the actor in the repository

  Usage:

```
import_actor -R
\textit{actor\_name
```

  Example:

```
import_actor -R gray
```

  path_to_the_actor_in_the_svn_repository is trunk/4.08b/imp5/electron_physics/gray
  actor_name is gray

  The script displays the location(s) of the actor in the svn repository (under trunk and tags subdirectories) and the user can choose the item he wants to import

- You know the location of the actor in the repository

  Usage:

```
import_actor -d
\textit{path\_to\_the\_actor\_in\_the\_svn\_repository actor\_name
```

  Example:

```
import_actor -d trunk/4.08b/imp5/electron_physics/gray gray
```

  path_to_the_actor_in_the_svn_repository is trunk/4.08b/imp5/electron_physics/gray
  actor_name is gray

11.4.3 How to import an actor from local location

Using the script `import_actor` , it is also possible to put into your private copy of Kepler an actor which is not stored in the SVN kepleractors repository.

In this case, you need an actor TAR file generated by the script `extract_actor` . This method is not recommended because the reproducibility of simulations cannot be ensured.

- Copy into the current directory or locate the tar file that contains the actor. The file does not have to be in your own directory. Only read permission is needed.

  Usage:

```
import_actor [
\textit{path
}]
\textit{actor\_name
```

  Example:

```
import_actor ~/private/ACTORS/gray
```

path is ~/private/ACTORS
actor_name is gray

path is only necessary if the tar file that contains the actor is not located in the current directory.

11.4.4 Additional options of the script import_actor

- -h : display usage information
- -f : force import of an already present actor
- -p : partial import; copy files but do not compile Kepler
- -s : skip import if the actor is already present

---

last update: 2019-01-31 by g2dpc

12 KeplerWorkflows

The **keplerworflows** project under GForge [74] is used for exchanging Kepler [75] workflows among ITM users.

Under this project, ITM developers should store the XML file describing the Kepler [76] workflow. The workflow should not be dependent on any other file. To be reproducible, public ITM workflows must use only ITM actors from the public kepleractors (16.1.2.1.1) repository with a unique naming convention.

Input datasets used by the workflow must be copied from the private to the public ITM database (email to Frederic.Imbeaux@cea.fr for the moment).

12.1 Structure of the workflows repository

The **keplerworkflows** repository is under SVN under the Gforge [77] project **KeplerWorkflows** .

It is recommended to use the `tags` directory for release versions and `trunk` for development versions: only public versions of `trunk/datastructure_version/` should be tagged as `tags/datastructure_version/` .

To check out the repository please do

```
svn checkout https://gforge6.eufus.eu/svn/keplerworkflows target_dir
```

The workflow is organised in the following way:

```
keplerworkflows/trunk/datastructure_version/IMP/physical_topic/workflowname
```

Currently the **keplerworkflows** repository has the following subdirectories for the UAL release versions (6.1):

```
- 4.07b
- 4.07c
- 4.08a
- 4.08b
- 4.09a
```

[74] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_gforge
[75] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_kepler
[76] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_kepler
[77] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_gforge

Each UAL [78] release version hosts the following subdirectories for the ITM projects:

```
- amns
- edrg
- isip
- imp12
- imp3
- imp4
- imp5
- ism
```

Below these the following physics topics are currently defined:

```
   imp12:
    - fixed_boundary_equilibrium
    - free_boundary_equilibrium
    - linear_MHD
```

12.2 Procedure to put a workflow in the workflows repository

The script `put_workflow` allows to commit a workflow (in fact, the xml file created by Kepler when the workflow is saved) in the svn repository https://gforge6.eufus.eu/svn/keplerworkflows.

12.2.1 Pre-requisites

- Have a copy of Kepler installed in your environment
- Run the script `ITMv1` to specify the working kepler directory (private) and set the environment variables (for the UAL)
- Go into a directory in which you have write permission

12.2.2 How to commit a workflow in the repository

Copy into the current directory or locate the xml file describing the workflow. This xml file is created by Kepler when you save a designed workflow. The file does not have to be in your own directory. Only read permission is needed.

Usage :

```
put_workflow [options] d
\textit{svnpath
} [
\textit{path
}/]
\textit{workflowname
```

svnpath is mandatory after -d to point out the target location of the workflow in the svn repository.
path is only necessary if the xml file of the workflow is not located in the current directory.

Example :

```
put_workflow d trunk/4.08b/isip/examples/loopexample ./loopexample/loopexample.xml
```

[78] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_ual

svnpath is isip/examples/loopexample
path is ./loopexample
workflowname is loopexample.xml

The script `put_workflow` copies the workflow xml file into the folder *svnpath* under the **KeplerWorklows** SVN repository.

NB : The subdirectory of `trunk` or `tags` corresponding to the data structure version must match the current $DATAVERSION environment variable

12.2.3 Additional options

- -h : display usage information
- -c : create the target directory in the svn repository **keplerworkflows** if it does not exist
- -m : message for svn import

12.3 Procedure to get a workflow from the workflows repository

The script `import_svn_workflow` allows you to copy a workflow from the svn repository https://gforge6.eufus.eu/svn/kep in your workspace.

12.3.1 Pre-requisites

- Have a copy of Kepler installed in your environment
- Run the script `ITMv1` to specify the working kepler directory (private) and set the environment variables (for the UAL)
- Go into a directory in which you have write permission

12.3.2 How to import workflow from svn repository

- You do not know the location of the workflow in the repository

  Usage:

  ```
  import_svn_workflow -R
  \textit{workflowname
  ```

  Example:

  ```
  import_svn_workflow R loopexample.xml
  ```

  The script displays the location(s) of the workflow in the svn repository (in `trunk` or `tags` subdirectories) and the user can choose the item he wants to import.

- You know the location of the workflow in the repository

  Usage:

  ```
  import_svn_workflow -d
  \textit{path\_to\_the\_workflow\_in\_the\_svn\_repository workflowname
  ```

  Example:

  ```
  import_svn_workflow -d trunk/4.08b/isip/examples/loopexample  loopexample.xml
  ```

path_to_the_workflow_in_the_svn_repository is trunk/4.08b/isip/examples/loopexample
workflowname is loopexample.xml

- Import the actors used by the workflow

  If the option **-a** is added to the previous command lines, the script `import_svn_workflow` also imports the actors used by the workflow, if they exist in the **KeplerActors** SVN repository.newline/¿ For each actor, the script `import_svn_workflow` launch the command

  ```
  import_actor -R
  \textit{actor\_name
  ```

  and search in the **Kepleractors** SVN repository if the actor exists, and if exists, import it in the private copy of Kepler.

  For more details about the script `import_actor` , see the section

  How to import an actor from svn repository (16.1.2.1.18).

NB : The subdirectory of `trunk` or `tags` corresponding to the data structure version must match the current $DATAVERSION environment variable

12.3.3 Additional options

- -h : display usage information
- -a : import the actors used by the workflow from the svn repository
- -f : ignored if -a not present; force import of an already present actor
- -p : ignored if -a not present; partial import: copy files but do not compile Kepler"
- -s : ignored if -a not present; skip import if the actor is already present

---

last update: 2019-01-31 by g2dpc

13 Integrated Simulation Editor (ISE)

The Integrated Simulation Editor ISE [79] allows to visualise and edit data from an ITM database entry. It also allows running a Kepler [80] workflow based on the opened data entry.

ISE is launched by simply typing **ise** in a terminal.

A short ISE tutorial can be found here [81].

---

last update: 2011-05-26 by imbeaux

14 Tools

14.1 MDSplus

Default private databases are MDSplus. Before using MDSplus tools to access them you must define an environment variable:

```
setenv euitm_path $HOME/public/itmdb/itm_trees/test/4.08a/mdsplus/0
```

[79] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_ise
[80] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_kepler
[81] https://www.efda-itm.eu/ITM/imports/isip/public/isip_IntroductionISE.pdf

Now you can use **mdstcl** , **jTraverser** and **jScope** to access the data.

To access MDSplus data using jTraverser, select *file* > *open* , then Tree: euitm, Shot: <the number following euitm_ in the database file name>.

---

last update: 2010-08-03 by konz

15 Gateway

ENEA Gateway Documentation: [82] Read this first!

OpenAFS Home: [83] Everything you want to know about AFS.

Gateway User's Guide: [84] Older gateway documentation.

15.1 Access Forms for the ITM Gateway

15.1.1 How to get an account on the ITM Gateway

As a new contributor to the ITM-TF or a new user on the ITM-TF Gateway [85] you are requested to sign the ITM-TF Gateway User Agreement (doc) [86](pdf) [87]. This is required to get an account on the Gateway the ITM-TF development home.

Please fill in the requested information and send the signed document to

Att: ITM-TF/Gloria Falchetto
Association EURATOM-CEA
DSM/IRFM/SCCP, bt. 513/141
CEA-Cadarache
13108 Saint Paul-Lez-Durance Cedex
France

Or fax to:

+33 442 25 6233

Or send as an e-mail attachment to (Subject: Gateway User Agreement):

gloria.falchetto@cea.fr

New Gateway User Greeting (doc) [88](pdf) [89]

15.1.2 ITM policy on Access Rights and Software Licencing

The ITM has defined a model licence for all physics codes and numerical tools (including the ITM infrastructure) that have been contributed/developed within the framework of the ITM Workprogramme. This model licence [90] was approved by the EFDA Steering Committee on October 2009.

---

last update: 2010-11-23 by konz

[82] http://www.efda-itm.eu/docs/documentation.php
[83] http://www.openafs.org
[84] https://www.efda-itm.eu/ITM/imports/isip/public/isip_ITM_gateway_users_guide_v3-1.pdf
[85] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_gateway
[86] https://www.efda-itm.eu/ITM/imports/itm/public/gateway/GatewayUserAgreement_ITM.doc
[87] https://www.efda-itm.eu/ITM/imports/itm/public/gateway/GatewayUserAgreement_ITM.pdf
[88] https://www.efda-itm.eu/ITM/imports/itm/public/gateway/GUA_invite.doc
[89] https://www.efda-itm.eu/ITM/imports/itm/public/gateway/GUA_invite.pdf
[90] https://www.efda-itm.eu/ITM/imports/itm/public/gateway/Model_licence_for_the_ITM.pdf

15.2 Using SSH

You can login to the Gateway [91] directly with SSH:

```
ssh -X <user>@gateway.efda-itm.eu
```

-X is required if you want to run X applications over the connection.

15.3 Using SFTP

Use SFTP to transfer files to/from the Gateway:

```
sftp <user>@gateway.efda-itm.eu
```

If sftp fails with 'Connection closed', or you get an error message like 'Received message too long (or "Bad packet length") 1416586337', your shell startup script (eg ~/.cshrc, ~/.bashrc) on the Gateway system is writing to standard output and confusing it. This is a known problem with sftp. Note that .login is evidently not run when and sftp session is started; standard output from that does not upset it. The solution is to disable the output or wrap the offending script in a conditional statement so it is only executed if the shell is interactive. For example, with cshrc:

```
if ($?prompt) then
   <original contents>
endif
```

For more see faq1 [92] and faq2 [93]. Thanks to Francesco Iannone for this information.

15.4 Using NX

NX from NoMachine [94] allows you to run a remote X11 login session on the Gateway from your local machine. To use it you must first download an NX client from NoMachine and install it locally. The default installation location for the NX executable is /usr/NX/bin/nxclient. When you run NX for the first time, after logging in with your assigned user name and password, and providing a session name, a configuration dialogue appears. Set the following values:

```
Host:    gateway.efda-itm.eu
Port:    22
Key:     default
Network: WAN
Desktop: Unix, KDE
Display: Available area
```

Do not disable encryption in the Advanced tab.

When this is done, the Gateway window manager should appear each time you run NX with the session name you provided.

15.5 Disk Quota

Users' home directories are in the Andrew File System. To check your disk quota do

```
fs listquota
```

[91] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_gateway
[92] http://www.tfn.net/~bsbaker/support/helplink/html/supp/supp-connect-use-sftp.html
[93] http://www.snailbook.com/faq/sftp-corruption.auto.html
[94] http://www.nomachine.com

Note that Kepler [95] ITM workflow core dumps can be ¿1.6GB in size!!

last update: 2010-09-17 by konz

16 Portal

WORK IN PROGRESS

https://portal.efda-itm.eu/portal [96]

16.1 GForge

16.1.1 GForge Documentation

The GForge Group L.L.C. has published extensive documentation material in the form of the GForge AS User Manual [97] and the GForge AS Project Administrator Manual [98]. Please refer to these manual if in doubt on how to use a feature of the ITM GForge system.

16.1.2 GForge Projects

16.1.2.1 Kepler Projects

The projects listed here are all related to the workflow orchestration tool Kepler [99] .

16.1.2.1.1 KeplerActors

The **kepleractors** project under GForge [100] is used for exchanging Kepler [101] actors [102] among ITM users.

This project and its associated SVN repository have been created related to the GForge [103] project **Keplerworkflows (16.1.2.1.12)** for enhancing traceability and reproducibility of simulations.

We describe here a complete procedure that is recommended for traceability purpose. The goal is to be able to keep track of how the actor has been generated (Gforge project of the source code, FC2K [104] parameters, ...).

For public releases a unique name shall be assigned to the actor:

All **public actors** should be generated (and thus appear in Kepler) with a name which is the concatenation of `actor_base_name + actor_public_version` .
`actor_base_name` : this can be chosen arbitrarily, though we suggest that it corresponds to project name under Gforge, i.e. the name of the source code's project under Gforge when there is a one-to-one link between the Gforge project and the actor.
`actor_public_version` : is the number of public release for this particular actor.
`actor_name` : concatenation of actor_base_name and actor_public_version

16.1.2.1.2 Structure of the actor repository

The actor repository is under SVN under the Gforge project **kepleractors** .

The policy is that people use SVN/trunk for development versions, while official releases should be done under tags.

[95] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_kepler
[96] https://portal.efda-itm.eu
[97] https://www.efda-itm.eu/ITM/imports/isip/public/GFAS_User_Manual_5.4.pdf
[98] https://www.efda-itm.eu/ITM/imports/isip/public/GFAS_Project_Admin_Manual_5.4.pdf
[99] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_kepler
[100] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_gforge
[101] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_kepler
[102] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_actor
[103] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_gforge
[104] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_fc2k

To check out the repository please do

```
svn checkout https://gforge6.eufus.eu/svn/kepleractors target_dir
```

The structure of the actor_repository directory is:

```
kepleractors/trunk/datastructure_version/IMP/physics_topic/actor_base_name/ \
actor_public_version/
```

`IMP` : is imp12 .. imp5 or isip. IMPs can add another level fo more detailed classification, e.g. fixed boundary equilibrium, free boundary equilibrium, linear MHD, etc.

It is the responsibility of the actor provider to create the appropriate directories under SVN.

Currently the **kepleractors** repository has the following subdirectories for the UAL release versions (6.1):

```
- 4.07b
- 4.07c
- 4.08a
- 4.08b
- 4.09a
```

Each UAL [105] release version hosts the following subdirectories for the ITM projects:

```
- amns
- edrg
- isip
- imp12
- imp3
- imp4
- imp5
- ism
```

Below these the following physics topics are currently defined:

```
imp12:
 - fixed_boundary_equilibrium
 - free_boundary_equilibrium
 - linear_MHD
 - NTM
 - numerical_tools
 - RWM
 - sawtooth
```

[105] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_ual

16.1.2.1.3 Content of the actor repository

All files are stored at the bottom level of the tree structure. These are:

- actor TAR file generated by the `extract_actor` script (via the `put_repository` script)
- `actor_info.xml` file generated by the `put_repository` script
- FC2K parameter XML file (the FC2K parameters with which the actor has been created, obtained by selecting `save` in the FC2K menu)
- `actor_doc` file (PDF or TAR). Any useful and up-to-date documentation file (PDF recommended) should be gathered in a TAR archive with standardised name `actor_doc.tar` .

The `actor_info.xml` file is in an XML file gathering the following information (aiming at establishing a book-keeping link between the actor files and the source code, as stored in Gforge:

- Actor_name (as defined above)
- Gforge project name: name of the source code's project under Gforge
- SVN_rev: revision number of the source code in the SVN repository
- SVN_path: path of the source code in the SVN repository (e.g. `tag/v4.0`)
- FC2K_version: evaluated on the fly from `$FC2K`
- datastructure_version: evaluated on the fly from `$UAL`
- KEPLER_version: in ITM numbering

16.1.2.1.4 Procedure to put an actor in the actor repository

This procedure describes how to proceed to update the kepleractor repository according to the above organization.

16.1.2.1.5 Pre-requisites

- Have a copy of Kepler installed in your environment
- Run the script `ITMv1` to specify the working kepler directory (private) and set the environment variables (for FC2K and the UAL [106])
- Go into a directory in which you have write permission

16.1.2.1.6 How to

When you have generated a Kepler actor with FC2K, using the name convention as indicated above (actor_name), you can update the SVN repository :

- Prepare any `actor_doc` file (PDF or TAR)
- Run the script `put_repository`

  ```
  put_repository actor_name Gforge_project_name SVN_rev SVN_path
  ```

  The actions done by the `put_repository` script are:

  - Run the `extract_actor` script to take out the actor from the private Kepler, and generate an actor TAR file
  - Generate the `actor_info.xml` file, evaluating some information on the fly from environment variables as stated above
- Use the standard SVN commands to move in the relevant place in the SVN kepleractors repository the following files :

[106] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_ual

- the FC2K parameter file
- the `actor.tar` file generated by the `put_repository` script
- the `actor_doc.tar` file
- the `actor_info.xml` file generated by the `put_repository` script

16.1.2.1.7 Procedure to get an actor from the actor repository

The script `import_actor` allows to import an existing actor directly from the kepleractors SVN repository into your own distribution of Kepler.

The script search the actor repository under the Gforge project **kepleractors** (in the trunk and in the tags sections). Its content (actor TAR file, `actor_info.xml` file, and `actor_doc` file if exists) is copied in the current working directory and the actor extracted from the TAR file is added in the private copy of Kepler.

16.1.2.1.8 Pre-requisites

- Have a copy of Kepler installed in your environment
- Run the script `ITMv1` to specify the working kepler directory (private) and set the environment variables (for the UAL)
- Go into a directory in which you have write permission

16.1.2.1.9 How to import an actor from svn repository

- You do not know the location of the actor in the repository
  Usage:

```
import_actor -R
\textit{actor\_name
```

  Example:

```
import_actor -R gray
```

  path_to_the_actor_in_the_svn_repository is trunk/4.08b/imp5/electron_physics/gray
  actor_name is gray

  The script displays the location(s) of the actor in the svn repository (under trunk and tags subdirectories) and the user can choose the item he wants to import

- You know the location of the actor in the repository
  Usage:

```
import_actor -d
\textit{path\_to\_the\_actor\_in\_the\_svn\_repository actor\_name
```

  Example:

```
import_actor -d trunk/4.08b/imp5/electron_physics/gray gray
```

  path_to_the_actor_in_the_svn_repository is trunk/4.08b/imp5/electron_physics/gray
  actor_name is gray

16.1.2.1.10 How to import an actor from local location

Using the script `import_actor` , it is also possible to put into your private copy of Kepler an actor which is not stored in the SVN kepleractors repository.

In this case, you need an actor TAR file generated by the script `extract_actor` . This method is not recommended because the reproducibility of simulations cannot be ensured.

- Copy into the current directory or locate the tar file that contains the actor. The file does not have to be in your own directory. Only read permission is needed.

  Usage:

  ```
  import_actor [
  \textit{path
  }]
  \textit{actor\_name
  ```

  Example:

  ```
  import_actor ~/private/ACTORS/gray
  ```

  path is ~/private/ACTORS
  actor_name is gray

  path is only necessary if the tar file that contains the actor is not located in the current directory.

16.1.2.1.11 Additional options of the script import_actor

- -h : display usage information
- -f : force import of an already present actor
- -p : partial import; copy files but do not compile Kepler
- -s : skip import if the actor is already present

---

last update: 2019-01-31 by g2dpc

16.1.2.1.12 KeplerWorkflows

The **keplerworflows** project under GForge [107] is used for exchanging Kepler [108] workflows among ITM users.

Under this project, ITM developers should store the XML file describing the Kepler [109] workflow. The workflow should not be dependent on any other file. To be reproducible, public ITM workflows must use only ITM actors from the public kepleractors (16.1.2.1.1) repository with a unique naming convention.

Input datasets used by the workflow must be copied from the private to the public ITM database (email to Frederic.Imbeaux@cea.fr for the moment).

16.1.2.1.13 Structure of the workflows repository

The **keplerworkflows** repository is under SVN under the Gforge [110] project **KeplerWorkflows** .

It is recommended to use the `tags` directory for release versions and `trunk` for development versions: only public versions of `trunk/datastructure_version/` should be tagged as `tags/datastructure_version/` .

To check out the repository please do

[107] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_gforge
[108] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_kepler
[109] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_kepler
[110] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_gforge

```
svn checkout https://gforge6.eufus.eu/svn/keplerworkflows target_dir
```

The workflow is organised in the following way:

```
    keplerworkflows/trunk/datastructure_version/IMP/physical_topic/workflowname
```

Currently the **keplerworkflows** repository has the following subdirectories for the UAL release versions (6.1):

```
- 4.07b
- 4.07c
- 4.08a
- 4.08b
- 4.09a
```

Each UAL [111] release version hosts the following subdirectories for the ITM projects:

```
- amns
- edrg
- isip
- imp12
- imp3
- imp4
- imp5
- ism
```

Below these the following physics topics are currently defined:

```
    imp12:
     - fixed_boundary_equilibrium
     - free_boundary_equilibrium
     - linear_MHD
```

16.1.2.1.14 Procedure to put a workflow in the workflows repository

The script `put_workflow` allows to commit a workflow (in fact, the xml file created by Kepler when the workflow is saved) in the svn repository https://gforge6.eufus.eu/svn/keplerworkflows.

16.1.2.1.15 Pre-requisites

- Have a copy of Kepler installed in your environment
- Run the script `ITMv1` to specify the working kepler directory (private) and set the environment variables (for the UAL)
- Go into a directory in which you have write permission

16.1.2.1.16 How to commit a workflow in the repository

Copy into the current directory or locate the xml file describing the workflow. This xml file is created by Kepler when you save a designed workflow. The file does not have to be in your own directory. Only read permission is needed.

Usage :

[111] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_ual

```
put_workflow [options] d
\textit{svnpath
} [
\textit{path
}/]
\textit{workflowname
```

svnpath is mandatory after -d to point out the target location of the workflow in the svn repository.
path is only necessary if the xml file of the workflow is not located in the current directory.

Example :

```
put_workflow d trunk/4.08b/isip/examples/loopexample ./loopexample/loopexample.xml
```

svnpath is isip/examples/loopexample
path is ./loopexample
workflowname is loopexample.xml

The script `put_workflow` copies the workflow xml file into the folder *svnpath* under the **KeplerWorklows** SVN repository.

NB : The subdirectory of `trunk` or `tags` corresponding to the data structure version must match the current $DATAVERSION environment variable

16.1.2.1.17 Additional options

- -h : display usage information
- -c : create the target directory in the svn repository **keplerworkflows** if it does not exist
- -m : message for svn import

16.1.2.1.18 Procedure to get a workflow from the workflows repository

The script `import_svn_workflow` allows you to copy a workflow from the svn repository https://gforge6.eufus.eu/svn/kep in your workspace.

16.1.2.1.19 Pre-requisites

- Have a copy of Kepler installed in your environment
- Run the script `ITMv1` to specify the working kepler directory (private) and set the environment variables (for the UAL)
- Go into a directory in which you have write permission

16.1.2.1.20 How to import workflow from svn repository

- You do not know the location of the workflow in the repository

  Usage:

  ```
  import_svn_workflow -R
  \textit{workflowname
  ```

  Example:

```
import_svn_workflow R loopexample.xml
```

The script displays the location(s) of the workflow in the svn repository (in `trunk` or `tags` subdirectories) and the user can choose the item he wants to import.

- You know the location of the workflow in the repository

  Usage:

```
import_svn_workflow -d
\textit{path\_to\_the\_workflow\_in\_the\_svn\_repository workflowname
```

  Example:

```
import_svn_workflow -d trunk/4.08b/isip/examples/loopexample  loopexample.xml
```

  path_to_the_workflow_in_the_svn_repository is trunk/4.08b/isip/examples/loopexample
  workflowname is loopexample.xml

- Import the actors used by the workflow

  If the option **-a** is added to the previous command lines, the script `import_svn_workflow` also imports the actors used by the workflow, if they exist in the **KeplerActors** SVN repository.newline/¿ For each actor, the script `import_svn_workflow` launch the command

```
import_actor -R
\textit{actor\_name
```

  and search in the **Kepleractors** SVN repository if the actor exists, and if exists, import it in the private copy of Kepler.

  For more details about the script `import_actor` , see the section

  How to import an actor from svn repository (16.1.2.1.18).

NB : The subdirectory of `trunk` or `tags` corresponding to the data structure version must match the current $DATAVERSION environment variable

16.1.2.1.21 Additional options

- -h : display usage information
- -a : import the actors used by the workflow from the svn repository
- -f : ignored if -a not present; force import of an already present actor
- -p : ignored if -a not present; partial import: copy files but do not compile Kepler”
- -s : ignored if -a not present; skip import if the actor is already present

---

last update: 2019-01-31 by g2dpc

16.1.2.2 Infrastructure Projects

---

last update: 2010-08-24 by konz

---

last update: 2012-07-18 by coster

---

last update: 2010-08-15 by konz

17 Training

17.1 Tutorials

UAL tutorial [112]: only for those who want to test their ITM compliant program OUTSIDE Kepler
Kepler tutorial [113]: shows the steps you have to go through to build an actor from a physics subroutine and use it in a Kepler workflow

17.2 Garching, March 2012

Tutorial website: http://scilla.man.poznan.pl/garching2012 [114]

Presentations:

- ETS-C Workflow (T. Aniel, V. Basiuk, P. Huyn): Slides [115]
  Material is available at /afs/efda-itm.eu/isip/user/huynh/public/GARCHING2011
- ETS-A Workflow (D. Kalupin): Tutorial [116]
- Stability chain (W. Zwingmann) Slides [117]
  Material is available at ~zwolf/public/GARCHING2012/
- AMNS interface (D.Coster) Slides [118], Readme [119]
- C actor (H. Klingshirn) Slides [120], Readme [121]
- General grid description (H. Klingshirn) Slides [122], Readme [123]

17.3 Garching, September 2011

ITM-Serpens-Garching2011 [124]:

- installation of Kepler,
- creation of basic workflows,
- conditional execution and looping,
- embedding and execution of Python code inside of Kepler actor,
- usage of FC2K for generation of actors from Fortran or C++ code,
- usage of Integrated Simulation Editor (ISE) for data visualization and Kepler workflow execution,
- usage of HPC2K for generation of grid/HPC actors running Fortran or C++ code remotely,
- submission of parametric grid jobs.

17.4 Cadarache May 2009

Introduction: [125]general presentation, CPOs, database
Exercises: [126]manipulate ITM databases and create an ITM physics subroutine
Kepler Tutorial: [127]introduction to the Kepler workflow system
Kepler Exercises: [128]build a Kepler workflow from an ITM physics subroutine

[112] https://www.efda-itm.eu/ITM/imports/isip/public/isip_UAL_TUTORIAL.pdf
[113] https://www.efda-itm.eu/ITM/imports/isip/public/isip_TutorialKepler.pdf
[114] http://scilla.man.poznan.pl/garching2012
[115] https://www.efda-itm.eu/ITM/imports/isip/public/isip_Training_201203_Huynh_ETS.pdf
[116] https://www.eufus.eu/documentation/ITM/html/ETS_in_KEPLER.html
[117] https://www.efda-itm.eu/ITM/imports/isip/public/isip_Training_201203_WZwingmann_equistab_V2.1.pdf
[118] https://www.efda-itm.eu/ITM/imports/isip/public/isip_Training_201203_Coster_Using_AMNS_tools.pdf
[119] https://www.efda-itm.eu/ITM/imports/isip/public/isip_Training_201203_Coster_Using_AMNS_tools_README.txt
[120] https://www.efda-itm.eu/ITM/imports/isip/public/isip_Training_201203_Klingshirn_CActor.pdf
[121] https://www.efda-itm.eu/ITM/imports/isip/public/isip_Training_201203_Klingshirn_CActor.README.txt
[122] https://www.efda-itm.eu/ITM/imports/isip/public/isip_Training_201203_Klingshirn_Grid.pdf
[123] https://www.eufus.eu/documentation/ITM/html/imp3_grid_tutorial.html
[124] https://www.efda-itm.eu/ITM/imports/isip/public/ITM-Serpens-Garching2011.pdf
[125] https://www.efda-itm.eu/ITM/imports/isip/public/isip_ISIP_Training_May2009.pdf
[126] https://www.efda-itm.eu/ITM/imports/isip/public/isip_ISIP_ExercisePhysicsModule_May2009.pdf
[127] https://www.efda-itm.eu/ITM/imports/isip/public/isip_KeplerTutorial_BG_v1.pdf
[128] https://www.efda-itm.eu/ITM/imports/isip/public/isip_KeplerExercises_BG_v1.pdf

Fortran XML Parser: [129]F95 library for parsing XML code parameters
Experimental Data: [130]overview of experimental data tools

---

last update: 2019-01-31 by g2dpc

18 Timeline

This ISIP timeline for 2010 was prepared by F. Imbeaux and G. Manduchi.

In 2009, we have for the first time assembled a complete version of the platform, from reading experimental data, actor [131] generation and management, to robust workflows including time evolution.

In 2010, ISIP needs to expand the functionality of these tools and upgrade them in a coordinated way with the needs of the IMPs.

Milestone for 2010 (expected for July) : full set of tools operational including visualisation tools, link to the simulation catalogue and operability of Kepler [132] actors on GRID & HPC

We propose the following timeline :

- **March:**
  - Deliver the second version of Kepler (in ITM numbering) : new features : improved expression actors, semantic type checking, Matlab actor ->DONE
  - Debugging procedure in Kepler ->DONE
  - Batch and MPI modes for ITM Kepler actors (FC2K [133]) ->DONE
  - Deliver ISE [134] ->DONE
- **April:**
  - Deliver data structure 4.08a, with many additions from IMPs (new CPOs [135]) ->DONE
  - Collect requirements of IMPs on Visit
  - Finalised tool for visualisation (Matplotlib)
- **May:**
  - Add Visit Kepler Actor (composite version) to the public Kepler distribution
  - Finalised release of the ITM control toolbox (link between Scicos and Kepler)
  - Evaluate strategy for parallel I/O for the UAL [136]
- **June:**
  - Deliver requirements for simulation catalogue querying tool
  - Implement the possibility of arrays of structures in the UAL and data structure
  - Technical meeting with the Kepler team
  - Deliver web service actor and grid/HPC actor generators : WS2K and HPC2K
- **July:**
  - Add automated generation of "remote" UAL to the UAL distribution

[129] https://www.efda-itm.eu/ITM/imports/isip/public/isip_FortranXMLParser.pdf
[130] https://www.efda-itm.eu/ITM/imports/isip/public/isip_ExperimentalDataITM_v3.pdf
[131] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_actor
[132] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_kepler
[133] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_fc2k
[134] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_ise
[135] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_cpo
[136] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_ual

- New data structure version with Visit representation tags
- Deliver "newcomer ITM platform User Guide"
- Implement full link with database through the UAL in workflows : UALinit, UALcollector communicate with the simulation catalogue (technically : change CPO definition for Kepler, change in FC2K)
- Memory caching activated in Kepler workflows (running on a single node) (technically : changes on UAL, UALinit, UALcollector)

- **September:**
  - Improved code parameter edition forms in ISE/Kepler
- **October:**
  - ITM control toolbox : deliver link between Simulink and Kepler
  - Memory caching in Kepler workflows (memory shared by multiple nodes on the Gateway [137])
- **November:**
  - Launch a full ITM workflow on GRID
  - Deliver simulation catalogue querying tool
  - Merge HPC2K, WS2K and FC2K into a unified tool
  - Visit for visualisation of time dependence

---

last update: 2010-08-24 by konz

19 Links

19.1 Overviews

Par Strand's RUSA 2009 Presentation [138]

19.2 GForge Projects

Gateway User Board [139] > Wiki: [140] Gateway management
ITM Portal [141] > SVN [142] documentation and code
Data Structure [143] > SVN: [144] XML schemas
UAL [145] > SVN: [146] Universal Application Layer code
FC2K [147] > SVN: [148] Fortran/C to Kepler wrapper function generator
ISE [149] > Wiki [150] I SVN: [151] Integrated Simulation Editor
XMLLIB [152] > SVN: [153] F95 library for parsing XML code parameters

---

last update: 2019-01-31 by g2dpc

[137] https://www.efda-itm.eu/ITM/html/itm_glossary.html#g_gateway
[138] https://www.efda-itm.eu/ITM/imports/isip/public/isip_Integrated_Tokamak_Modeling.pdf
[139] https://gforge6.eufus.eu/project/gub
[140] https://gforge6.eufus.eu/project/gub/wiki
[141] https://gforge6.eufus.eu/project/itmportal
[142] https://gforge6.eufus.eu/project/itmportal/scmsvn
[143] https://gforge6.eufus.eu/project/datastructure
[144] https://gforge6.eufus.eu/project/datastructure/scmsvn
[145] https://gforge6.eufus.eu/project/ual
[146] https://gforge6.eufus.eu/project/ual/scmsvn
[147] https://gforge6.eufus.eu/project/fc2k
[148] https://gforge6.eufus.eu/project/fc2k/scmsvn
[149] https://gforge6.eufus.eu/project/ise
[150] https://gforge6.eufus.eu/project/ise/wiki
[151] https://gforge6.eufus.eu/project/ise/scmsvn
[152] https://gforge6.eufus.eu/project/xmllib
[153] https://gforge6.eufus.eu/project/xmllib/scmsvn

20 Meetings

20.1 2010/09/13-17 ITM General Meeting in Lisbon

20.1.1 Posters

- *WebService Actor Generator* (ppt [154]), by B. Guillerminet
- *HPC2K - GRID and HPC Actor Generator* (ppt [155]), by B. Guillerminet et al.
- *Parallel I/O in Simulation Workflows* (ppt [156]), by A. Galonska et al.
- *Exp2ITM - a generic access to shot based data for European Tokamaks* (ppt [157]), by J. Signoret et al.
- *Integrated Simulation Editor* (ppt [158]), by J. Signoret et al.
- *Feedback control Simulation under the ITM platform* (pdf [159]), by O. Barana et al.
- *Control Toolbox* (ppt [160]), by N. Signoret and G. Manduchi
- *The ITM-TF Simulation Catalogue* (ppt [161]), by F. Imbeaux et al.

---

last update: 2011-02-10 by konz

---

last update: 2010-11-23 by konz

21 The Welcome ITM platform User Guide

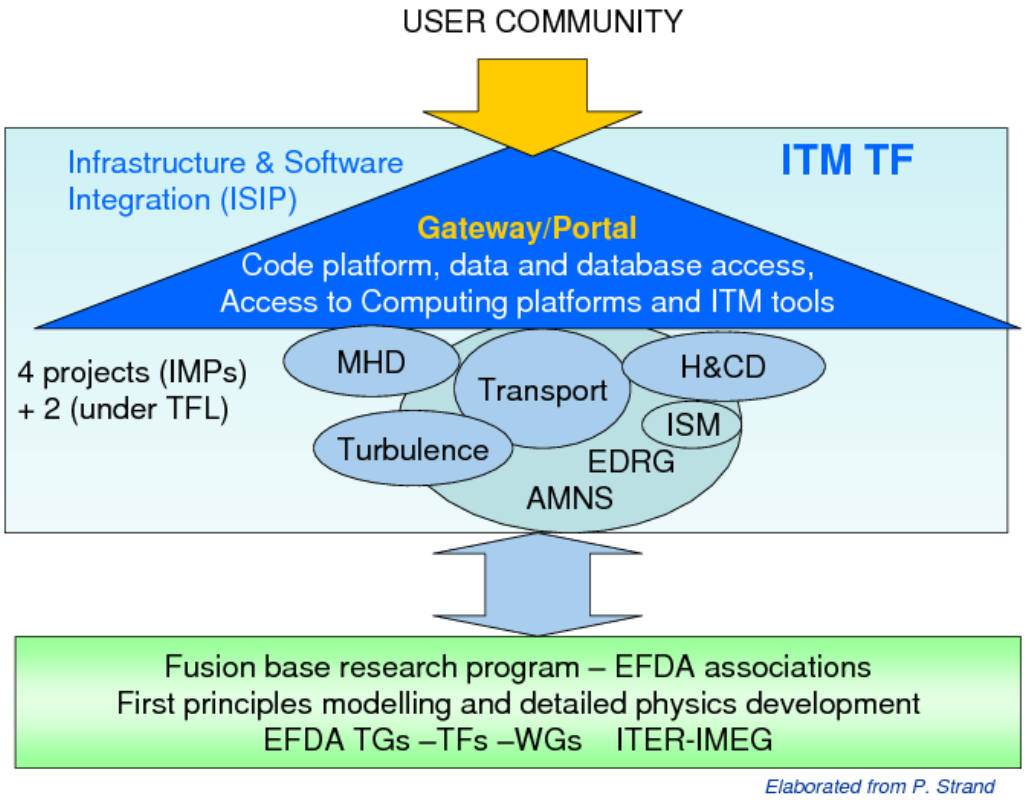
21.1 Welcome!

Welcome to the Newcomer's guide to EFDA-ITM platform!

The Integrated Tokamak Modeling [162] is a project of European Fusion Development Agreement [163] with the aim to providing software instruments to forecast and interpret discharges from ITER [164] and from DEMO. Therefore it deals with making codes developed by the European fusion community interact, with promoting the developing of codes on issues not yet faced and with validating the codes themselves.

To this purpose it is organised in 4 Integrated Modeling Projects (IMP12: MHD, IMP3: Transport, IMP4: Turbulence, IMP5: Heating and Current Drive), one Infrastructure and Software Integration Project (ISIP, with the purpose of providing developing and data analysis tools specific for fusion, together with the connective tissue among different codes and with the support to ITM members) and 2 interdisciplinary projects: the Experimentalist and Diagnostician Resource Group (EDRG) and the Atomic, Molecular Nuclear and Surface (AMNS).

[154] https://www.efda-itm.eu/ITM/imports/isip/public/meetings/20100913-17_Lisbon/poster_WS2K_v1.ppt
[155] https://www.efda-itm.eu/ITM/imports/isip/public/meetings/20100913-17_Lisbon/poster_HPC2K_v1.ppt
[156] https://www.efda-itm.eu/ITM/imports/isip/public/meetings/20100913-17_Lisbon/Poster_Parallel_UAL.ppt
[157] https://www.efda-itm.eu/ITM/imports/isip/public/meetings/20100913-17_Lisbon/Exp2ITM-GM2010.ppt
[158] https://www.efda-itm.eu/ITM/imports/isip/public/meetings/20100913-17_Lisbon/ISE-GM2010.ppt
[159] https://www.efda-itm.eu/ITM/imports/isip/public/meetings/20100913-17_Lisbon/ITM_Poster_Barana.pdf
[160] https://www.efda-itm.eu/ITM/imports/isip/public/meetings/20100913-17_Lisbon/Poster_T12-092010.ppt
[161] https://www.efda-itm.eu/ITM/imports/isip/public/meetings/20100913-17_Lisbon/SimulationCataloguePoster.ppt
[162] https://portal.efda-itm.eu/portal/
[163] http://www.efda.org/
[164] http://www.iter.org/



Moreover, it disposes of a Linux cluster of 128 cores with a peak performance of 1 teraflops: Gateway [165], which can be used for the developing and testing of the codes, and which is provided with *ad hoc* tools of pre- and post- processing and of interconnection between codes. Gateway also hosts the ITM data base storage (100TB), the ITM web site and the ITM cooperative work portal [166]. For massive production it is possible to use part of the supercomputer with 8640 cores and a peak performance of 0.1 petaflop called HPC-FF [167].

Every year at the beginning of September the ITM members meet for the ITM General Meeting to make the point on the advance status of the projects, update about the projects different from their own and learn how to use the new ISIP tools. Here [168] it is possible to have a look to previous year meetings to gather an idea about ITM activities

As regards organisation, ITM presents a Task Force Leader with 2 deputy leaders, and a Project Leader with one or more deputy leader for every IMP and for ISIP. Moreover the Gateway User Board, composed with the TFL and the leader of every IMP as a representative of the IMP users, determines the policy of Gateway administration. The EFDA-ITM project is accountable to the EFDA steering committee and its CSU is Denis Kalupin.

Every researcher of a country associated to EURATOM [169] is encouraged to cooperate with these projects. To do this the first thing is to contact the Contact Person at your Research Unit [170] who will, among other things, provide you with the name(s) of the project leader(s) of the project(s) you are interested in. People outside EURATOM Association interested in joining the task-force need to contact the Task Force Leader. Their request will be considered on an individual basis. The participation is ruled by EFDA calls issued once every year and sent to HRU and Contact Persons. Next call will be in November 2010.

---

last update: 2011-01-08 by marchett

[165] http://www.efda-itm.eu/
[166] https://portal.efda-itm.eu/portal/
[167] http://www.fz-juelich.de/jsc/juropa/configuration/
[168] http://itm2010.efda-itm.eu/index.php
[169] http://ec.europa.eu/research/energy/euratom/fusion/funding/index_en.htm
[170] https://www.efda-itm.eu/ITM/html/itm_contact_list_2010.html#itm_contact_list_2010

22 The Newcomer's ITM platform User Guide

Welcome to the EFDA-ITM Project and thank you for choosing to cooperate!

This guide is intended for a new user to his/her first access to ITM platform and therefore provides the basic instructions for a code developer, which might also serve as reference for everyday use to the expert user. Special cases or instructions for curious users or for those who want to develop infrastructure tools can be found on the documents linked.

22.1 Getting Started

If you have not yet done it, please contact the ITM Contact Person at your Research Unit [171].

To get an **access to the ITM web Site** Intranet (Portal) [172], with news about every project, tool and code documentation, specific mail lists and the collaborative work instruments, ask a login and a password to your Project Leader [173].

To get an **access to Gateway** [174], EFDA-ITM dedicated Linux cluster for code developing, download, fill and sign the Gateway User Agreement [175] and send it by fax to the Task Force Leader. [176] The GUA was written as a cooperation among the Gateway User Board [177] (composed by a leader from every project representing his/her users, determines the policy for Gateway administration), EURATOM legal office and the HRU's and protect the intellectual property of your work as well as it allows you to use software under license installed in Gateway (ITM policy still remains completely toward open source [178]). Without signing the GUA it is not possible to have access to Gateway and to most of information on these pages. The TFL will contact your representative inside GUB authorising him/her to give you a login and a password and he/she will communicate it to you by subscribing you to the **mail list of your project** (that makes no traffic: one mail every month on the average).

22.2 Let's work!

Having a look to the iter [179] a code must follow to finally become a part of the integrated tokamak modeling we can summarise the simplest activities you can do in ITM platform as:

- code porting and developing: for this you need to know the Gateway Development Environment (22.2.1)
- specific data visualisation and creation of a simulation -> beside the usual visualisation libraries, installed on Gateway, ISIP has developed the specific tool ISE (22.2.5)
- validation of a code against other codes and experimental data -> to correctly exchange data all the codes must have the same I/O structure and format -> Data Structure (22.2.2)
- exchanging data between a code and another -> again Data Structure (22.2.2)
- link codes to form a unique workflow -> kepler (22.2.3) and fc2k (22.2.4)
- code developing and maintenance with your colleagues maybe living elsewhere -> gforge (22.2.6) and svn (22.2.7)
- code preparation to be delivered to other users -> gforge (22.2.6)

22.2.1 How to use Gateway

Information about Gateway can be gather at the Gateway web site [180] where you can also download the Gateway User Guide [181].

You can access Gateway via ssh and sftp in the standard way.

[171] https://www.efda-itm.eu/ITM/html/itm_contact_list_2010.html#itm_contact_list_2010
[172] https://portal.efda-itm.eu/portal/
[173] https://www.efda-itm.eu/ITM/html/itm_contact_list_2010.html#itm_contact_list_2010
[174] http://www.efda-itm.eu/
[175] https://www.efda-itm.eu/ITM/imports/itm/public/gateway/GatewayUserAgreement_ITM.pdf
[176] http://www.efda-itm.eu
[177] http://www.efda-itm.eu
[178] http://opensource.org/docs/osd
[179] http://solps-mdsplus.aug.ipp.mpg.de:8080/ITM
[180] http://www.efda-itm.eu/
[181] http://www.efda-itm.eu/docs/docs5.php

You can enter it via NX from Nomachine [182], a remote desktop application which allows you to use features as graphical interfaces to applications (indispensable to user most of the ISIP tools) or graphical output from postprocessing software or multi terminals.... To do this follow this Euforia instructions [183], namely:

-get the NX client for your local OS from NX download page [184] and install it

-start NX Connection Wizard and follow the configuration steps

22.2.2 Data Structure

22.2.3 Kepler

22.2.4 FC2K

22.2.5 ISE

22.2.6 gforge

22.2.7 SVN

22.3 In case of trouble...

---

last update: 2012-07-18 by coster

---

last update: 2019-01-31 by g2dpc

[182] http://www.nomachine.com/index.php
[183] http://scilla.man.poznan.pl:8080/confluence/display/euforia/NX+setup
[184] http://www.nomachine.com/download.php