DE LA RECHERCHE À L'INDUSTRIE

The IMAS Data Dictionary : an introduction

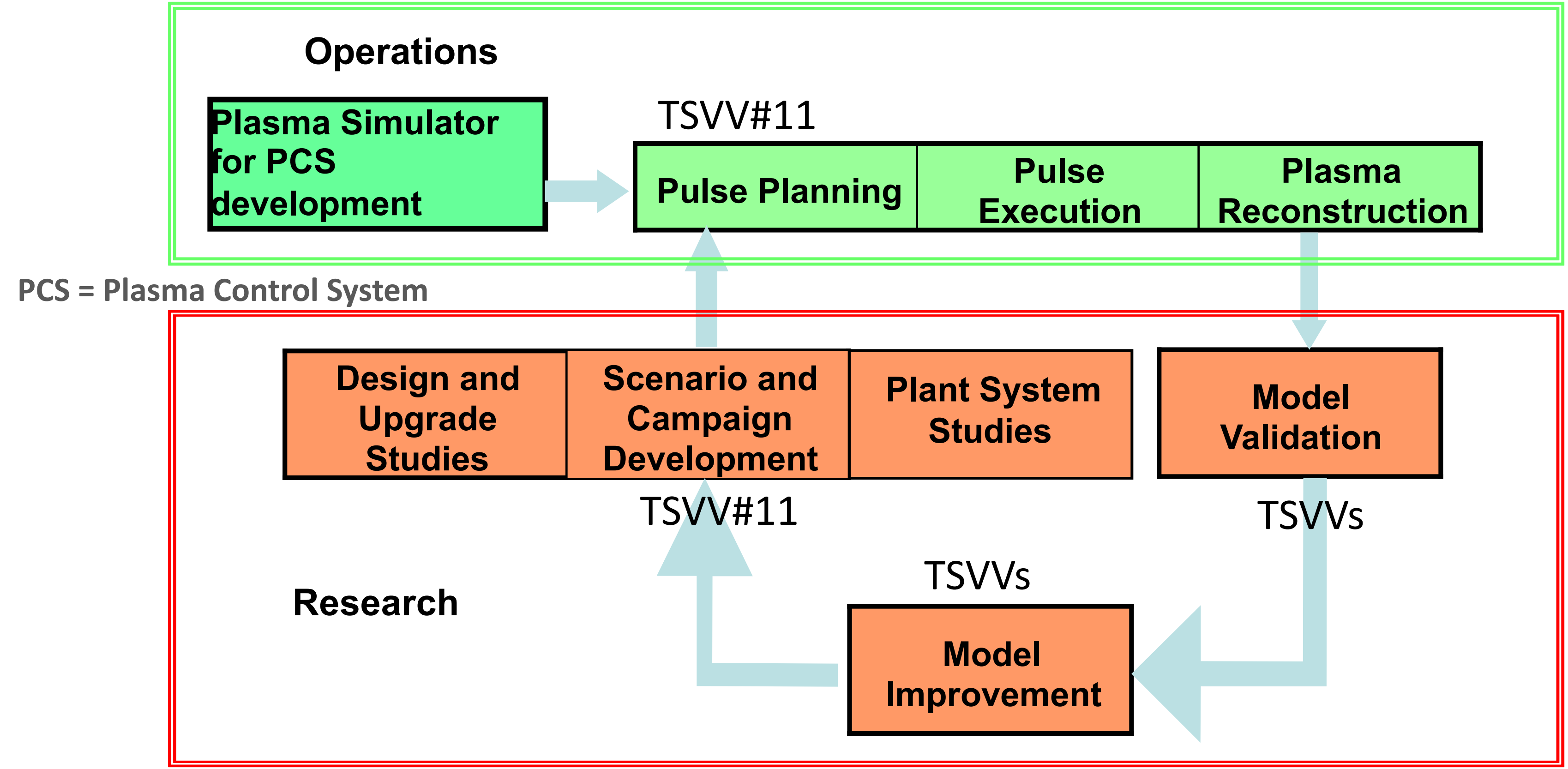
F. Imbeaux

20/09/2021

What is IMAS ?

- IMAS is the ITER Integrated Modelling and Analysis Suite
- Infrastructure :
  - Data Dictionary : a machine generic ontology for magnetic fusion :
    - What data exist ?
    - What are they called ?
    - How are they structured ?
  - Data Access : functions to read/write objects defined in the Data Dictionary
  - Workflow component generator : encapsulate physics codes to turn them into components that can be coupled in a workflow
- Physics applications : components (TSVV codes, adapted to use Data Dictionary objects as input/output) and workflows

- IMAS is the standard framework for joint scientific exploitation of ITER experiments by the ITER members



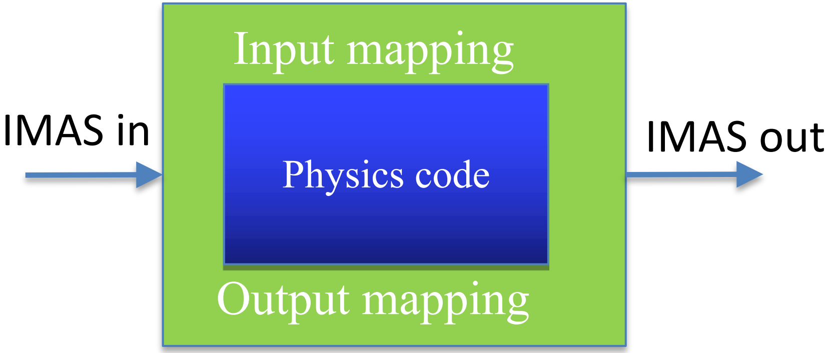
The IMAS Data Dictionary : a fusion standard

- The IMAS backbone is a machine-generic ontology : the physics Data Dictionary
  - Capable of covering all experiment subsystems and plasma physics, and is extensible
  - It represents simulation and experimental data with the same data structures, enabling direct comparisons
  - The Interface Data Structures (IDSs) are specific entry points of the Data Dictionary. They typically describe a tokamak subsystem (diagnostic, heating system, …) or an abstract physical concept (equilibrium, set of core plasma profiles, wave propagation, MHD, …)
  - They define standard interfaces between physics components in an IMAS workflow
- The IMAS Data Dictionary is being promoted as the standard to enable Interoperability in the FAIR and open science requirements for FP9 (Fair4Fusion project, EUROfusion Data Management Plan working group).

- Step 0 : the TSVV physics code

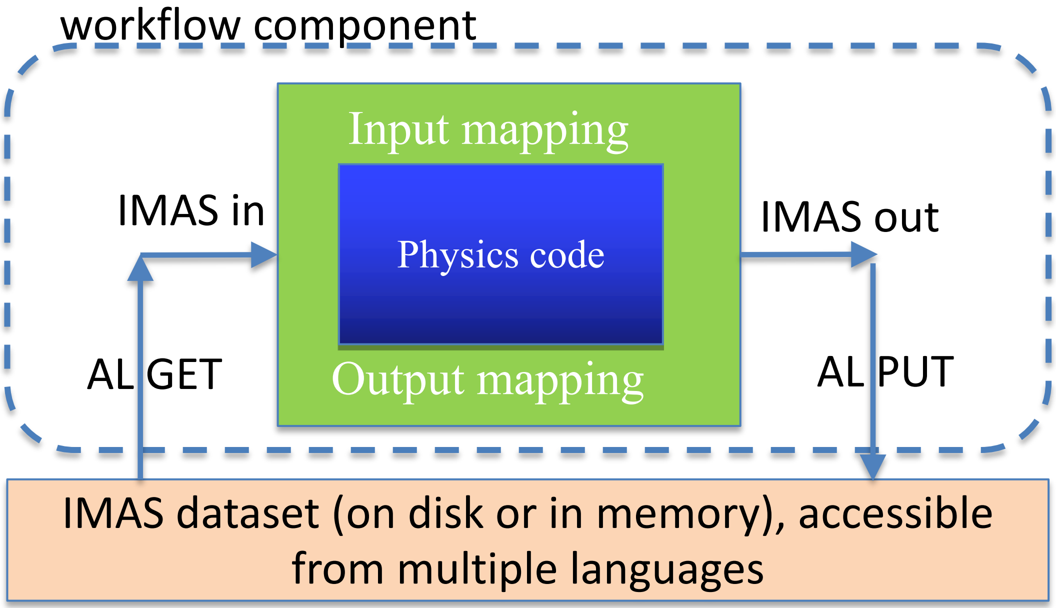
Physics code

- Step 1 : the TSVV physics code with I/O mapped to IMAS Data Dictionary

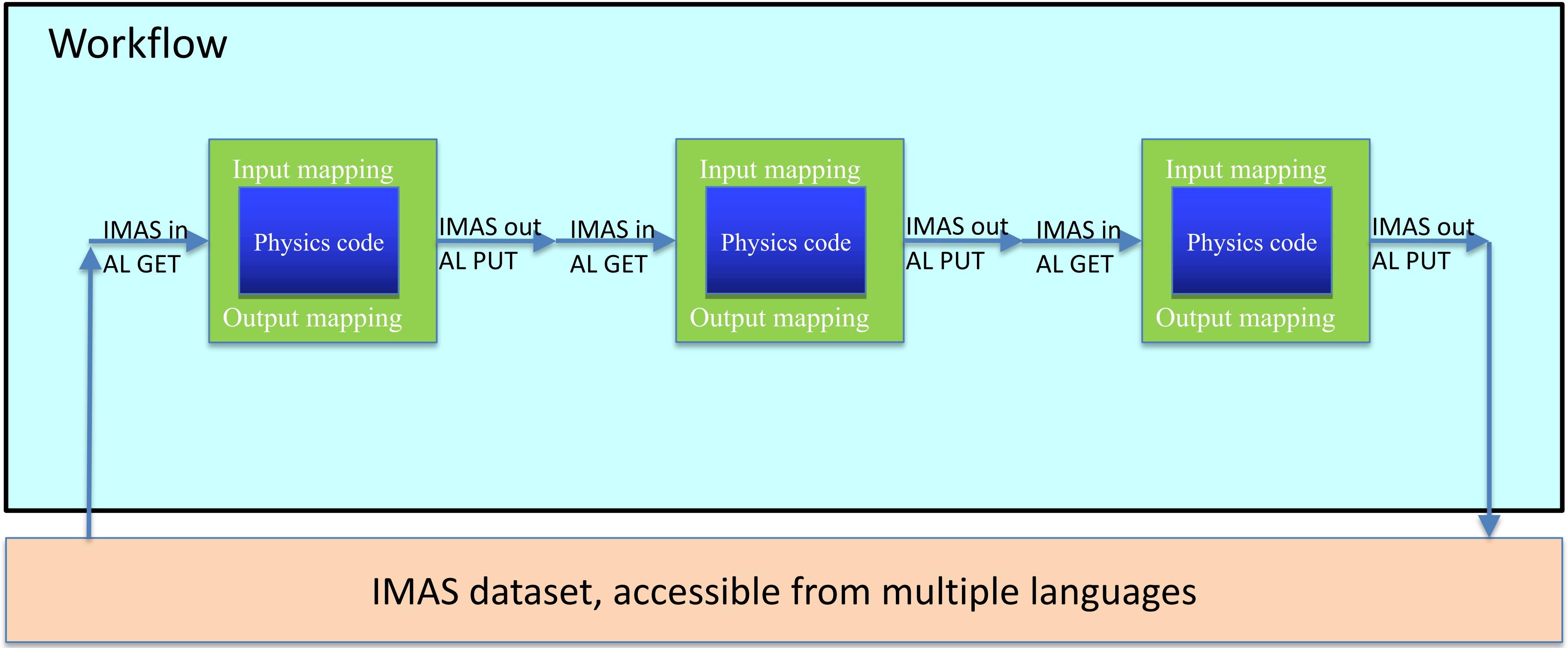


IMAS integration steps

- Step 2 : the TSVV physics code with I/O mapped to IMAS Data Dictionary uses the Access Layer to read/write data
- NB : this step, encapsulated, results in a potential workflow component



- Step 3 (TSVV#11 only): multiple TSVV physics code with I/O mapped to IMAS Data Dictionary use the Access Layer to communicate together in a workflow



What TSVVs can do with the IMAS infrastructure ?

- **Database : store/publish data using a fusion-standard ontology**
  - Store simulations results and compare to an experiment
  - Exchange simulation results with other codes (benchmarking, reuse of input datasets)
  - Create a catalogue of simulations that can be searched/browsed (various catalogue prototypes are under development : IO, Fair4Fusion, …)
  - Make data FAIR and Open (Fair4Fusion demonstrator)

- **Assemble a workflow of physics components** (Integrated Modelling, Simulation post-processing, Plasma Reconstruction Chains, …)
  - E.g. process synthetic diagnostic output from a simulation and compare to an experiment

What means « adapting a code to IMAS » ?

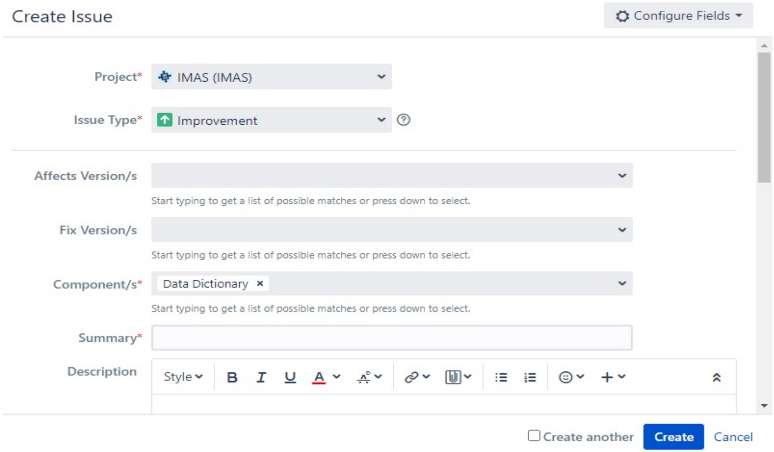
- Basic level (minimal requirement for EUROfusion standard software)
  - Agree on a minimal set of input/output data to be mapped to IMAS
  - Create mapping script that will do the mapping and read/write data to an IMAS database
- Full IMAS interface (required at some stage ?)
  - Map full I/O to IMAS, including code-specific parameters, and optionally restart files as well
- IMAS component
  - Full IMAS interface + generate component directly usable in workflows (Python, Kepler, …)

Minimal workflow for reading from / writing to IMAS
In green : what the TSVVs have to do

- The IMAS infrastructure has an API (Access Layer, AL) to read (GET) and write (PUT) IDS structures from a variable in your favorite language (Fortran, C++, Matlab, Python, Java) to a file (MDS+, ASCII, HDF5).
- OPEN the data entry of interest (for your input)
- GET the IDSs of interest for the input to your code
- Map the IMAS input to your code's data model
- Run your code
- Map your code output to IMAS
- OPEN/CREATE your output data entry
- PUT the output IDSs to the output data entry
- CLOSE data entries
- You are done !

TSVVs may have needs that are not covered yet by the IMAS Data Dictionary

- The IMAS Data Dictionary is extensible
- It has precise lifecycle procedure to be able to evolve and be jointly developed by multiple teams
- It has precise design rules to ensure global homogeneity
- Question/feature request ? : go to https://jira.iter.org/ and create an "issue" for the IMAS project, component "Data Dictionary"



Going deeper inside the IMAS DD structure

Deeper insight : Interface Data Structure

- The Data Model has a tree structure, for the sake of clarity
- At the top level, a collection of modular structures representing
  - Abstract physical quantities (e.g. distribution functions)
  - Tokamak subsystems (e.g. PF systems)
- By default, data access is made at the level of these structures (Write and Read)
- These modular structures have the appropriate granularity for exchange in an IM workflow → they also represent standardised interfaces for communication between codes, named Interface Data Structure (IDS)
  - Each has an “ids_properties” substructure (metadata + comments + timebase usage)
  - Each has a “code” substructure (trace the code-specific parameters of the code that has generated this IDS)
  - Each has a global timebase (“time”)

Data Dictionary documentation

- After having loaded an IMAS module, typing “dd_doc” will open the DD documentation (for the version that has been loaded)
- It first shows the list of all IDSs. For each of them, a detailed documentation:
  - Full path name: name of all variables of the IDSs, with their path in the structure. Replace “/” by the structure operator in a programming language, e.g. “%” in Fortran, “.” in C++, Matlab, Java, Python
  - Definition
  - Units in []
  - In {}, whether it is STATIC (constant over a range of pulses, e.g. machine configuration), CONSTANT (constant over the pulse or the simulation), or DYNAMIC (time-dependent within the pulse or the simulation)
  - Data_Type: indicates whether it is a string, an integer or a real, and its dimension (0D, 1D, 2D, …)
  - Coordinates: for each dimension, the full path name to the related coordinate. If the dimension simply refers to a quantity not present in the Data Model, it is indicated as “1…N”
  - DD lifecycle information

Example of using the documentation

- Go to the DM documentation and answer the following questions:
- In which IDS can I find the equilibrium ? (that’s an easy one)
- In this IDS, where can I find the toroidal flux profile calculated by my equilibrium code ?
- What are its units ?
- Does it vary during the pulse ?
- How many dimensions does it have ?
- What are its coordinates ?
- Assume I have retrieved a full equilibrium structure in my Fortran program, what syntax would I use for this variable ?

Example of using the documentation : solutions !

- Go to the DM documentation and answer the following questions:
- In which IDS can I find the equilibrium ? (that's an easy one) Equilibrium IDS
- In this IDS, where can I find the toroidal flux profile calculated by my equilibrium code ? search for "toroidal flux", found at path time_slice(:)/profiles_1d/phi
- What are its units ? Wb
- Does it vary during the pulse ? Yes, it's "dynamic"
- How many dimensions does it have ? 1D (float) at the leaf level, but note a time dimension is also there at the higher "time_slice" level
- What are its coordinates ? time_slice(:)/profiles_1d/psi
- Assume I have retrieved a full equilibrium structure in my Fortran program, what syntax would I use for this variable ? equilibrium% time_slice(:)%profiles_1d%phi

Time bases in IDSs

- "time" is a reserved node name for any timebase in the DD. Such nodes are recognized and used by the Access Layer when getting or putting time slices (GET_SLICE / PUT_SLICE functions).
- An IDS may contain quantities with different timebases in order to have the ability to describe experimental data as it is acquired in the experiment.
- However, an IDS can also be filled in a synchronous way (i.e. all dynamic quantities are stored on a unique timebase)
- There are therefore two possible usages of the IDS, with two possible locations for the "time" coordinate related to a given node. **Homogeneous_time is set by the data provider**.

Value of ids_properties/homogeneous_time	Location of the time coordinate for dynamic nodes
0	Dynamic nodes may be asynchronous, their timebase is located as indicated in the "Coordinates" column of the documentation.
1	All dynamic nodes are synchronous, their common timebase is the "time" node that is the child of the nearest parent IDS.
2	Means that no dynamic node is filled in the IDS (dynamic nodes will skipped by the Access Layer)

You don't need to fill everything when writing an IDS

- An IDS can contain a fairly large number of physical quantities and covers a wide range of applications. Therefore there will be many cases in which they are only partially filled.
- The only requirement regarding empty fields are:
  - The ids_properties/homogeneous_time field must be filled
  - When a quantity is filled, the coordinates of this quantity must be filled as well
- Not meeting these requirements when one of the coordinates is a time will cause PUT methods to return an error.

Arrays of structure

- Arrays of structures are frequently used in IDSs, to describe a list of elements that may have nodes of different sizes, in order to avoid creating large sparse arrays
- The two typical cases are :
- Case 1: lists of objects that may contain asynchronous nodes, e.g. PF coils may be acquired with different timebases :
  - pf_active%coil(i)%current%data(itime)
  - pf_active%coil(i)%current%time(itime)
  - These Case 1 AoS are used essentially in IDSs representing tokamak subsystems
- Case 2: list of time slices. The structure contains only dynamic and synchronous nodes, e.g. equilibrium/time_slice(:). This time slice representation allows the size of the children to vary as a function of time (e.g. variable grid size). These Case 2 AoS are used essentially in IDSs representing abstract physical quantities

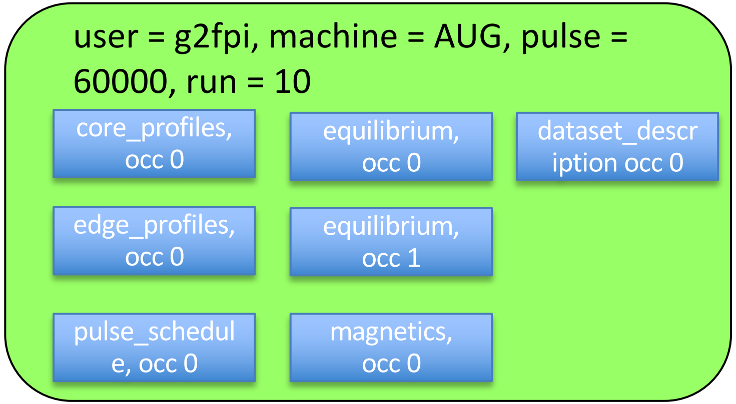
Lifecycle information

- The DD is a living object that evolves and expands with the needs. Therefore the lifecycle status of each node is documented
  - Some parts of the DD are recent and may evolve rapidly “lifecycle_status = alpha”
  - Some other parts are used for a longer time and are more stable “lifecycle_status = active”. If they need to be changed, they will become “obsolescent” but will not suddenly disappear (until a Major Release)
  - Some other parts are deprecated and shouldn’t be used “lifecycle_status = obsolescent”
- The lifecycle status of IDS nodes is described in the documentation. It applies to all descendants of a node, unless a descendant carries a different lifecycle status

IMAS Data entries

IMAS Data Entries for storing simulation input/output

- A Data Entry is a collection of potentially all IDS, gathered as a logical dataset (e.g. all IDSs corresponding to a given simulation output)
- A specific IDS, “dataset_description” is the placeholder for description of the content of the dataset
- Multiple occurrences of a given IDS can co-exist, e.g. multiple equilibria calculated by different codes / assumptions
- A Data Entry is defined by:
  - IMAS “major version” (=“3”)
  - User name
  - Machine name
  - Pulse number
  - Run number (multiple simulations related to the same pulse)
- Choose a File Backend to write to disk (MDS+, ASCII, HDF5)
  - Create or open a Data Entry
  - Then GET or PUT individual IDSs from/in it
- More generic way to define/localize data entries is under way (URI)



Which IMAS Backend to use ?

- Three Backends are available to write IMAS data to files
  - MDS+ Backend it the historical one. Advantage : well validated. Drawback : creates huge data files even for small amounts of data
  - ASCII Backend is not recommended for large data size, but may be interesting for testing purposes. Reduced functionalities (no time slice operation)
  - HDF5 Backend has been developed recently and IO is pushing for this technology. Contains already a number of performance and disk space optimization. We recommend using this one and report on any issue you have with your use case

- There are also other Backends
  - Memory Backend allows faster exchange of IDSs in memory (e.g. between components written in different languages)
  - UDA Backend allows reading (not writing, so far) data remotely, and includes an optional data conversion step (e.g. for reading experimental data not natively in IMAS format)

IMAS file structure when using the MDS+ Backend

- The standard location on the user account is:
  - For RUN numbers within 0-9999 : ~/public/imasdb/DatabaseName/IMASMajorVersion/0
  - For RUN numbers within 10000-19999 : ~/public/imasdb/DatabaseName/IMASMajorVersion/1
  - …
  - For RUN numbers within 90000-99999 : ~/public/imasdb/DatabaseName/IMASMajorVersion/9
- If the User name starts with a “/”, then it is interpreted as the absolute base path for the location of the IMAS data files:
  - For RUN numbers within 0-9999 : <Username>/imasdb/DatabaseName/IMASMajorVersion/0
- The present file names are (for the MDS+ backend):
  - ids_PulseRun.tree
  - ids_PulseRun.datafile
  - ids_PulseRun.characteristics
- Where Pulse is the pulse number and Run is the 4 rightmost digits of the run number of the Data Entry.
- Example: PULSE 22, RUN 2 consists of 3 files:
  - ids_220002.tree
  - ids_220002.datafile
  - ids_220002.characteristics
- In principle, users do not need to access directly those files, since data operations should go through the Access Layer.

IMAS file structure when using the ASCII Backend

- Data Entry is stored in an ASCII file on disk (by default this file is written in the current directory with a name like "<dbname>_<shot>_<run>_<idsname>.ids").
- Only PUT and GET are implemented so far (no *_SLICE operation).

IMAS file structure when using the HDF5 Backend

- All pulse files are located in the user's account under the folder: ~/public/imasdb/DatabaseName/IMASMajorVersion/SHOT/RUN
- Modular organization:
- One file per IDS and occurrence
- One master file with the references

« master » pulse file

```
HDF5 "/home/ITER/fleuryl/public/imasdb/test/3/9998/9998/master.h5" {
GROUP "/" {
  ATTRIBUTE "HDF5_BACKEND_VERSION" {
    DATATYPE  H5T_STRING {
      STRSIZE 10;
      STRPAD H5T_STR_NULLTERM;
      CSET H5T_CSET_UTF8;
      CTYPE H5T_C_S1;
    }
    DATASPACE  SCALAR
    DATA {
    (0): "1.0"
    }
  }
  ATTRIBUTE "RUN" {
    DATATYPE  H5T_STD_I32LE
    DATASPACE  SCALAR
    DATA {
    (0): 9998
    }
  }
  ATTRIBUTE "SHOT" {
    DATATYPE  H5T_STD_I32LE
    DATASPACE  SCALAR
    DATA {
    (0): 9998
    }
  }
  EXTERNAL_LINK "edge_profiles" {
    TARGETFILE "./ edge_profiles.h5"
    ...
  } EXTERNAL_LINK "edge_transport" {
    TARGETFILE "./edge_transport.h5"
    ...
  } EXTERNAL_LINK "edge_transport_1" {
    TARGETFILE "./edge_transport_1.h5"
    ...
  } EXTERNAL_LINK "equilibrium" {
    TARGETFILE "./ equilibrium.h5"
    ...
  }
```

Backend version

Run number

Shot number

Link to edge_transport IDS pulse file (occurrence 0)

Link to edge_transport IDS pulse file (occurrence 1)

Link to equilibrium IDS pulse file (occurrence 0)

→ One file per IDS and per occurrence

→ One master file referencing IDSs pulse files

```
[fleuryl@sdcc-login03]$ ls -alh ~/public/imasdb/test/3/9998/9998
total 60M
drwxrwsr-x. 2 fleuryl fleuryl 4.0K Jan 25 14:15 .
drwxrwsr-x. 3 fleuryl fleuryl 4.0K Jan 25 13:58 ..
-rw-rw-r--. 1 fleuryl fleuryl 6.4M Jan 25 14:15 edge_profiles.h5
-rw-rw-r--. 1 fleuryl fleuryl  17M Jan 25 14:15 edge_transport_1.h5
-rw-rw-r--. 1 fleuryl fleuryl  34M Jan 25 14:15 edge_transport.h5
-rw-rw-r--. 1 fleuryl fleuryl 2.4M Jan 25 14:15 equilibrium.h5
-rw-rw-r--. 1 fleuryl fleuryl 2.3K Jan 25 14:15 master.h5
```

edge_transport IDS pulse file (occurrence 1)

```
HDF5 "/home/ITER/fleuryl/public/imasdb/test/3/9998/9998/edge_transport_1.h5" {
GROUP "/" {
   ...
     GROUP "edge_transport_1" {
      ...
       DATASET "grid_ggd[]&grid_subset[]&base[]&jacobian" {
          DATATYPE  H5T_IEEE_F64LE
          DATASPACE  SIMPLE { ( 3, 3, 3, 3 ) / ( 3, 3, 3, 3 ) }
          DATA {
          (0,0,0,0): 320.188, 352.19, 687.804,
          (0,0,1,0): 548.362, 99.6258, 361.164,
          (0,0,2,0): 685.705, 298.591, 708.262,
       ...
     }
...
```

- Use the documentation:
  - dd_doc describes all IDSs, their structure and their nodes property
  - The IMAS Physics Data Model User Guide describes both the DD and the Access Layer
- Ask questions, feature requests on the Data Dictionary on JIRA https://jira.iter.org/