UK Atomic Energy Authority

DMP Discussion

MAST-U IMAS Mappings

May 12, 2023

Adam Parker, Jonathan Hollocombe, Stephen Dixon, Lucy Kogan

CCFE
CULHAM CENTRE FOR FUSION ENERGY

Outline

IMAS Introduction

IMAS Data Structure

Previous IDS Mappings

MAST-U Mappings Approach

JSON Mapping Structure

Summary

IMAS

- **Integrated Modelling & Analysis Suite** (IMAS)
  Software collection intended for all physics modelling and analysis for ITER

- Based on a standard data format and representation:
  - includes both experimental and simulation data,
  - combines machine description,
  - and provides a consistent coordinate system.

More information at https://imas.iter.org*

*ITER account needed

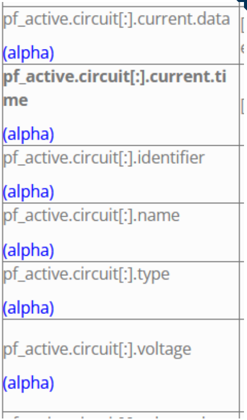
IMAS Data Structure

- Tree structure based on **Interface Data Structures** (IDS)

  Modular entities that separate physics components, diagnostics and machine independent concepts.

- The list of IDSs is defined in the **Data Dictionary** (DD)

  Describes structure and nomenclature. Continuously changing and grows depending on needs.



pf_active.circuit[:].current.data

(alpha)

pf_active.circuit[:].current.time

(alpha)

pf_active.circuit[:].identifier

(alpha)

pf_active.circuit[:].name

(alpha)

pf_active.circuit[:].type

(alpha)

pf_active.circuit[:].voltage

(alpha)

pf_active IDS

Current Data Dictionary version: 3.38.1*

IDS Version Example

Core IDS
core_profiles
core_sources
equilibrium
pf_active

Heating Systems
ec_launchers
ic_antennas
nbi

Metadata
summary
dataset_description
(ids_properties)

Diagnostics
barometry
bolometer
calorimetry
camera_ir
langmuir_probes
magnetics
mse
radiation
spectrometer_mass
thomson_scattering

And many many more...

Strategy

Ideally, all experiments and codes would be developed with IMAS in mind..

IMAS compatibility options:

1. **Update each experiment to output straight to IDSs and adapt physics codes to use IDSs for I/O**

   Time consuming, large amount of effort. Long-term goal?

2. Dynamically create IDSs by mapping data in another format to the IMAS structure

   Strategy used by JET, MAST, WEST, etc.

Strategy

Ideally, all experiments and codes would be developed with IMAS in mind..

IMAS compatibility options:

1. Update each experiment to output straight to IDSs and adapt physics codes to use IDSs for I/O

   Time consuming, large amount of effort. Long-term goal?

2. **Dynamically create IDSs by mapping data in another format to the IMAS structure**

   Strategy used by JET, MAST, TCV, etc.

Previous IDS Mappings

MAST

- Functional but not active maintained
- Minimal signals mapped and not much support to improve
- Mappings in Postgres DB, very dependent on S. Dixon
- Poor performance (hours per ids)

JET

- Extension to the CPO mappings
- Not updated, not complete, not designed to be robust solution
- There has not been interest or support from JET

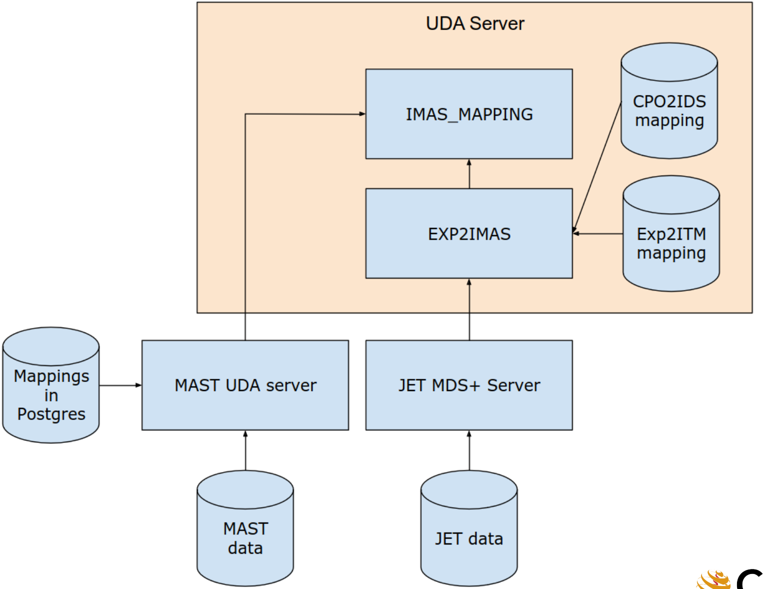
<insert other experiment>

- No doubt has their own strategy

Bottom-line, there is no standard method

👎 duplication of effort

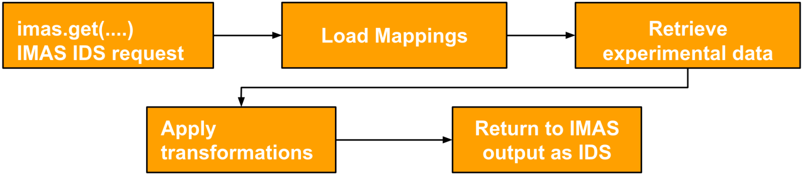
Previous IDS Mappings



MAST-U New Approach (WIP)

Motivation and improvements:

- **Speed** (Previous implementations noted to be slow)
- **RO managed**
  (Mappings must be handled by experts - not computing)
- **Readability** (XMLs or databases can be difficult to read and review)
- **Deployment**
  (Available outside of UKAEA, self-serve)
- **Active development + extendibility**
  (hard to update and keep up with data dictionary advances)



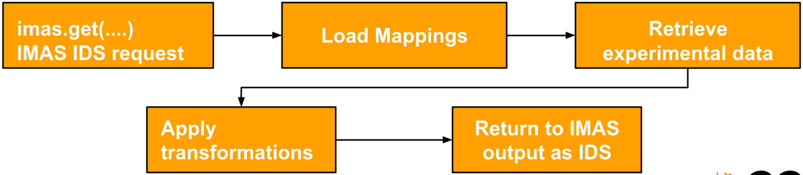
MAST-U New Approach (WIP)

Backend/Plugin (Github Repo)

- Reroutes IDS/IMAS request to mapping plugin
- Loads experiment mappings
- Written in C++ (interface to legacy C)
- Handles experimental data retrieval, type conversion, deserialisation, tree traversal, and data output

Mappings (UKAEA Repo)

- Mappings from IDS to MAST-U signal defined in JSON mapping files per IDS
- Human readable (somewhat)
- Managed by ROs and experts
- Handles multiple DD versions and updates



MAST-U New Approach (WIP)

Syntax/Naming/Helper file

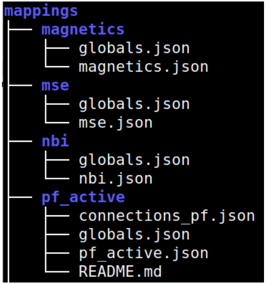
globals.json

+

Mapping file

<ids>.json

Note, will be extended to include multiple DD versions



```
mappings
├── magnetics
│   ├── globals.json
│   └── magnetics.json
├── mse
│   ├── globals.json
│   └── mse.json
├── nbi
│   ├── globals.json
│   └── nbi.json
├── pf_active
│   ├── connections_pf.json
│   ├── globals.json
│   ├── pf_active.json
│   └── README.md
```

Mapping type / transformations: (~90% of cases)

- Value mapping
- Direct unmodified mapping
- Simple operation (+/× by float)
- Slightly more complicated expression

... and then 10% of everything else

JSON Structure - Introduction

```
{
    "PF_COIL_NAMES": [
        "D1U", "D2U", "D3U", "D5U", "D6U", "D7U", "DPU", "P4U",
        "P5U", "P6U", "PXU", "D1L", "D2L", "D3L", "D5L", "D6L",
        "D7L", "DPL", "P4L", "P5L", "P6L", "PXL"
    ],
    "PF_COIL_NAMES_GEOM": [
        "d1_upper", "d2_upper", "d3_upper", "d5_upper", "d6_upper",
        "d7_upper", "dp_upper", "p4_upper", "p5_upper", "p6_upper",
        "px_upper", "d1_lower", "d2_lower", "d3_lower", "d5_lower",
        "d6_lower", "d7_lower", "dp_lower", "p4_lower", "p5_lower",
        "p6_lower", "px_lower", "p1_inner", "p1_outer", "pc"
    ],
    "i_COIL": "{{ at(PF_COIL_NAMES, indices.0) }}"
    "i_COIL_GEOM": "{{ at(PF_COIL_NAMES_GEOM, indices.0) }}"
    "UNIT_SF": 1000,
}
```

Global keys, values, and names unique to each experiment/IDS (– Used in templating and expression evaluation)

Note, **indices** [*array of ints*] is defined by IMAS per request

JSON Structure - Value/Map

```
"pf_active/coil/#/element/#/geometry/geometry_type": {
    "MAP_TYPE": "VALUE",
    "VALUE": 2
    "COMMENT": "Can take many types (3.5, Hello, [3.4, 3.6, 4.3])"
},
"pf_active/coil/#/element/#/geometry/rectangle/width": {
    "MAP_TYPE": "MAP",
    "KEY": "/magnetics/pfcoil/{{ i_COIL_GEOM }}",
    "VAR": "geom_elements.dR",
    "PLUGIN": "GEOMETRY"
},
```

"MAP_TYPE" Keyword describes the type of mapping operation

"VALUE" – IDS path assigned straight from JSON value
"MAP" – MAST-U data retrieval and direct mapping to IDS path

Note, "PLUGIN" enum "base" or "geom" refers to the MAST-U plugin for data access

JSON Structure - Offset/Scale

```
"pf_active/coil/#/current_offset" : {
    "MAP_TYPE": "OFFSET",
    "KEY": "/AMC/ROGEXT/{{ i_COIL }}",
    "OFFSET": 3.0
    "PLUGIN": "UDA",
},
"pf_active/coil/#/current_scale" : {
    "MAP_TYPE": "SCALE",
    "KEY": "/AMC/ROGEXT/{{ i_COIL }}",
    "SCALAR": 2.5
    "PLUGIN": "UDA",
},
```

"OFFSET" – Mirror of "MAP" transformation with the added step of adding "OFFSET" parameter to the data before returning to request

"SCALE" – Multiply data by "SCALAR" parameter before mapping

JSON Structure - Expression

```
"pf_active/coil/#/element/#/area": {
    "MAP_TYPE": "EXPR",
    "PARAMETERS": {
        "WIDTH": "pf_active/coil/#/element/#/geometry/rectangle/width",
        "HEIGHT": "pf_active/coil/#/element/#/geometry/rectangle/height",
    },
    "EXPR": "WIDTH*HEIGHT"
},
"magnetics/b_field_pol_probe/#/poloidal_angle": {
    "MAP_TYPE": "EXPR",
    "PARAMETERS": {
        "Z": "magnetics/_temp/pickup/#/unit_vector/z",
        "R": "magnetics/_temp/pickup/#/unit_vector/r"
    },
    "EXPR": "2*pi-atan(Z/R)"
}
```

"EXPR" – The string in the "EXPR" field is parsed and evaluated before mapping. The "PARAMETERS" dictionary maps the variables for use in the expression, "globals" keys are also available.

Halfway parameters preceded by an "_"

JSON - Dimension/Shape_of

```
"pf_active/coil/Shape_of" : {
    "MAP_TYPE": "VALUE"
    "VALUE": "{{ len(PF_COIL_NAMES) }}"
},
"pf_active/coil/#/element/Shape_of" : {
    "MAP_TYPE": "DIMENSION",
    "DIM_PROBE": "pf_active/coil/#/element/#/geometry/rectangle/z"
},
```

"DIMENSION" – Special case of mapping, where IMAS requires a successful Shape_of request for array structures.
However, for MAST-U data is structured differently so you need to retrieve a signal to then calculate the size.

In this case for example, IMAS is requesting the number of elements for this coil.
"DIM_PROBE" is the element z position which returns a vector from MAST-U

From there the size can be returned ⟶ Shape_of

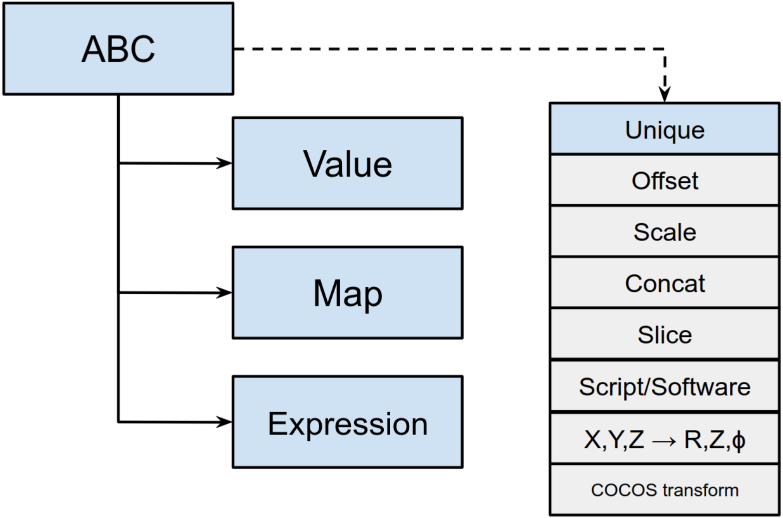
JSON Structure - Function Library

Still a work in progress...

```
"pf_active/circuit/#/connections" : {
    "MAP_TYPE": "CONNECTIONS",
    "...": "..."
},
"mse/channel/#/polarisation_angle" : {
    "MAP_TYPE": "SLICE",
    "KEY": "AMS/GAMMA/POLARISATION_ANGLE"
},
"ids_name/angle/phi" : {
    "MAP_TYPE": "COCOS",
    "KEY": "mse/channel/#/polarisation_angle",
    "FROM": 11,
    "TO": 17
},
```

Abstract mapping base class gives the freedom to define a library of helper functions as separate mapping types if they're used often and as they are needed

JSON Structure



Dependencies

Nlohmann JSON for Modern C++

Documentation Repository

Inja - Template engine for modern C++, loosely inspired by jinja

Documentation Repository

ExprTk - The C++ Mathematical Expression Toolkit Library

Documentation Repository

GSL - Guidelines Support Library for C++98, C++11 above

Many implementations, Microsoft/GSL or header-only gsl-lite

Boost - Boost C++ Libraries

Documentation

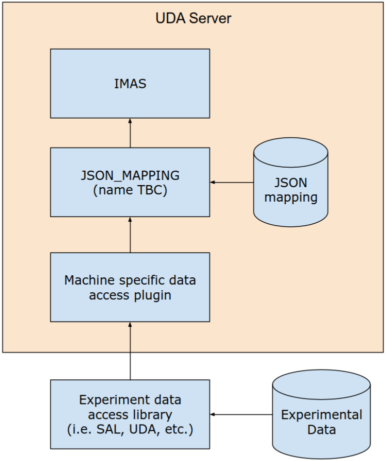
Deployment

(Relatively)
Easy to install at site:

- IMAS
- UDA
- JSON mapping plugin
- JSON mapping files

```
> import imas
> mastu = imas.DBEntry(imas.imasdef.UDA_BACKEND,
          'MAST-U', 47000, 0, 'adam')
> mastu.open()
> ids = mastu.get("wall")
```

LIVE DEMO


UDA Server
IMAS
JSON_MAPPING (name TBC)
JSON mapping
Machine specific data access plugin
Experiment data access library (i.e. SAL, UDA, etc.)
Experimental Data

Management

Mapping repository can be private and local to the experiment
If made public, easier to support but not necessary

CI JSON validation tests:

- Valid JSON check
- Schema validation
- Template completion
- Expression evaluation

Future test implementation:

- Mapping of each IDS for *Standard Candle* shot
- Verify important IDS fields present (discussed last week)

Documentation on mappings hosted on repository GitPages
Skeleton templates can be easily provided

MAST-U Mapping Status

🟥 Unmapped	🟨 Process started	🟩 Mapped and *'available'* Requires validation	🟦 Deployed and fully available

*as of Data Dictionary v3.38.1

Interface Data Structure (IDS)	Status*	Notes and Comments
magnetics	🟩	
pf_active	🟩	
pf_passive	🟩	
wall	🟩	Essential signals, e.g., limiter coords.
mse	🟨	Contacted ROs, initial mappings started
nbi	🟨	Contacted ROs, initial mappings started
thomson_scattering	🟨	Contacted ROs, initial mappings started
pulse_schedule	🟨	Contacted ROs, initial mappings started
summary	🟥	Required by UKAEA Computing Division

Timeline + Demand

Task began as a **nice to have**

→ Several ITER IA tasks now depend on MAST-U mappings

→ EuroFusion/F4F support for imasification

→ Contacted by collaborators in Italy and Finland asking for MAST-U IDSs

However, difficulties in acquiring support from experts has caused delays

Timeline:

- Initial release of plugin - end of June
- Write white paper following months
- Training event planned at ITER October/November

Development Plan:

- Performance improvements
- Access layer 5
- Map other IDSs for MAST-U

Designed to be experiment agnostic

Supported by ITER:

- Mentioned in submitted IAEA TM abstract (S. Pinches)

- IMAS AL meetings: “shall form the basis for a general mapping plugin”

Summary

- Initial IMAS mapping infrastructure in place for MAST-U
- Mapping of IDSs for MAST-U is ongoing
- If interested in using the tool, happy to support
- Contact : adam.parker@ukaea.uk or Teams

Questions?