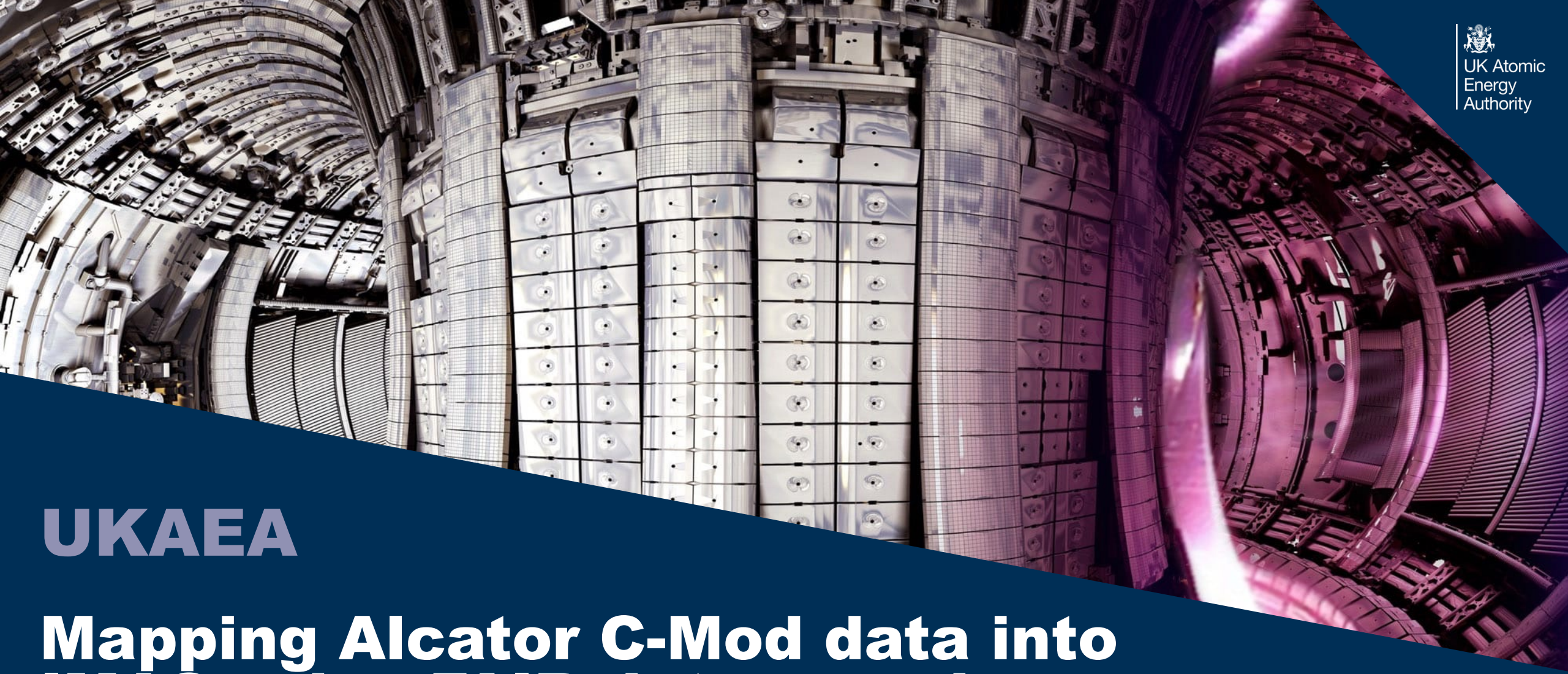


UK Atomic Energy Authority

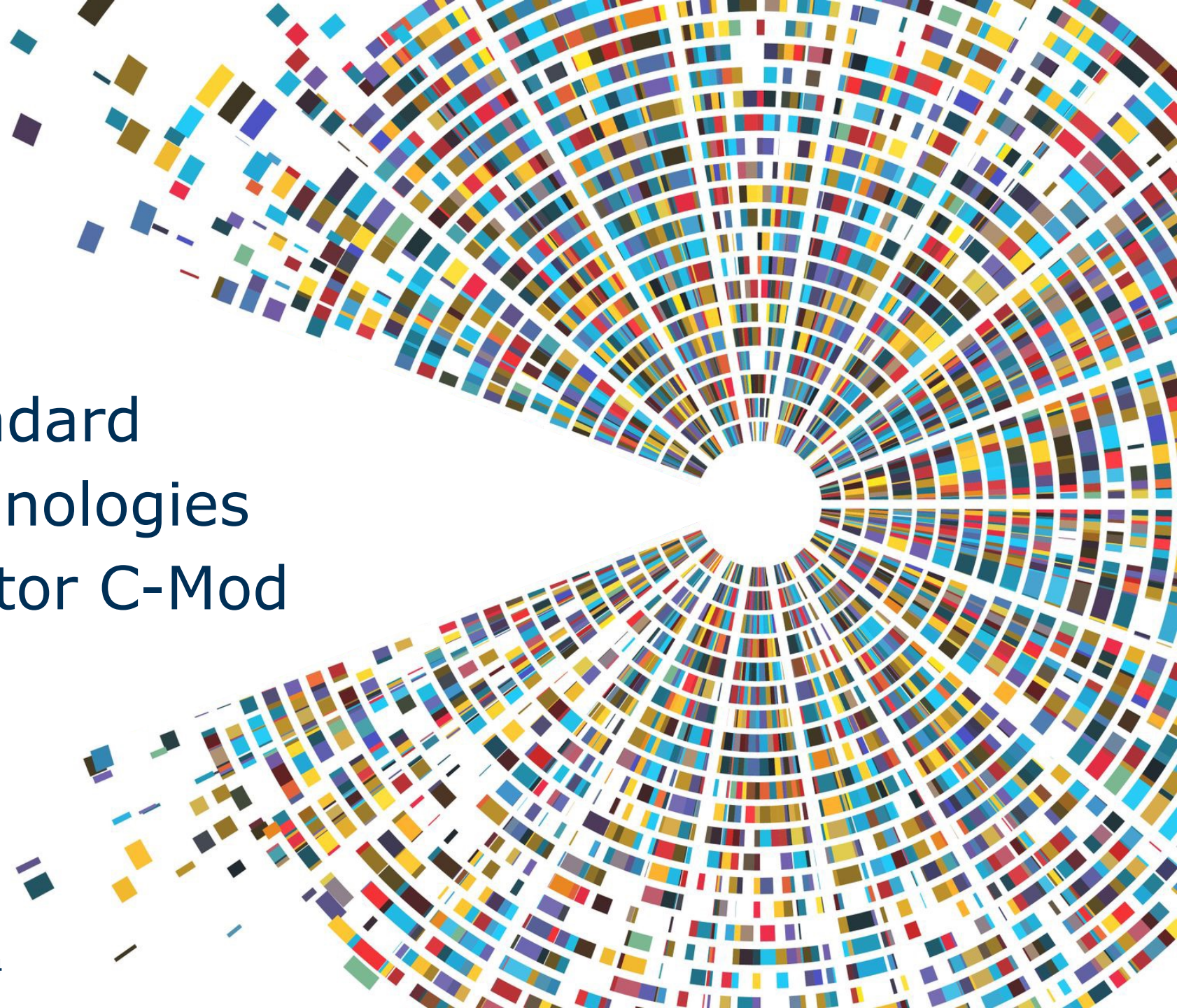
UKAEA

Mapping Alcator C-Mod data into IMAS using FAIR data mappings

J. Hollocombe (UKAEA), A. Parker (UKAEA)

IAEA workshop, Sao Paulo, Brazil, 17th July 2024

Contents



1. Motivation
2. Mapping Standard
3. Mapping Technologies
4. Mapping Alcator C-Mod
5. Future Work

UK Atomic Energy Authority

1. Motivation

Motivation

UK Atomic Energy Authority

Data

- Lots of existing experimental data – untapped resource
- Common data model exists in IMAS data dictionary

Mappings

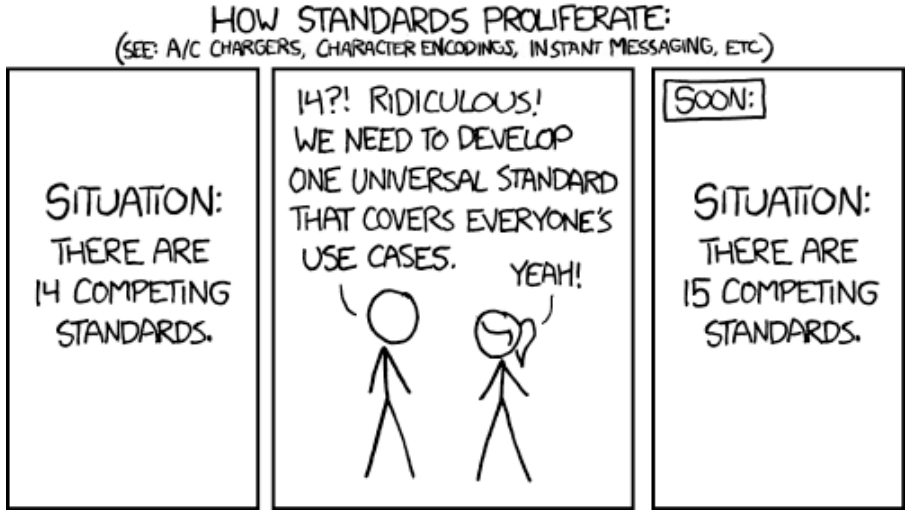
- Many ways of mapping the data – technologies & representation
- Off-line vs on-line mapping – different trade-offs
- Only want to write these mappings once – people's time is precious
- Interoperable/sharable mappings would allow common tools to be developed

Create a Mapping Standard

UK Atomic Energy Authority

- Create a mapping standard like the Common Workflow Language† for workflows
- Using a standard human maintainable format – able to be updated and maintained by non-software engineers
- Ability to run validity checks – in development and CI
- Should allow for description fields to be added to mapping to enable self-documentation
- Encode as much as possible in the mapping but being pragmatic

† https://www.commonwl.org/



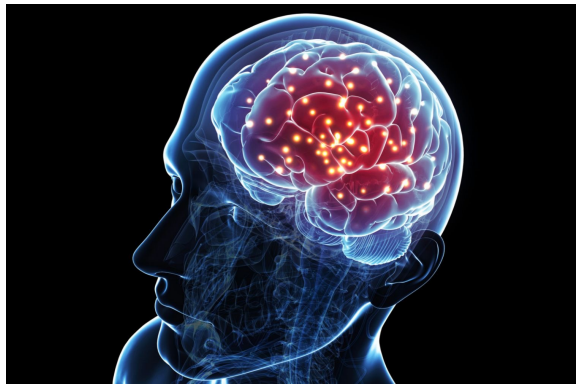
UK Atomic Energy Authority

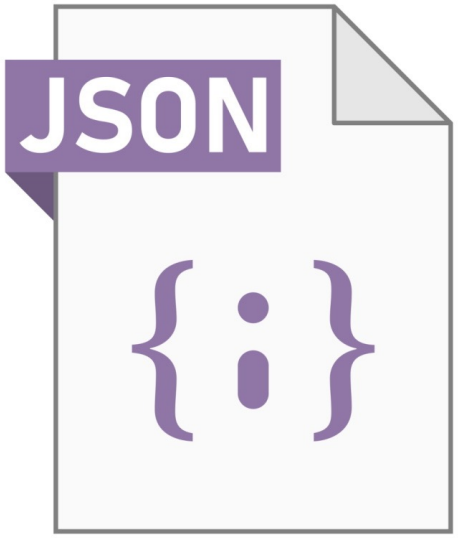
2. Mapping Standard (TokaMap)

TokaMap mapping syntax

UK Atomic Energy Authority

- Devised a JSON based mapping syntax
- Designed to be flexible, template-able and extensible
- Can be used to populate an IDS on request using the UDA IMAS plugin or used in a mapping tool such as pytokamap to generate the mapped data off-line
- Capture all that precious machine knowledge





IMAS mappings

```
Data dictionary
- amns_data
- b_field_non_axisymmetric
- barometry
- …
- waves
- workflow
```

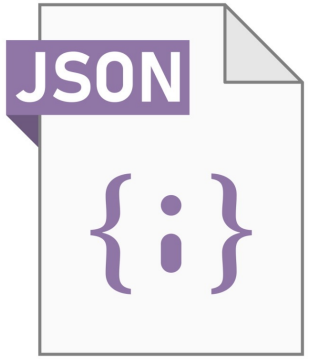
magnetics/mappings.json

Nested structures

Data fields

Array of structures

```
magnetics IDS
- ids_properties
    - comment
    - field
    - …
- flux_loop(N)
    - name
    - flux/data(:)
    - …
- …
```

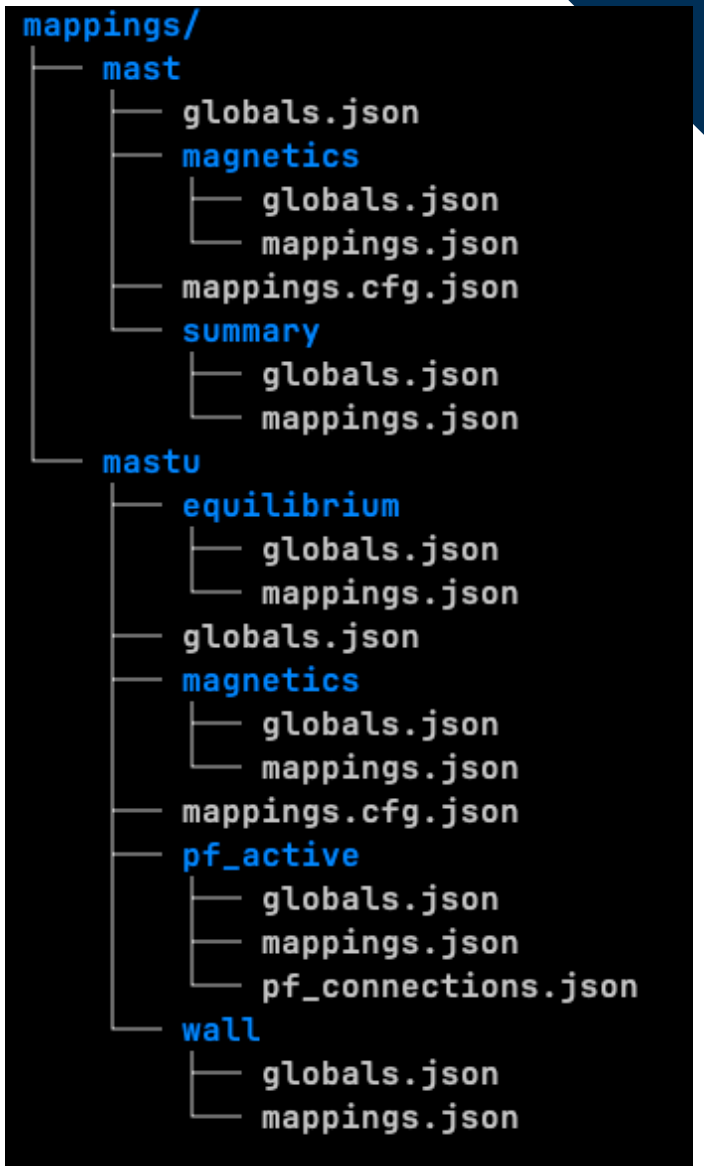



```
“flux_loop[3]/flux/data” : {
    “MAP_TYPE”: “…”
}
```

Mapping Files: Directory Structure

UK Atomic Energy Authority

- Inside mappings directory we have a folder for each machine
- Top level “globals.json” contains common data for all IDSs
- Top level “mappings.cfg.json” specifies which IDSs have been mapped for specific DD versions
- Each IDS is in its own folder
  - “globals.json” contains common data
  - “mappings.json” contains actual mappings
- Work being done to split mappings by DD version and shot range to allow for more granularity



Mapping Files: globals.json example

UK Atomic Energy Authority

```
{
    ...
   "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"
   ],
   "COIL_NAMES": [
      "D1U", "D2U", "D3U", "D5U", "D6U", "D7U", "DPU",
      "P5U", "P6U", "PXU", "D1L", "D2L", "D3L", "D5L",
      "D7L", "DPL", "P4L", "P5L", "P6L", "PXL"
   ],
   "UNIT_SF": 1000.0,
   ...
}
```

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

MAST-U real example
COIL_NAMES used to access signal :
/AMC/FLUX_LOOPS/D1U

MAST-U currents are also in kAmps so are scaled by **UNIT_SF** for the IDS

Mapping Files: Map Types

UK Atomic Energy Authority

- **VALUE**
  - Hard-coded value in mapping file
  - Good for static machine description, etc.
- **DIMENSION**
  - Reading the dimension of another mapping
  - Good for mapping lengths of array structures, i.e. flux_loops(N)
- **PLUGIN** (may rename)
  - Call out to a data reader
  - Most common map type
  - Used to fetch actual data from file or data system, i.e. UDA/MDS+
  - Has arguments for scaling, offsetting & slicing returned data

Mapping Files: Map Types

UK Atomic Energy Authority

- **EXPR**
  - Uses powerful expression toolkit to perform more complex mappings
- **CUSTOM**
  - Small set of additional useful transformations that can be used in mappings
  - E.g. COCOS transformation, array concatenation
  - Will be extended as needed but each library that supports the mappings would have to implement funcitonality

```
"b_field_pol_probe[#]/poloidal_angle": {
    "MAP_TYPE": "EXPR",
    "PARAMETERS": {
        "Z": "_pickup[#]/unit_vector/Z",
        "R": "_pickup[#]/unit_vector/R"
    },
    "EXPR": "2*PI-atan2(Z,R)"
},
"_pickup[#]/unit_vector/Z": {
   ...
},
"_pickup[#]/unit_vector/R": {
   ...
},
```

Dynamic Templating

We support dynamic templating in the mapping files using a syntax based on the **Jinja** library for python. In C++ we use the **Inja** library which can perform similar templating.

Simple Example

```
name = "World";
post_inja_string = inja::render("Hello {{ name }}!", name);
-> "Hello World!"
```

We will support a subset of templating functions supported by both **Inja** and **Jinja**

Used extensively in MAST-U Mappings

Able to dynamically substitute strings

JSON Mapping Example

```
"SPECIAL_COIL": "P1",
"COIL_NAMES": ["D1U", "D2U", "D3U"],
...
"signal": "/AMC/PF_COIL/{{ SPECIAL_COIL }}",
-> "/AMC/PF_COIL/P1"
…
"signal_2": "/AMC/PF_COIL/{{ at(COIL_NAMES, 2) }}"
-> "/AMC/PF_COIL/D3U"

"VALUE": "{{ length(COIL_NAMES) }}"
-> "3" -> can be converted to Integer
```

Mapping Indices

UK Atomic Energy Authority

- To avoid mapping duplication indices are extracted from array structures
  - I.e. if “magnetics/flux_loop[3]/flux/data” is not found in mapping file then look for “magnetics/flux_loop[#]/flux/data”
  - Extracted indices are available in mappings as magic “indices” template variable

```
"FL_TYPE": "{{ at(FLUX_LOOPS, indices.0).TYPE }}"
```

```
"flux_loop[#]/identifier": {
    "MAP_TYPE": "VALUE",
    "VALUE": "FLUX_LOOP_{{ indices.0 + 1 }}"
},
"flux_loop[#]/type/name": {
    "MAP_TYPE": "VALUE",
    "VALUE": "{{ FL_TYPE }}"
},
"flux_loop[#]/type/index": {
    "MAP_TYPE": "VALUE",
    "VALUE": "{{ at(FL_TYPE_MAP, at(FLUX_LOOPS, indices.0).TYPE) }}"
},
```

Hidden Mappings and Comments

UK Atomic Energy Authority

Naming convention

Not every entry within the mapping file must be an IDS path

Any entry can be used for intermediate calculations or *'turned off'*

Our naming convention is to prefix keys by an underscore '_' for non-IDS mappings

Comments

JSON by default does not allow comments

Within the mapping file, each map type is allowed a COMMENT field.

This is purely for the person making the mappings, the field is ignored when the mappings are ingested

```
{
    "_b_field_pol_probe[#]/poloidal_angle": {
        "MAP_TYPE": "EXPR",
        "PARAMETERS": {
            "Z": "_pickup[#]/unit_vector/Z",
            "R": "_pickup[#]/unit_vector/R"
        },
        "EXPR": "2*PI-atan2(Z,R)"
    },
    "_sub_mapping": {
        "MAP_TYPE": "PLUGIN",
        "PLUGIN": "UDA",
        "ARGS": {
            signal": "/AMC/PLASMA_CURRENT"
        }
    },
    "_pickup[#]/unit_vector/R": {
        "MAP_TYPE": "VALUE",
        "VALUE": 23.4,
        "COMMENT": "This is a random comment"
    },
}
```

Mapping examples

UK Atomic Energy Authority

See MAST-U mappings (https://github.com/ukaea/IMAS_MASTU_mappings/) for many more examples of how fields can be mapped

IDS: pf_active

```
"coil[#]/element[#]/geometry/rectangle/width": {
    "MAP_TYPE": "PLUGIN",
    "PLUGIN": "GEOMETRY",
    "ARGS": {
        "signal": "/magnetics/pfcoil/{{ PF_GEOM_NAME }}",
        "key": "geom_elements.dR"
    },
    "SLICE": "[{{ indices.1 }}]"
},
```

IDS: tf

```
"b_field_tor_vacuum_r": {
    "MAP_TYPE": "PLUGIN",
    "PLUGIN": "UDA",
    "ARGS": {
        "signal": "/AMC/ROGEXT/TF"
    },
    "SCALE": 0.0048,
    "COMMENT": "From Ampere's law at r=1metre, takes into account
the number of TF coils"
}
```

Mapping Schema and Validation

UK Atomic Energy Authority

Testing and validation step available through pytest (and CI)

1) **JSON Verification**
   Test that all JSON globals.json and mappings.json are valid JSON syntax and can be read / parsed

2) **JSON Schema Validation**
   Validate all mappings.json against the top-level schema defining map types and required fields

3) **Inja Templating Render**
   Using globals.json, verify that all Inja templating strings complete and render to a valid string

4) **EXPR Map Type Validation**
   Test EXPR strings, substitute random floats to the parameters, check the output is sensible and of float type

Mapping Schema and Validation

```
tests/test_schema.py::test_valid_globals[map_globals7] PASSED
tests/test_schema.py::test_valid_globals[map_globals8] PASSED
tests/test_schema.py::test_valid_globals[map_globals9] PASSED
tests/test_schema.py::test_valid_structure[map_schema0] FAILED
tests/test_schema.py::test_valid_structure[map_schema1] PASSED
tests/test_schema.py::test_valid_structure[map_schema2] PASSED
tests/test_schema.py::test_valid_structure[map_schema3] PASSED
tests/test_schema.py::test_valid_structure[map_schema4] PASSED
tests/test_schema.py::test_valid_structure[map_schema5] PASSED
tests/test_schema.py::test_valid_structure[map_schema6] PASSED
tests/test_schema.py::test_valid_structure[map_schema7] PASSED
tests/test_schema.py::test_valid_structure[map_schema8] PASSED
tests/test_schema.py::test_valid_structure[map_schema9] PASSED
tests/test_templating_inja.py::test_cython_include PASSED
tests/test_templating_inja.py::test_template_syntax_fail[/magnetics/pfcoil/{{indices.1/test-expected0] PASSED
tests/test_templating_inja.py::test_template_syntax_fail[/magnetics/pfcoil/{{unknown}}/test-expected1] PASSED
tests/test_templating_inja.py::test_templating_in_json[get_keys_and_globals0] FAILED
tests/test_templating_inja.py::test_templating_in_json[get_keys_and_globals1] PASSED
tests/test_templating_inja.py::test_templating_in_json[get_keys_and_globals2] PASSED
tests/test_templating_inja.py::test_templating_in_json[get_keys_and_globals3] PASSED
tests/test_templating_inja.py::test_templating_in_json[get_keys_and_globals4] PASSED
tests/test_templating_inja.py::test_templating_in_json[get_keys_and_globals5] PASSED
tests/test_templating_inja.py::test_templating_in_json[get_keys_and_globals6] PASSED
tests/test_templating_inja.py::test_templating_in_json[get_keys_and_globals7] PASSED
tests/test_templating_inja.py::test_templating_in_json[get_keys_and_globals8] PASSED
tests/test_templating_inja.py::test_templating_in_json[get_keys_and_globals9] PASSED
tests/test_validator.py::test_json_open PASSED
tests/test_validator.py::test_json_open_fail PASSED
tests/test_validator.py::test_json_schema_pass PASSED
tests/test_validator.py::test_json_schema_fail PASSED
tests/test_validator.py::test_json_schema_comp PASSED
```

UK Atomic Energy Authority

3. Mapping Technologies

Mapping technologies

UK Atomic Energy Authority

pytokamap

- Work in progress library developed by Samuel Jackson
- Using the JSON mappings to transform MAST data
- https://pypi.org/project/pytokamap/ (warning: alpha code)
- Plans to develop this further to make more generic – maybe integrate with OMAS

UDA IMAS plugin

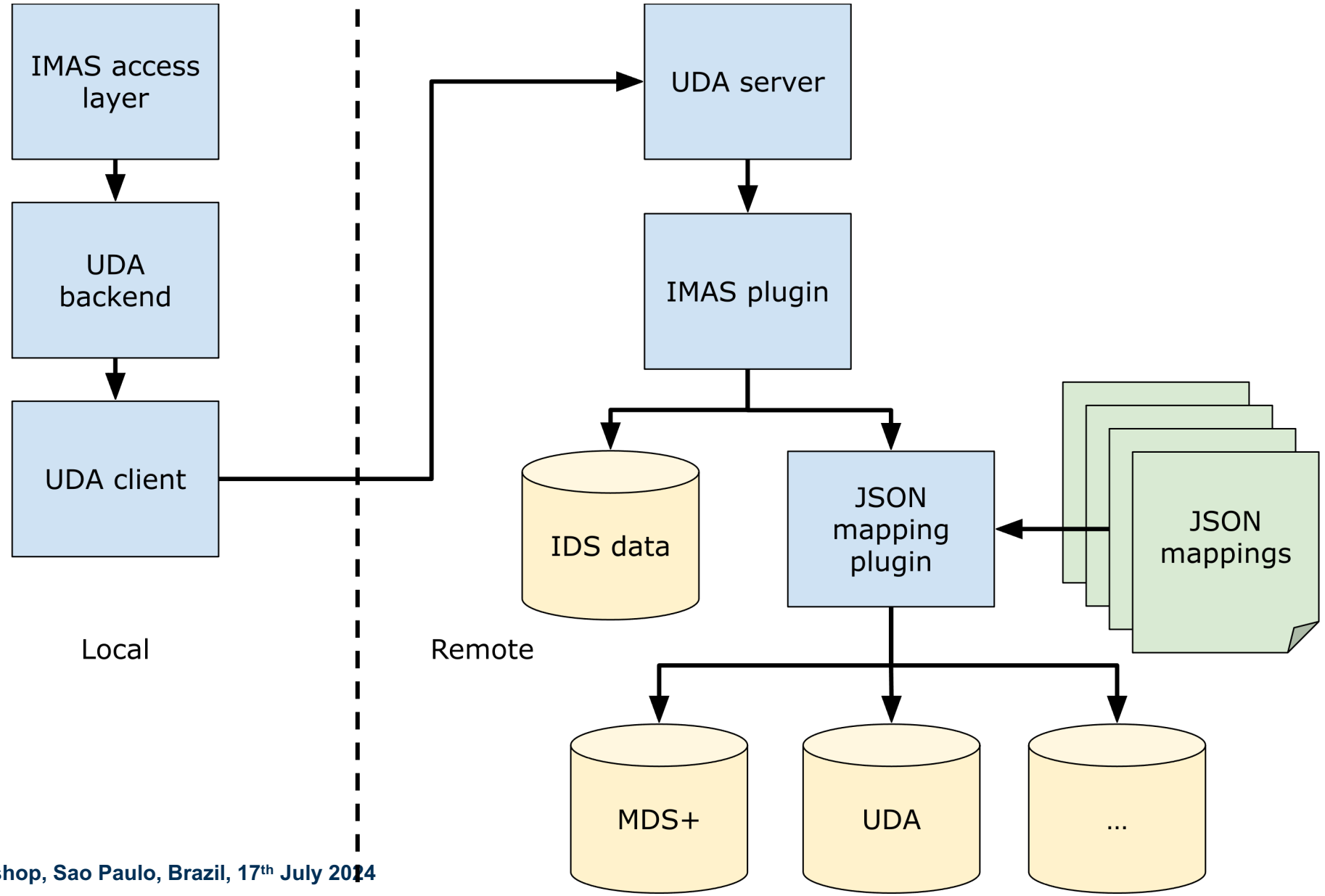
- On-line mappings
- Utilising existing IMAS remote data functionality (new in AL5)
  - "imas://server:port/uda?path=/path/to/ids/data"
- "imas://server:port/uda?mapping=machine?machine_args"

IMAS UDA stack

UK Atomic Energy Authority

- Makes use of existing remote data fetching capabilities of IMAS (AL5)
- The IMAS plugin can handle requests with “path=…” in the URI as normal but requests with URI containing “mapping=…” get forwarded to a mapping plugin
- A new mapping plugin has been created to handle the JSON mapping files
- Requests for data are handled using existing (or new) data access plugins, e.g.
  - For MAST/MAST-U a UDA plugin can read the data from the MAST UDA server
  - For C-Mod a (new) MDSplus plugin can read the data from the C-Mod MDSplus server
  - For NetCDF/HDF5 files the file reader plugins can be used
  - Bridge plugins for other data systems/files can be created as required

IMAS UDA stack



pytokamap

UK Atomic Energy Authority

- Tool created by Sam Jackson to use the TokaMap mappings
- https://github.com/ukaea/pytokamap
- Python tool which reads MAST data from MAST UDA data system, maps and then saves to zarr or NetCDF format

```
import pytokamap
mapper = pytokamap.load_mapping("uda.jinja", "globals.json")
mapper.to_zarr(30420, "30420.zarr")
mapper.to_netcdf(30420, "30420.nc")
```

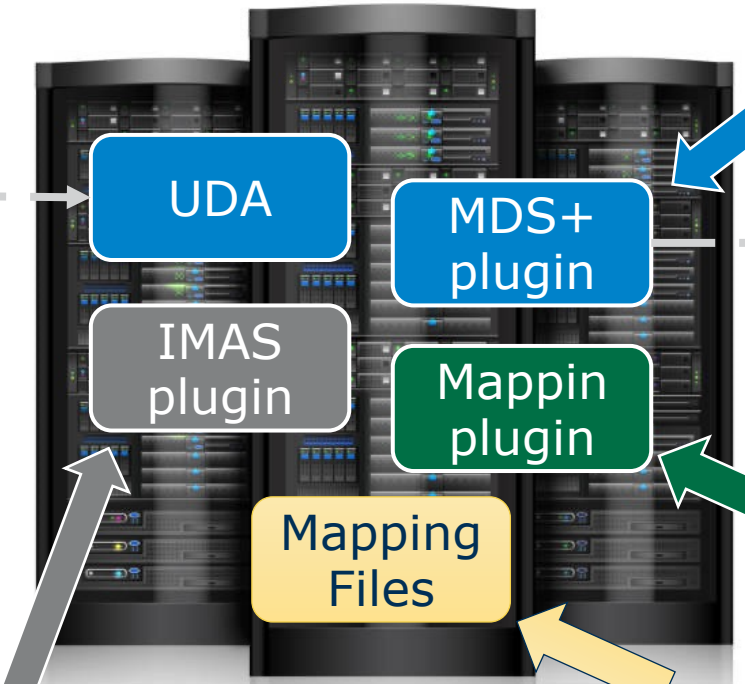
- WIP package with plans on extending – see later
- Available on pypi but in very alpha state

UK Atomic Energy Authority

4. Mapping Alcator C-Mod

Installation at MIT

Thanks to Stephen Lane-Walsh for his help with setting up mappings at MIT



IMAS

https://github.com/ukaea/uda

C-Mod MDS+ Server

https://github.com/ukaea/JSON-mapping-plugin

https://github.com/MIT-PSFC/IMAS-CMod-mappings

https://git.iter.org/projects/IMAS/repos/uda-plugins/browse

Installation at MIT

UK Atomic Energy Authority

```
# Install UDA
cmake -B build-release -DCMAKE_INSTALL_PREFIX=../install-release -DBUILD_SHARED_LIBS=ON -DCMAKE_BUILD_TYPE=Release
cmake --build build-release/ -j
cmake --install build-release/

# Install IMAS plugins
export PKG_CONFIG_PATH=$HOME/UDA/install-release/lib/pkgconfig
cmake -B build-release -DCMAKE_INSTALL_PREFIX=../install-release -DBUILD_PLUGINS=imas -DCMAKE_BUILD_TYPE=Release
cmake --build build-release/ -j
cmake --install build-release/
./build-release/scripts/activate-plugins.sh

# Install IMAS mapping plugin
cmake -B build-release -DCMAKE_INSTALL_PREFIX=../install-release -DCMAKE_BUILD_TYPE=Release
cmake --build build-release/ -j
cmake --install build-release/
./build-release/scripts/activate-plugins.sh
```

MIT mappings

UK Atomic Energy Authority

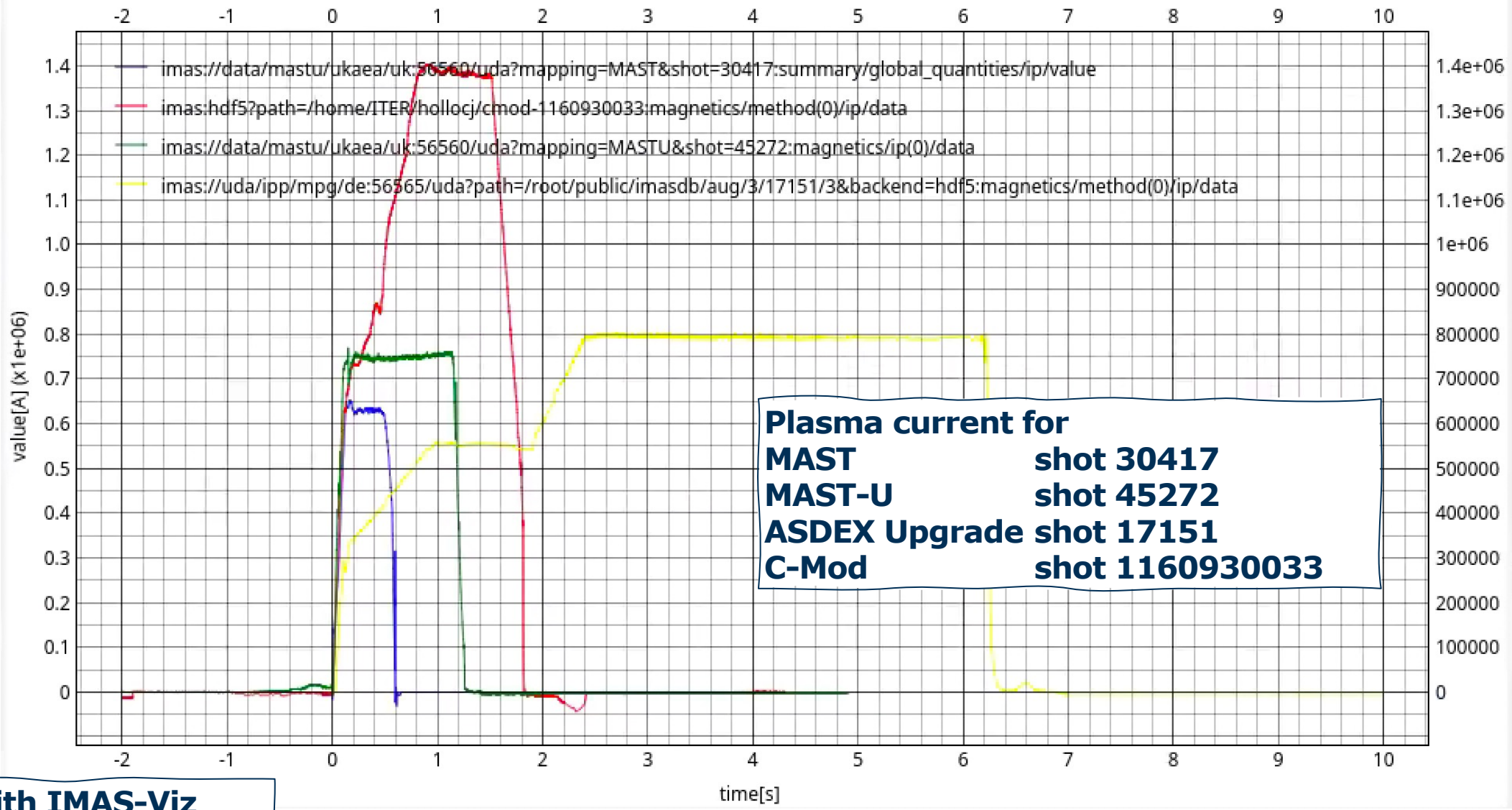
- Initial development of mappings for Alcator C-Mod
  - Available at https://github.com/MIT-PSFC/IMAS-CMod-mappings

- Developed a MDSPLUS plugin to read from C-Mod MDSplus data system

- Only a limited set of signals have been mapped so far – hope to provide ongoing support for more mappings!

```
"globals_quantities/ip/value": {
    "MAP_TYPE": "PLUGIN",
    "PLUGIN": "MDSPLUS",
    "ARGS": {
        "expression": "\\ip"
    }
},
```

Results



Plotting data with IMAS-Viz – MAST, MAST-U, C-Mod & AUG

***WIP mappings – COCOS for C-Mod data not validated**

UK Atomic Energy Authority

5. Future Work

Future Work

- Plans on making the mappings themselves FAIR
  - Make sure the mappings are self describing
  - Ensure the mappings can be used in multiple tools
  - Provide tools for maintaining & finding mappings
  - Building on Simple Standard for Sharing Ontological Mappings (SSSOM)† work
- OSCARS proposal – hope to hear about this soon
- Extend pytokamap – potential ideas:
  - Integrate with OMAS to make use of its ability to read and write multiple formats
  - Ensure compatibility between mappings used in IMAS plugin and pytokamap
- Extend mapping syntax and functionality as needed

†https://mapping-commons.github.io/sssom/

UK Atomic Energy Authority

Questions?

Thank you for your time