UKAEA

IMAS data mapping with libTokaMap

Jonathan HOLLOCOMBE, Adam PARKER, Stephen DIXON

1st ITER Mapping Workshop – March 2026

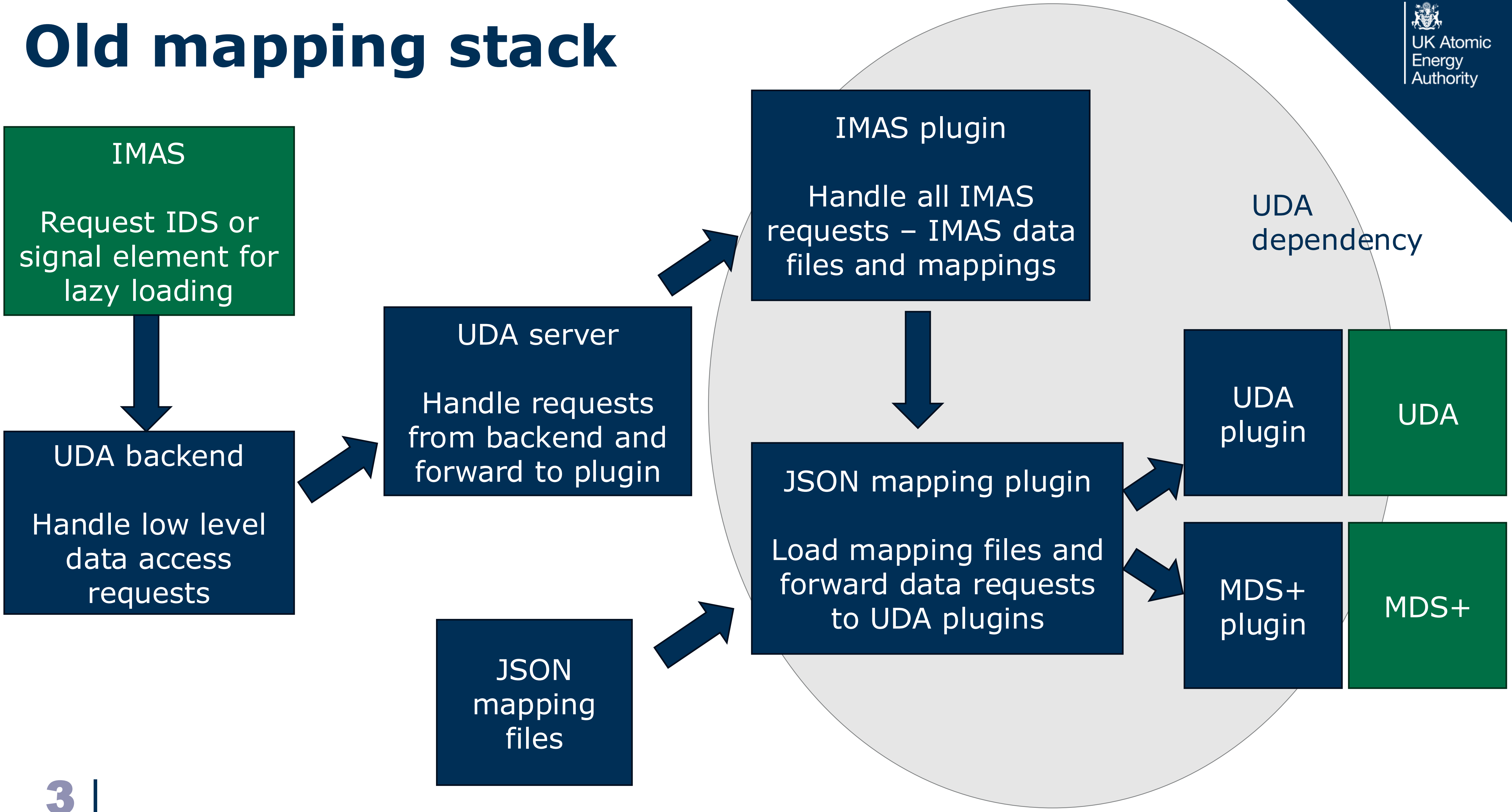
Introduction

Follow on from Adam’s discussion of TokaMap mapping syntax.

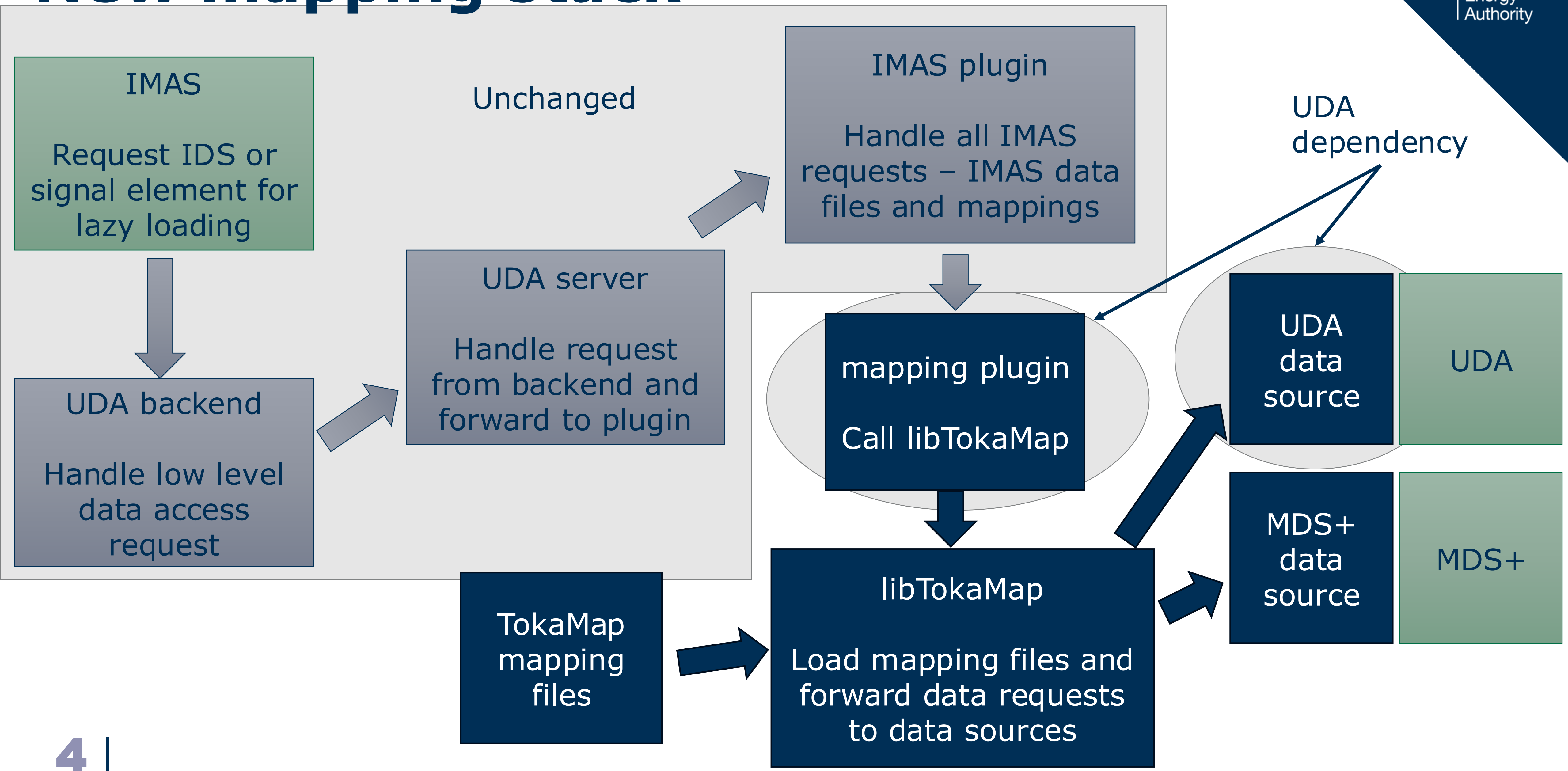
See https://ukaea.github.io/tokamap/develop/ for details on schemas

- Discussion of changes to mapping stack in IMAS
- libTokaMap and it’s features
- Demo showing how it should be easier to test mappings (hopefully)

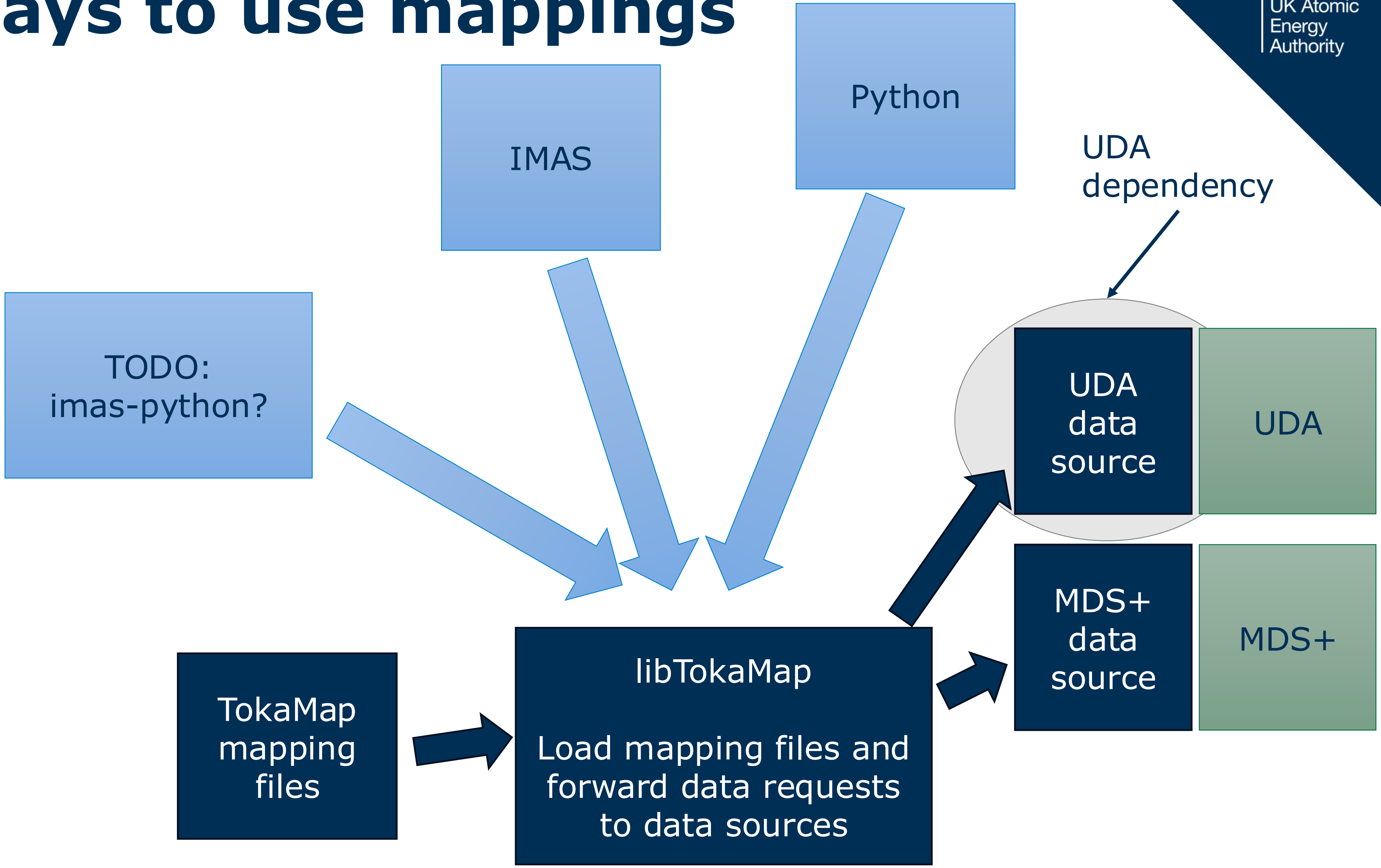
Old mapping stack


IMAS
Request IDS or signal element for lazy loading
UDA backend
Handle low level data access requests
UDA server
Handle requests from backend and forward to plugin
JSON mapping files
IMAS plugin
Handle all IMAS requests – IMAS data files and mappings
JSON mapping plugin
Load mapping files and forward data requests to UDA plugins
UDA dependency
UDA plugin
UDA
MDS+ plugin
MDS+

New mapping stack


IMAS
Request IDS or signal element for lazy loading
Unchanged
IMAS plugin
Handle all IMAS requests – IMAS data files and mappings
UDA dependency
UDA server
Handle request from backend and forward to plugin
UDA backend
Handle low level data access request
mapping plugin
Call libTokaMap
UDA data source
UDA
MDS+ data source
MDS+
TokaMap mapping files
libTokaMap
Load mapping files and forward data requests to data sources

Other ways to use mappings



libTokaMap – features

- Reference implementation of the TokaMap schemas
- C++20 library designed to be flexible and performant
- Extensible data sources that can be discovered and loaded at runtime
- Built-in RAM caching for improved performance
- Schema validation
- TOML/JSON configuration
- Pip installable Python wrapper
- Provenance ‘trace’ of all transformations used to generated data

libTokaMap – availability

UK Atomic Energy Authority

- Linux packages, tarball and zip available on releases:
  https://github.com/ukaea/libtokamap/releases/tag/0.2.3

- Available on macOS via Homebrew:
  brew install jholloc/formulae/libtokamap

- Pip install from PyPI (and testPyPi for dev releases):
  pip install libtokamap

- Pip install dev versions from testPyPI:
  pip install -i https://test.pypi.org/simple --extra-index-url https://pypi.org/simple libtokamap

Basic use

```
libtokamap::MappingHandler mapping_handler;

std::filesystem::path config_path = "./config.toml";
mapping_handler.init(config_path);

std::string path = "/path/to/map";
libtokamap::DataType data_type = libtokamap::DataType::Float;
int rank = 1;
nlohmann::json context = {{"shot", 42}};

auto result = mapping_handler.map(mapping, path, data_type, rank, context);
```

Config

```
mapping_directory = "/path/to/mappings"
schemas_directory = "/path/to/tokamap/schemas"
trace_enabled = true
cache_enabled = true
custom_function_libraries = [ "/path/to/custom/function/library.so" ]

[data_source_factories]
my_factory = "/path/to/data/source/library.so"

[data_sources.MySource]
factory = "my_factory"
args.factory_arg1 = "value1"
args.factory_arg2 = "value2"
```

Library with "C" API entry function to register factory function

Name that can be used in mappings

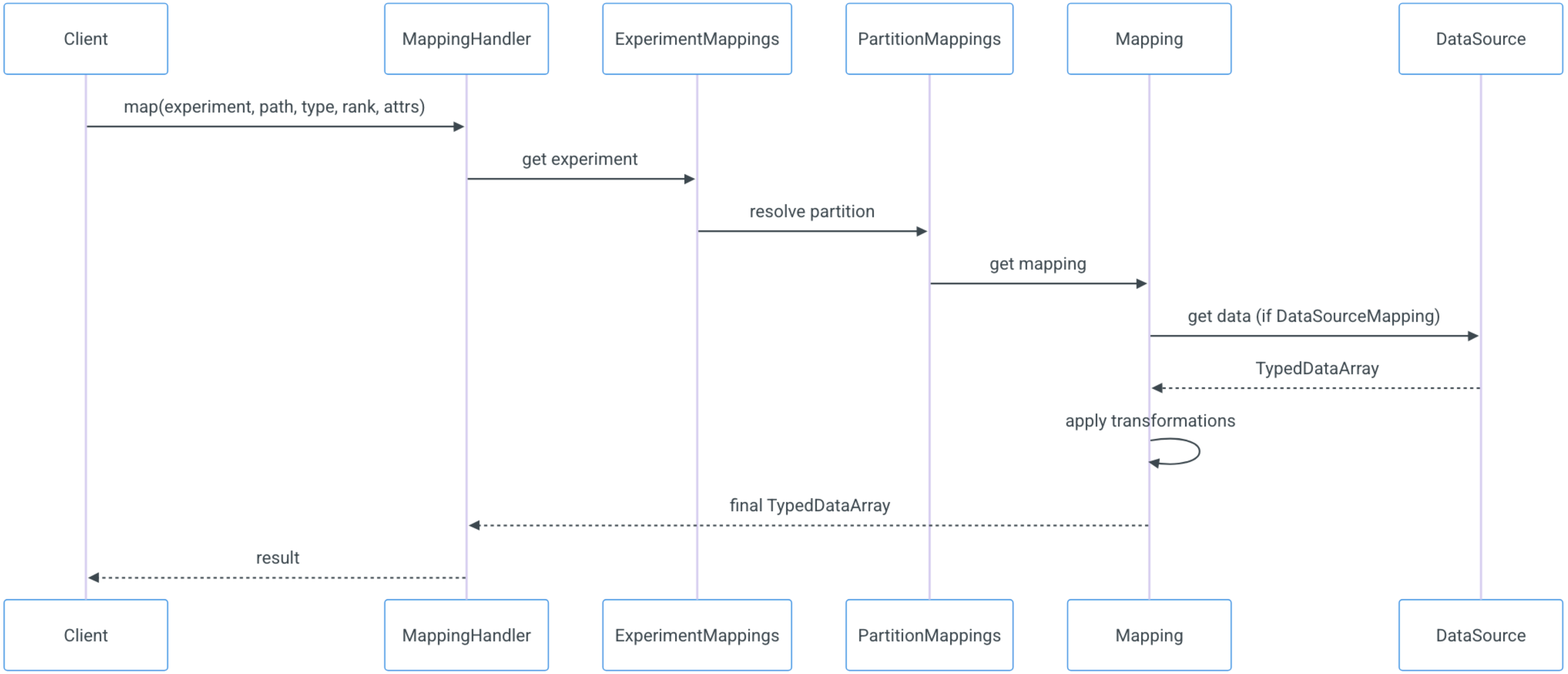
0+ arguments to customise instance of data source

Mappings

```
mappings
└── example_v1
    ├── globals.json
    ├── magnetics
    |   └── 40
    |       ├── globals.json
    |       └── mappings.json
    └── mappings.cfg.json
```

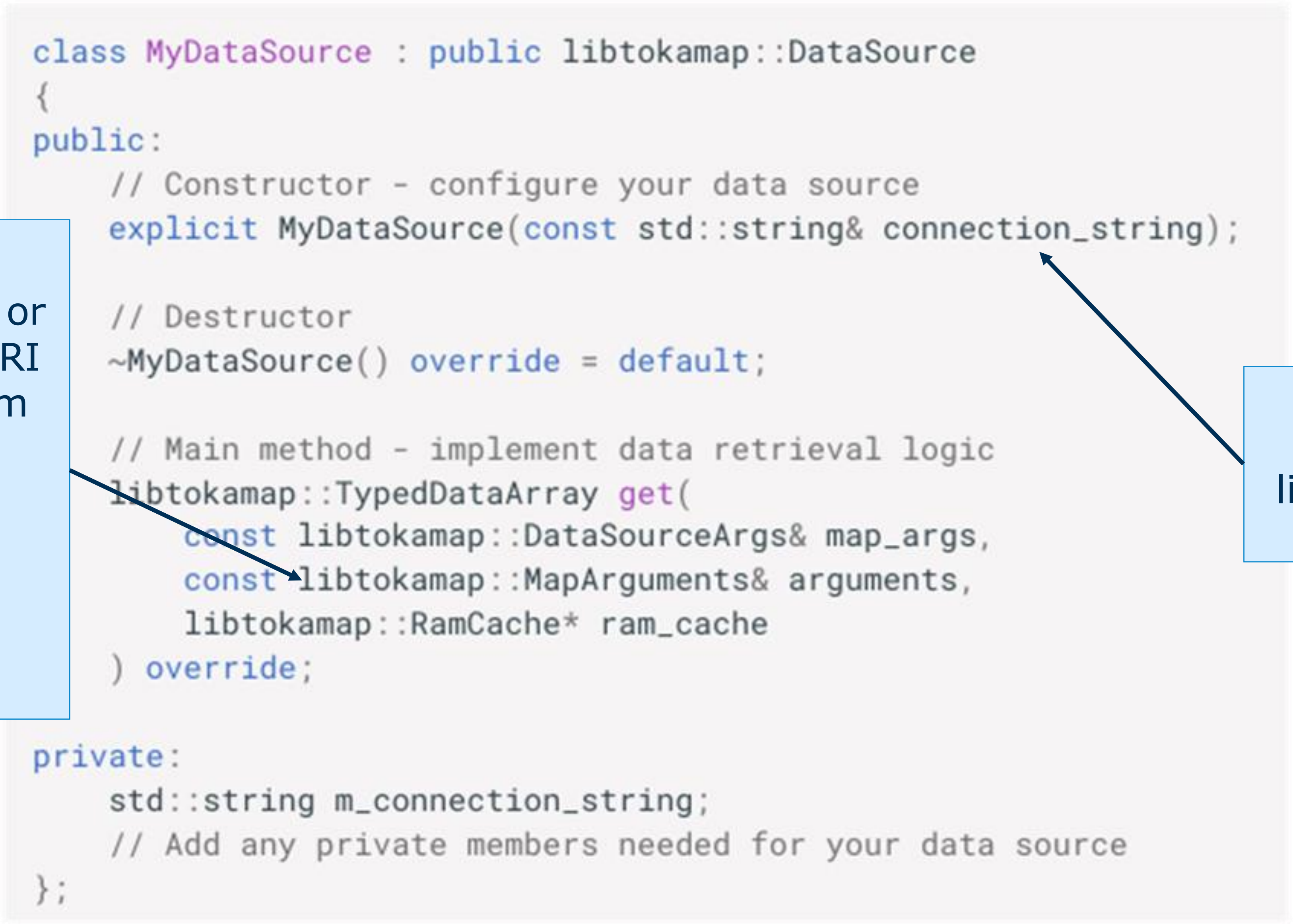
```
{
  "metadata": {
    "experiment": "example",
    "author": "John Doe",
    "version": "1.0.1"
  },
  "groups": ["magnetics"],
  "partitions": [
    {
      "attribute": "shot",
      "selector": "max_below”
    }
  ]
}
```

Mapping workflow



Data Sources

UK Atomic Energy Authority



```cpp
class MyDataSource : public libtokamap::DataSource
{
public:
    // Constructor - configure your data source
    explicit MyDataSource(const std::string& connection_string);

    // Destructor
    ~MyDataSource() override = default;

    // Main method - implement data retrieval logic
    libtokamap::TypedDataArray get(
        const libtokamap::DataSourceArgs& map_args,
        const libtokamap::MapArguments& arguments,
        libtokamap::RamCache* ram_cache
    ) override;

private:
    std::string m_connection_string;
    // Add any private members needed for your data source
};
```

Passed in via mappings files, or via ‘context’ (URI arguments from IMAS URI)

i.e. signal=“ip”

shot=45272

Passed in via libTokaMap config

Data Source Library

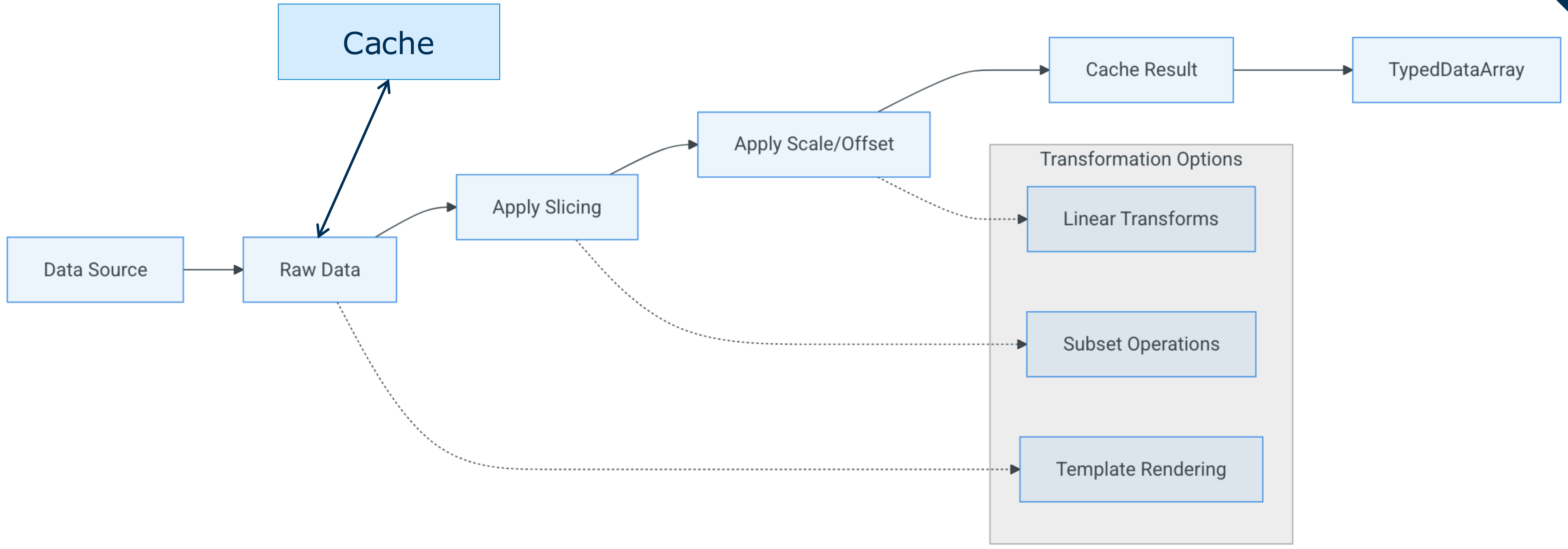
```
std::unique_ptr<libtokamap::DataSource>
json_data_source_factory(const libtokamap::DataSourceFactoryArgs& args)
{
    auto data_root = libtokamap::get_arg<std::string>(args, "data_root");
    return std::make_unique<JSONDataSource>(data_root);
}

extern "C"
void LibTokaMapFactoryLoader(libtokamap::FactoryEntryInterface& factory)
{
    factory.function = json_data_source_factory;
}
```

Function used to register data source factory

Function used to create customised instances of data source

Data Source operations

UK Atomic Energy Authority



Python wrapper – C++ data source

```
import libtokamap

config_path = "./config.toml"
mapper = libtokamap.Mapper(config_path)

experiment = "example"
res = mapper.map(experiment, "/path/to/path", {'shot': shot})
```

Python wrapper – Python data source

```
class UDADataSource(libtokamap.DataSource):
    def __init__(self, host: str, port: int, plugin_name: str):
        self.plugin_name = plugin_name
        pyuda.Client.server = host
        pyuda.Client.port = port
        self.client = pyuda.Client()

    @override
    def get(self, args: dict[str, str]) -> np.ndarray:
        ...
        result = self.client.get(request, '')
        return result.data
```

```
mapper.register_python_data_source(
  "UDA", UDADataSource("uda2.hpc.l", 59876, "UDA")
)
```

Find out more

Available at https://github.com/ukaea/libtokamap/
Documentation at https://ukaea.github.io/libtokamap/latest/

```
libtokamap/
 ├── include/          # Public headers
 ├── src/              # Implementation
|    ├── handlers/      # MappingHandler
|    ├── map_types/     # Mapping implementations
|    ├── utils/         # Utilities and helpers
|   └── exceptions/    # Exception types
 ├── examples/         # Usage examples
 ├── test/             # Unit tests
 ├── python/           # Python wrapper with example
└── docs/              # Documentation
```

Roadmap

UK Atomic Energy Authority

- General performance improvements including more aggressive caching
- Integration with other caching technologies such as Redis
- Improve documentation and examples
- Integrate xtensor for slicing/subsetting, etc.
- Implement multithreaded map execution
- Automatically pick up schemas installed with tokamap

Aspirational:
- Have the option to download mapping as required
- Have the option to download data sources as required
- Mapping portal acting for discoverability

Demo

Demo code available at:

https://github.com/jholloc/libtokamap_demo

Links

UK Atomic Energy Authority

Code repositories:

- TokaMap: https://github.com/ukaea/tokamap
- libTokaMap: https://github.com/ukaea/libtokamap
  - Including Debian, RPM, TGZ and ZIP packages

Documentation:

- TokaMap: https://ukaea.github.io/tokamap
- libTokaMap: https://ukaea.github.io/libtokamap

Homebrew package:

- https://github.com/jholloc/homebrew-formulae