Handover docs

CMake

How find_package(MantidFramework) works. Relies on a config file in the mantid repository, which defines the contents of the MantidFramework package:

```
# Defines the cmake targets for the Mantid framework.

@PACKAGE_INIT@

include(CMakeFindDependencyMacro)

find_dependency(Python @Python_VERSION_MAJOR@.@Python_VERSION_MINOR@ REQUIRED
              COMPONENTS Interpreter Development NumPy)
find_dependency(Boost COMPONENTS date_time regex serialization filesystem system python${P
find_dependency(Poco COMPONENTS Foundation Util XML Net Crypto NetSSL)
find_dependency(GSL)
find_dependency(ZLIB)
find_dependency(HDF5)
find_dependency(OpenSSL)
find_dependency(OpenMP COMPONENTS CXX)
find_dependency(HDF5 COMPONENTS C CXX HL)
find_dependency(Eigen3)

set(_mypackage_module_path_save "${CMAKE_MODULE_PATH}")
list(INSERT CMAKE_MODULE_PATH 0 "${CMAKE_CURRENT_LIST_DIR}/find_modules")

# Using custom find modules
find_dependency(Nexus)
find_dependency(JsonCPP)
find_dependency(TBB)
find_dependency(MuParser)

set(CMAKE_MODULE_PATH "${_mypackage_module_path_save}")
unset(_mypackage_module_path_save)

set(MODULES
    Types
    Json
    Kernel
    Parallel
    HistogramData
    Indexing
    Beamline
    Geometry
    API
    NexusGeometry
    DataObjects
    Catalog
```

```
    Nexus
    PythonInterfaceCore
)

foreach(module ${MODULES})
include("${CMAKE_CURRENT_LIST_DIR}/Mantid${module}Targets.cmake")
check_required_components(Mantid${module}Targets)
list(APPEND MantidFramework_Libraries "Mantid::${module}")
endforeach()
```

Each module in the MODULES list defines an importable target, e.g Mantid::Kernel, or Mantid::Types. These importable targets are created when we install libraries, which requires CONDA_BUILD=True and leads to the following install call for the kernel:

```
if(CONDA_BUILD)
  set(TARGET_EXPORT_NAME "MantidKernelTargets")
  mtd_install_framework_lib(TARGETS Kernel EXPORT_NAME ${TARGET_EXPORT_NAME})
endif()
```

The new mtd_install_framework_lib helper function does the following:

As well as the config and target files we also install our custom finders, e.g FindNexus, which ensures that the find_dependency macro will find don't have inbuilt find modules shipped with CMake.

Conda.

Conda provides a package manage to source our dependencies from. Environment files specify any package requirements we have. Our files follow a policy of specifying the minimum version of each package supported:

```
dependencies:
  - muparser>=2.3.2
```

If a library suitably following semantic versioning we shouldn't expect any incompatible API changes between minor versions of the libraries. However, this is not always the case. To get a broad overview of changes in binaries between versions the abi-laboratory tool can be used, e.g for boost

https://abi-laboratory.pro/index.php?view=timeline&l=boost