SnT 2023
CTBT: SCIENCE AND TECHNOLOGY CONFERENCE
HOFBURG PALACE - Vienna and Online
19 TO 23 JUNE

Development of software simulation pipeline for high resolution atmospheric transport modeling

Don Morton[1], Anne Tipka[2], Jolanta Kusmierczyk-Michulec[2], Robin Schoemaker[2]
[1]Boreal Scientific Computing, Fairbanks, Alaska, USA
[2]CTBTO, Vienna, Austria

Boreal Scientific Computing LLC

INTRODUCTION

This is an overview of an ongoing project to build a robust, easy-to-use, Atmospheric Transport Modeling (ATM) system for custom, high-resolution scenarios

METHODS/DATA

Implementation was based on creation of a layered architecture, presenting loosely-coupled components to the user, with rigorous, repeatable testing.

START

RESULTS

A workflow driver has been implemented, presenting users with the tools to define custom workflows within a workflow namelist

CONCLUSION

The project is ongoing, with current emphasis on "filling in gaps" and packaging according to best practices.

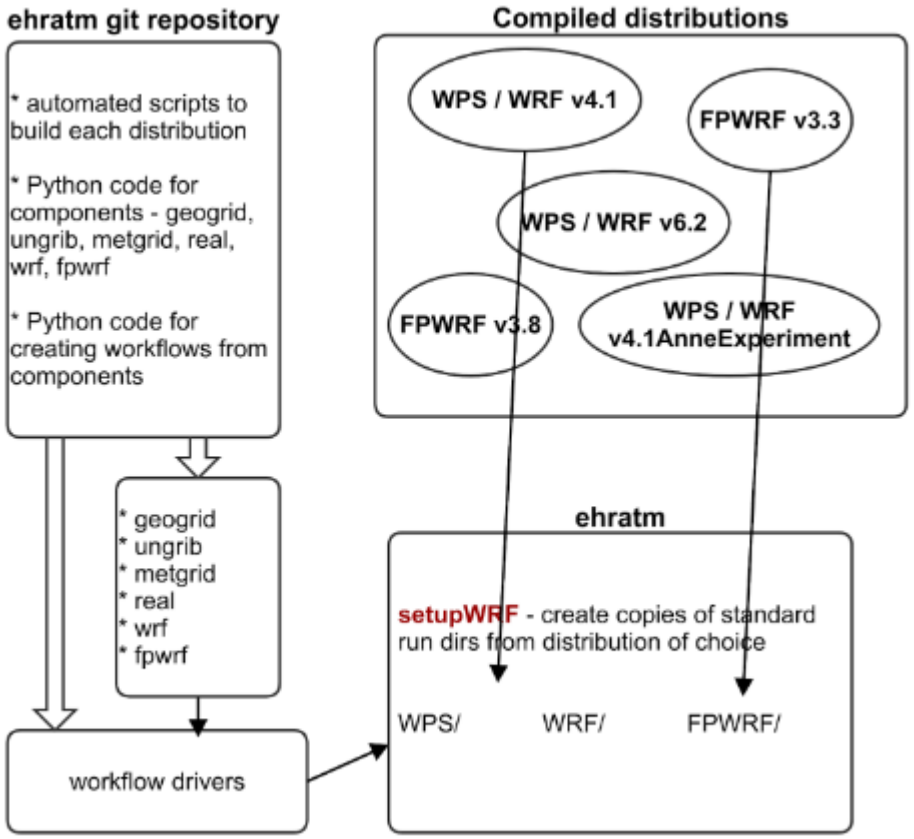
Please do not use this space, a QR code will be automatically overlayed

P3.4-488

Introduction

The Flexpart Atmospheric Transport Model (ATM) is traditionally driven by ECMWF and GFS meteorological model inputs. Flexpart-WRF is a variant of the standard model that accepts a wide range of WRF-generated meteorological inputs to support very high-resolution simulations over customised domains. The chain of activities needed to produce custom met files from WRF and feed them to Flexpart-WRF for a successful simulation is complex and prone to failure for a number of reasons, and the work described here is aimed at packaging all of the complexity into an easy-to-use system.

Building on the experiences gained from an exploratory prototype system built several years ago, this Enhanced High Resolution Atmospheric Transport Model (EHRATM) system is being developed in a Python-driven environment to support relatively simple and straightforward simulations to complex simulations with special requirements. Adopting the philosophy of some other well-known Python packages, our goal is to “make easy things easy and hard things possible.”



Please do not use this space, a QR code will be automatically overlayed

Objectives

The original HRATM was designed and well-implemented to primarily support a linear workflow of setting up and running a WRF simulation in order to generate custom meteorological data, and then to run Flexpart WRF using the custom met data. With some postprocessing for graphics production and generation of SRS-format output (used by CTBTO in many postprocessing activities), users are able to execute this workflow with a relatively simple command line interface.

Although the original HRATM has some support for performing partial workflows (doing some of the workflow at one time, and following up later with the rest of the workflow) for convenience, flexibility, and debugging, users expressed a need for much greater flexibility to substitute new and/or experimental components.

These needs for greater flexibility led to one of the primary design goals in the Enhanced HRATM (EHRATM) – the creation of a modular ecosystem of loosely-coupled distributed model components that could be inserted and removed in a plug and play fashion.

The primary objectives for this work are to

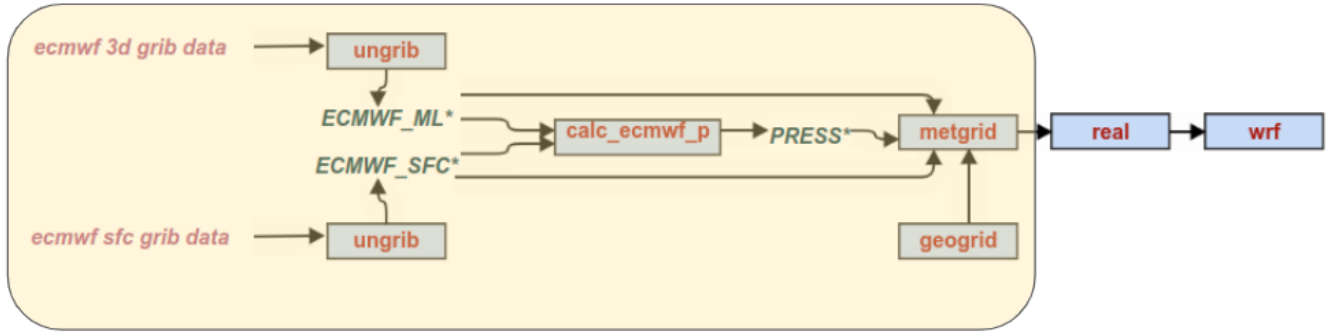
- build a Python package, *nwpservice*, of low-level WRF and Flexpart WRF components that can be executed and rigorously tested in standalone mode, independent of the other *nwpservice* components (in other words, loosely-coupled)
- build a higher-level Python package, *ehratm*, that provides the APIs for creating a custom workflow from the *nwpservice* components. Such a workflow is defined by a standard Fortran namelist file, and can range from specification of a single task based on a single *nwpservice* component, to a full workflow from WRF input to Flexpart WRF output. This package, too, has a number of components that undergo heavy unit and functional testing.
- build a prototype workflow driver, *ehratmwf.py*, that takes as a single argument the path to a user-created namelist file and runs the specified workflow. Additionally, implement a test environment of repeatable scenarios.

INTRODUCTION
OBJECTIVES
METHODS/DATA
RESULTS
CONCLUSION

Methods (Slide 1 of 2)

The *nwpservice* Python package is a collection of low-level, standalone components designed to operate as plug and play components of custom workflows.

Each component is specified by a well-defined Python API that allows it to be run by itself for operational, experimental or debugging activities. Extensive unit and functional testing provide a rigorous check on the correctness of the components, especially as they undergo future modifications.



Example of plug and play *nwpservice* components assembled into useful workflow

```
ctbtuser@localhost:~/git/hratm-experimentation/packagedev/nwpse
$ pytest test_components
============================= test session starts ==============================
platform linux -- Python 3.8.8, pytest-6.2.3, py-1.10.0, pluggy-0.13.1
rootdir: /home/ctbtuser/git/hratm-experimentation/packagedev/nwpservice/tests
plugins: anyio-2.2.0
collected 37 items

test_components/test_calcecmwfp/test_calcecmwfp_component.py .            [  2%]
test_components/test_flexwrf/test_flexwrf_component.py .....              [ 16%]
test_components/test_geogrid/test_geogrid_component.py ..                 [ 21%]
test_components/test_metgrid/test_metgrid_component.py ...                [ 29%]
test_components/test_namelistwps/test_namelistwps.py ..........           [ 59%]
test_components/test_real/test_real_component.py ....                     [ 70%]
test_components/test_ungrib/test_ungrib_component.py ....                 [ 81%]
test_components/test_wps_vizutils/test_vizutils.py ...                    [ 89%]
test_components/test_wrf/test_wrf_component.py ....                       [100%]

============================== 37 passed in 585.40s (0:09:45) ==============================
(ehratmv0.02) [EHRATM: ~/git/hratm-experimentation/packagedev/nwpservice/tests]
$
```

```
namelist_path = os.path.join(self._mydir, 'namelist.wps.ecmwfeu_sfc')
vtable_userdef_path = os.path.join(self._mydir, 'Vtable.ECMWF.sfc')
metdatatype = 'ecmwfsfc'
domainpath = os.path.join(self._tmpdir, 'MyTestDomain')
LOGGER.debug('domainpath: %s' % domainpath)
LOGGER.debug('self._tmp_output_dir: %s' % self._tmp_output_dir)
ungrib_obj = wps.ungrib.Ungrib(
                wpswrf_distro_path=WPSWRF_DISTRO_PATH,
                wpswrf_rundir=domainpath,
                namelist_wps=namelist_path,
                vtable_userdef=vtable_userdef_path,
                metdatadir=ECMWFEU_SFC_GRIB_DIR,
                metdatatype=metdatatype,
                output_dir=self._tmp_output_dir,
                log_level=logging.DEBUG
)

ungrib_obj.setup()
output_manifest = ungrib_obj.run()
LOGGER.debug('output_manifest: %s' % output_manifest)
LOGGER.debug('output_manifest keys: %s' %
             list(output_manifest['ungribbed']['files'].keys()))

self.assertIn('SURF:2014-01-24_03',
              list(output_manifest['ungribbed']['files'].keys()),
              msg="File in manifest")
```

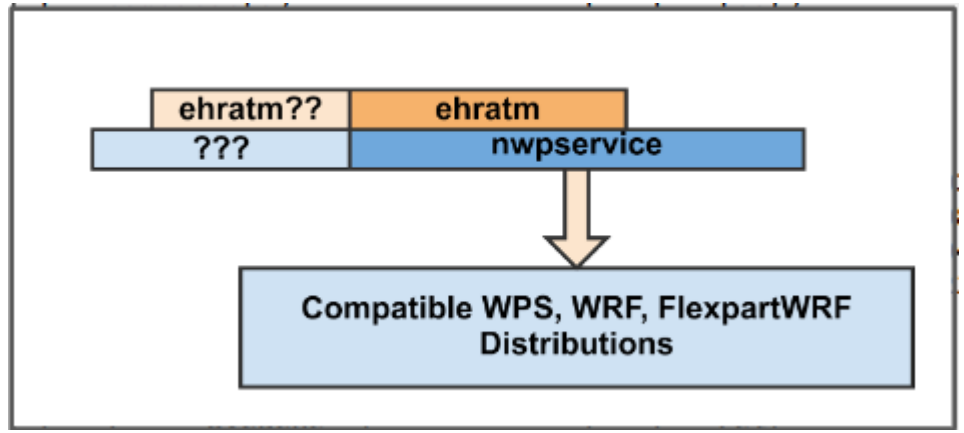
Example of initialisation and use of single *nwpservice ungrib* component for WRF preprocessing

INTRODUCTION
OBJECTIVES
METHODS/DATA
RESULTS
CONCLUSION

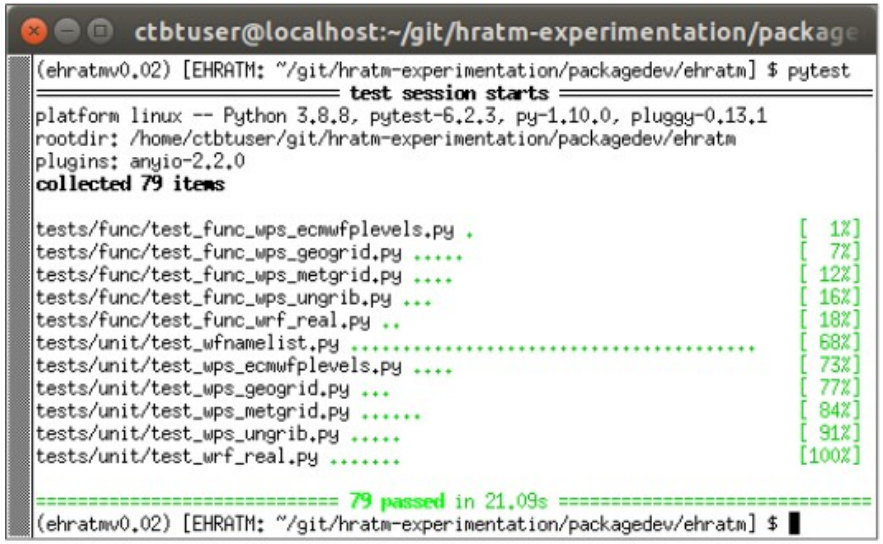
Methods (Slide 2 of 2)

The *ehratm* Python package is a collection of higher-level, standalone components intended to be called by *ehratm* worfklow drivers, handling many of the details so that, ultimately users only need to worry about creating correct workflow namelists (*wfnamelist*) for their custom projects.

Like the lower-level *nwpservice* components, the *ehratm* components are heavily instrumented with unit and functional tests.



Abstraction of the ehratm / nwpservice stack architecture. The nwpservice components may be called by any conforming software, and ehratm is one such collection. nwpservice depends on correctly installed distributions of the NWP models, and this installation is scripted in a repeatable way



ehratm.ehratmwf.EhratmWorkflow()	.run_ungrib()	.run_geogrid()	.run_ecmwfplevels()	.run_metgrid()
ehratm.wps	.ungrib.UngribWorkflow()	.geogrid.GeogridWorkflow()	.ecmwfplevels.EcmwfPlevelsWorkflow()	.metgrid.MetgridWorkflow()
nwpservice.wps	.ungrib.Ungrib()	.geogrid.Geogrid()	.calcecmwfp.CalcEcmwfPlevels()	.metgrid.Metgrid()

ehratm.ehratmwf.EhratmWorkflow()	.run_real()	.run_wrf()
ehratm.wrf	.real.RealWorkflow()	.wrf.WrfWorkflow()
nwpservice.wrf	.real.Real()	.wrf.Wrf()

ehratm.ehratmwf.EhratmWorkflow()	.run_flexwrf()
ehratm.flexwrf	.flexwrf.FlexwrfWorkflow()
nwpservice.flexwrf	.flexwrf.FlexWrf()

Stack architecture of the workflow components, depending on the *ehratm* components, which depend on the foundational *nwpservice* components. Each column's functionality is independent of the others, supporting the design goal of loosely-coupled distributed components.

Please do not use this space, a QR code will be automaticall y overlayed

Results

An EHRATM workflow driver (*ehratmwf*) is the top-level software component. It reads, parses and verifies the user-defined *wfnamelist* and runs the workflow components as defined in the workflow_list and the individual sections of the *wfnamelist*.

A Command Line Interface (CLI) batch processing test driver will execute and report on status of a large collection of test cases.

```
&control
  workflow_list = 'ungrib', 'geogrid', 'metgrid', 'real', 'wrf', 'flexwrf'
  log_level = 'debug'
  workflow_rootdir = '/tmp'
/

&time
  start_time = '2014012403'
  end_time = '2014012406'
  wrf_spinup_hours = 3
/

&grib_input1
  type = 'ecmwf_ml'
  hours_intvl = 3
  rootdir =
'/dvlscratch/ATM/morton/git/hratm-experimentation/packagedev/ehratm/tests/f
unc/era_metdata'
/

&grib_input2
  type = 'ecmwf_sfc'
  hours_intvl = 3
  rootdir =
'/dvlscratch/ATM/morton/git/hratm-experimentation/packagedev/ehratm/tests/f
unc/era_metdata'
/

&domain_defn
  domain_defn_path = 'small_domain_twonest.nml'
/

&geogrid
  num_mpi_tasks = 4
/
.
.
.
```

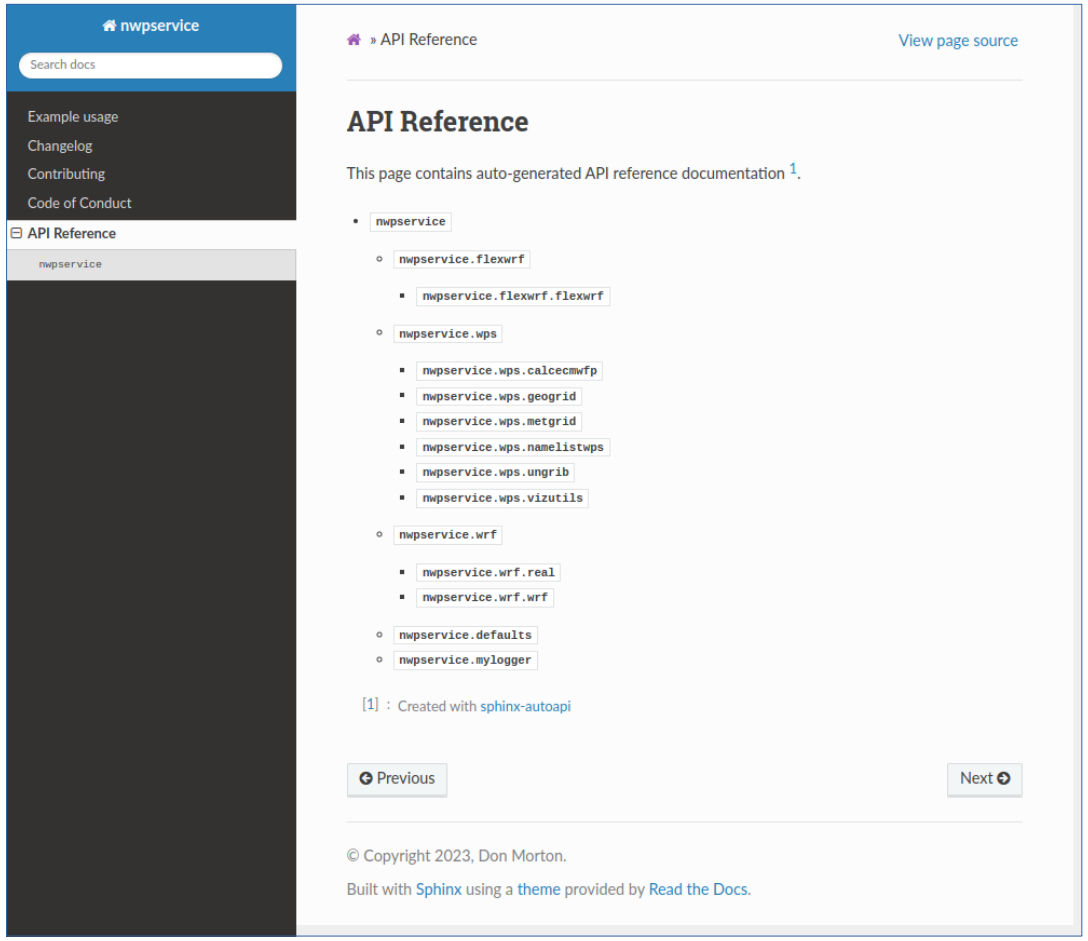
```
ctbtuser@localhost:~/git/hratm-experimentation/package
================================================================================
TEST 15/15
Running Test: wfnml_tests/wfnamelist.gfsungrib+geogrid-twonest


********************************************************************************
Last 10 stdout lines: wfnml_tests/wfnamelist.gfsungrib+geogrid-twonest
********************************************************************************
   .
   .
   .
========================================

Workflow Events
------------------
2023-02-15:01:50:39 --> Workflow started
2023-02-15:01:50:39 --> Start process_namelist()
2023-02-15:01:50:39 --> Started run_workflow(): /tmp/ehratmwf_20230215_015039.64
2523
2023-02-15:01:50:39 --> Start run_ungrib()
2023-02-15:01:50:44 --> Start run_geogrid()
2023-02-15:01:50:52 --> Workflow completed...



================================================
                SUMMARY RESULTS
================================================

return code   wfnamelist
-----------   ----------
     0        wfnml_tests/wfnamelist.ecmwf-metgridonly
     0        wfnml_tests/wfnamelist.ecmwf-metgridonly-twonest-4mpitasks
     0        wfnml_tests/wfnamelist.ecmwfungrib
     0        wfnml_tests/wfnamelist.ecmwfungrib+geogrid+metgrid-twonest-4mpitas
ks
     0        wfnml_tests/wfnamelist.ecmwfungrib+geogrid-twonest
     0        wfnml_tests/wfnamelist.ecmwfungrib+geogrid-twonest+4mpitasks
     0        wfnml_tests/wfnamelist.geogrid-onenest
     0        wfnml_tests/wfnamelist.geogrid-onenest-4mpitasks
     0        wfnml_tests/wfnamelist.geogrid-twonest
     0        wfnml_tests/wfnamelist.geogrid-twonest-4mpitasks
     0        wfnml_tests/wfnamelist.gfs-metgridonly
     0        wfnml_tests/wfnamelist.gfsungrib
     0        wfnml_tests/wfnamelist.gfsungrib+geogrid+metgrid-onenest
     0        wfnml_tests/wfnamelist.gfsungrib+geogrid-onenest
     0        wfnml_tests/wfnamelist.gfsungrib+geogrid-twonest


Number successful: 15/15


WARNING - workflow product directories NOT automatically deleted


153.30user 12.37system 1:35.47elapsed 173%CPU (0avgtext+0avgdata 206132maxreside
nt)k
0inputs+1281940outputs (36major+1818307minor)pagefaults 0swaps
(ehratmw0.02) [EHRATM: ~/git/hratm-experimentation/packagedev/ehratm] $
```

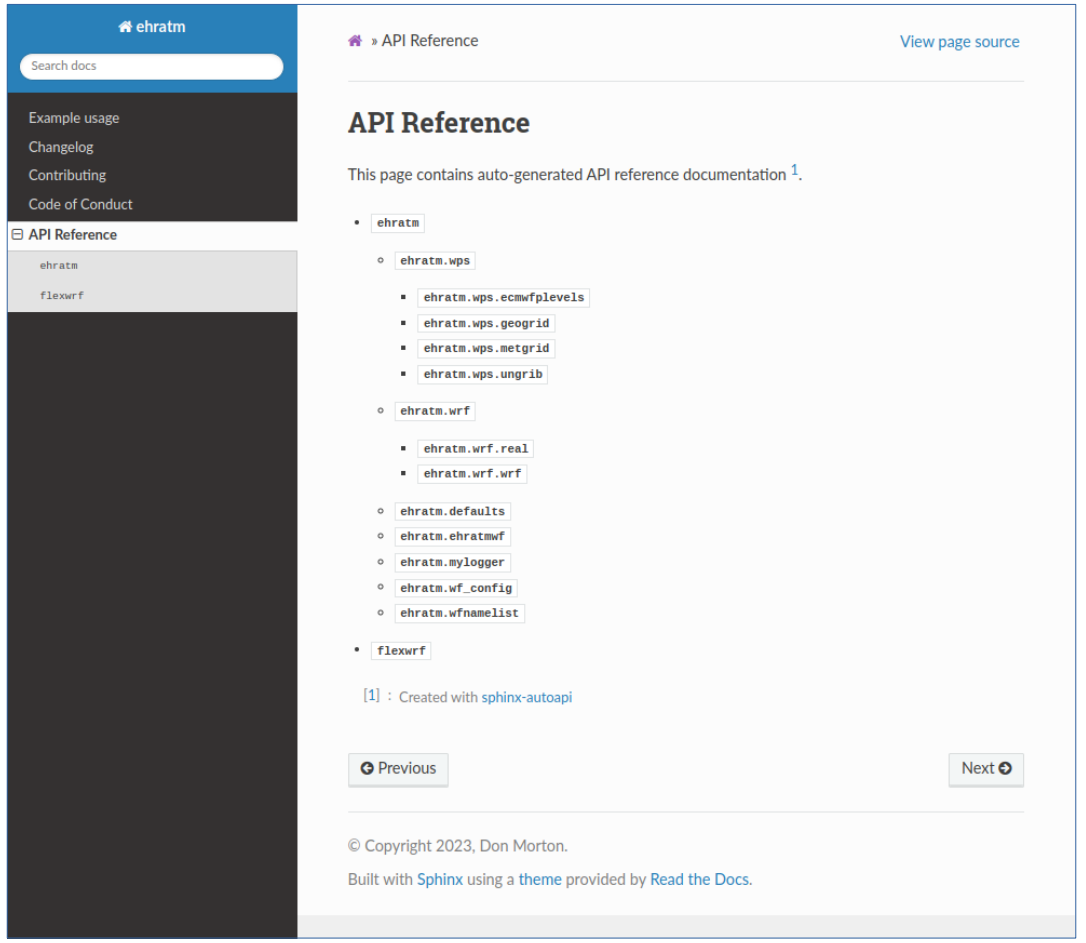
INTRODUCTION
OBJECTIVES
METHODS/DATA
RESULTS
CONCLUSION

Please do not use this space, a QR code will be automaticall y overlayed

Conclusion

This project is ongoing (and always will be), but the core functionality of a set of loosely-coupled distributed components, driven by namelist-specified workflow directives has been accomplished and tested rigorously. Continued work includes packaging up the experimental, prototype software into a best-practices environment with web-based documentation.





INTRODUCTION
OBJECTIVES
METHODS/DATA
RESULTS
CONCLUSION

Please do not use this space, a QR code will be automatically overlayed