UFS Medium-Range Weather App Users Guide

Sep 30, 2020

CONTENTS

CHAPTER

ONE

INTRODUCTION

The Unified Forecast System (*UFS*) is a community-based, coupled, comprehensive Earth modeling system with applications that span local to global domains and predictive time scales. It is designed to be the source system for NOAA's operational numerical weather prediction applications while enabling both research and capabilities for the broader weather enterprise. For more information about the UFS, visit the UFS Portal at https://ufscommunity.org/.

The Unified Forecast System (*UFS*) can be configured into multiple applications (see a complete list at https://ufscommunity.org/#/science/aboutapps). The first of these to be released to the community is the UFS Medium-Range (MR) Weather Application (App), which targets predictions of atmospheric behavior out to about two weeks. The MR Weather App v1.1 includes a prognostic atmospheric model, pre- and post-processing tools, and a community workflow The release is available on GitHub and is designed to be a code that the community can run and improve. It is portable to a set of commonly used platforms. A limited set of configurations of the release, such as specific model resolutions and physics options, are documented and supported. This documentation provides an overview of the release components, a description of the supported capabilities, a quick start for running the application, and information on where to find more information and obtain support.

1.1 Pre-processor and initial conditions

The MR Weather App is distributed with the *chgres_cube* pre-processing software. It converts the Global Forecast System (GFS) analyses to the format needed as input to the model, which is six tiles in netCDF format. Additional information about chgres_cube can be found in the chgres_cube User's Guide.

GFS analyses for initializing the MR Weather App can be in Gridded Binary v2 (*GRIB2*) format (in 0.50, or 1.0 degree grid spacing), the NOAA Environmental Modeling System (*NEMS*) Input/Output (*NEMSIO*) format, or Network Common Data Form (*NetCDF*). Initialization from dates starting on January 1, 2018 are supported. Dates before that may work, but are not guaranteed. GFS public archives can be accessed through the THREDDS Data Server at NCEI. A small sample of files in all supported formats can be found at the EMC FTP site. The initial conditions may be pre-staged on disk by the user or automatically downloaded by the workflow.

1.2 Forecast model

The prognostic model in the MR Weather App is the atmospheric component of the UFS Weather Model, which employs the Finite-Volume Cubed-Sphere (*FV3*) dynamical core. The atmospheric model in this release is an updated version of the atmospheric model that is being used in the operational GFS v15. A User's Guide for the UFS Weather Model is here.

Supported grid configurations for this release are the global meshes with resolutions of C96 (~100 km), C192 (~50 km), C384 (~25 km), and C768 (~13 km), all with 64 vertical levels. The NOAA Geophysical Fluid Dynamics Laboratory website provides more information about FV3 and its grids. Additional information about the FV3 dynamical core is at here. Interoperable atmospheric physics, along with the Noah land surface model, are supported through the use of the Common Community Physics Package (*CCPP*; described here). There are four physics suites supported for the release. Two of them are variations of an updated version of the physics *suite* used in the operational GFS v15, while the other two are variations of an experimental suite that includes a subset of the developments for the next version of GFS, GFS v16. The variations pertain to how the sea surface temperature (SST) is initialized and parameterized to evolve, and are chosen depending on the type of initial conditions for the App. Initial conditions in *GRIB2* format have a single two-dimensional field to initialize the SST, which must be kept constant throughout the forecast. Initial conditions in *NEMSIO* or *netCDF* format have two two-dimensional fields that describe the baseline SST and its near-surface perturbation related to the diurnal cycle, enabling the use of the near-sea-surface-temperature (NSST) physical parameterization to forecast the temporal variation in SST due to the diurnal cycle.

A scientific description of the CCPP parameterizations and suites can be found in the CCPP Scientific Documentation, and CCPP technical aspects are described in the CCPP Technical Documentation. The model namelists for the physics suites differ in ways that go beyond the physics to optimize various aspects of the model for use with each of the suites. The use of *stochastic* processes to represent model uncertainty is an option in this release, although the option is off by default in both of the supported physics suites. Three methods are supported for use separately or in combination: Stochastic Kinetic Energy Backscatter (SKEB), Stochastically Perturbed Physics Tendencies (SPPT), and Specific Humidity perturbations (SHUM). A User's Guide for the use of stochastic physics is provided.

The UFS Weather Model ingests files produced by chgres_cube and outputs files in netCDF format on a Gaussian grid in the horizontal and model levels in the vertical.

1.3 Post-processor

The MR Weather App is distributed with a post-processing tools, the Unified Post Processor (UPP). The Unified Post Processor (UPP) converts the native netCDF output from the model to the *GRIB2* format on standard isobaric coordinates in the vertical. The UPP can also be used to compute a variety of useful diagnostic fields, as described in the UPP user's guide.

The UPP output can be used with visualization, plotting and verification packages, or for further downstream post-processing, e.g. statistical post-processing techniques.

1.4 Visualization Example

This release does not include support for model verification or visualization. Currently, only four basic NCAR Command Language (*NCL*) scripts are provided to create a basic visualization of model output. This capability is provided only as an example for users familiar with NCL, and may be used to do a visual check to verify that the application is producing reasonable results.

The scripts are available in the FTP site ftp://ftp.emc.ncep.noaa.gov/EIB/UFS/visualization_example/. File visualization_README describes the plotting scripts. Example plots are provided for the C96 5-day forecasts initialized on 8/29/2019 00 UTC using *GRIB2*, *NEMSIO*, or *netCDF* files as input datasets.

1.5 Workflow and Build System

The MR Weather App has a user-friendly workflow and a portable build system that invokes the CMake build software before compiling the codes. This release is supported for use with Linux and Mac operating systems, with Intel and GNU compilers. There is a small set of system libraries that are assumed to be present on the target computer, including CMake, a compiler, and the MPI library that enables parallelism.

A few select computational platforms have been preconfigured for the release with all the required libraries for building community releases of UFS models and applications available in a central place. That means bundled libraries (*NCEPLIBS*) and third-party libraries (*NCEPLIBS-external*), including the Earth System Modeling Framework (ESMF) have both been built. Applications and models are expected to build and run out of the box. In preconfigured platforms, users can proceed directly to the using the workflow, as described in the *Quick Start chapter*.

A few additional computational platforms are considered configurable for the release. Configurable platforms are platforms where all of the required libraries for building community releases of UFS models and applications are expected to install successfully, but are not available in a central place. Applications and models are expected to build and run once the required bundled libraries (*NCEPLIBS*) and third-party libraries (*NCEPLIBS-external*) are built.

Limited-test and Build-Only computational platforms are those in which the developers have built the code but little or no pre-release testing has been conducted, respectively. A complete description of the levels of support, along with a list of preconfigured and configurable platforms can be found here.

The workflow leverages the Common Infrastructure for Modeling the Earth (*CIME*) Case Control System (CCS). As described in the CIME documentation, it comes with two default configurations, or Component Sets (compsets). One compset is used to evoke the physics *suite* used in the operational GFS v15, while the other is used to evoke the experimental GFS v16 physics. Based on the type of initial conditions, the workflow determines whether or not to employ the variant with simple or more complex SST. The workflow provides ways to choose the grid resolution, as well as to change namelist options, such as history file frequency. It also allows for configuration of other elements of the workflow; for example, whether to run some or all of the pre-processing, forecast model, and post-processing steps. The CIME builds the forecast model and the workflow itself, but not the *NCEP* Libraries or the pre- and post-processing tools.

CIME supports a set of tests for the MR Weather App, including the Smoke Startup Test, the Exact Restart from Startup Test, and the Modified Threading OPENMP bit for bit Test. These tests are described in more detail later in this document and are intended for users to verify the App installation in new platforms and to test the integrity of their code in case they modify the source code.

1.6 User Support, Documentation, and Contributing Development

A forum-based online support system with topical sections provides a centralized location for UFS users and developers to post questions and exchange information. The forum complements the distributed documentation, summarized here for ease of use.

Table 1.1: Centralized list of documentation

Documentation	Location
MR Weather App v1.1 User's Guide	https://ufs-mrweather-app.readthedocs.io/en/ufs-v1.1.0
chgres_cube User's Guide	https://ufs-utils.readthedocs.io/en/ufs-v1.1.0
UFS Weather Model v1.1 User's Guide	https://ufs-weather-model.readthedocs.io/en/ufs-v1.1.0
FV3 Documentation	https://noaa-emc.github.io/FV3_Dycore_ufs-v1.1.0/html/index.html
CCPP Scientific Documentation	https://dtcenter.org/GMTB/v4.1.0/sci_doc
CCPP Technical Documentation	https://ccpp-techdoc.readthedocs.io/en/v4.1.0
Stochastic Physics User's Guide	https://stochastic-physics.readthedocs.io/en/ufs-v1.1.0
ESMF manual	http://www.earthsystemmodeling.org/esmf_releases/public/ESMF_8_0_0/ESMF_refdoc
Common Infrastructure for Modeling the Earth	http://esmci.github.io/cime/versions/ufs_release_v1.1/html/index.html
Unified Post Processor	https://upp.readthedocs.io/en/ufs-v1.1.0

The UFS community is encouraged to contribute to the UFS development effort. Issues can be posted in the GitHub repository for the App or the relevant subcomponent to report bugs or to announce upcoming contributions to the code base. For a code to be accepted in the authoritative repositories, the code management rules of each component (described in their User's Guides) need to be followed. Innovations involving the UFS Weather Model need to be tested using the regression test described in its User's Guide. The regression tests distributed with the UFS Weather Model differ from the CIME-base tests distributed with the MR Weather App because the former are part of the official NOAA policy to accept innovations in its code base, while the latter are meant as a sanity check for users.

1.7 Future Direction

Users can expect to see incremental capabilities in upcoming releases of the MR Weather App to enhance research options and support operational forecast implementations. Planned advancements include addition of component models for other Earth domains (such as oceans and sea ice), cycled data assimilation for model initialization, and tools for objective forecast verification. Releases of other UFS applications, such as the Stand-Alone Regional (SAR) application are also forthcoming and will be announced through the UFS Forum and the UFS Portal.

1.8 How To Use This Document

This guide instructs both novice and experienced users on downloading, building and running the MR Weather Application.

If you are a new user, we recommend reading the first few sections of the CIME documentation which is written so that, as much as possible, individual sections stand on their own. The CIME documentation can be scanned and sections read in a relatively ad hoc order.

```
Throughout the guide, this presentation style indicates shell
commands and options, fragments of code, namelist variables, etc.
```

Note: Variables presented as `$VAR` in this guide typically refer to variables in XML files in a MR Weather App experimental case. From within a case directory, you can determine the value of such a variable with `./xmlquery VAR`. In some instances, `$VAR` refers to a shell variable or some other variable; we try to make these exceptions clear.

CHAPTER

TWO

WORKFLOW QUICK START

The following quick start guide is applicable to versions of the MR Weather App that are on preconfigured machines as listed here. For other machines, please refer to Chapter 5 before using the quick start guide.

The workflow for building and running the App is built on the CIME (Common Infrastructure for Modeling Earth) framework. Please refer to the CIME Porting Documentation if CIME has not yet been ported to the target machine.

If you are new to CIME, please consider reading the CIME Case Control System Part 1: Basic Usage *after downloading the code*. The CIME Users Guide will be easier to follow after the directory structure has been created by the *git clone* command.

This is the procedure for quickly setting up and running a case of MR Weather App.

- Download the MR Weather App
- Create a case: use `create_newcase`
- Setup a case: use `case.setup`
- Build a case: use `case.build`
- Run a case: use `case.submit`

2.1 Downloading the MR Weather App code and scripts

Access to the code requires git. You will need access to the command line clients, `git` (v1.8 or greater). You can download the latest version of the release code:

```
git clone https://github.com/ufs-community/ufs-mrweather-app.git -b ufs-v1.1.0 my_ufs_
↪sandbox
cd my_ufs_sandbox
```

Note: When cloning the ufs-mrweather-app repository, the connection to github may time out. In this case, resubmit the `git clone` command.

The information of being a "detached HEAD" is a standard git notification about a release tag. If you plan to add development to the codes, you will need a development branch.

To checkout MR Weather Model components, including CIME, run the `checkout_externals` script from /path/to/my_ufs_sandbox.

```
./manage_externals/checkout_externals
```

The `checkout_externals` script will read the configuration file `Externals.cfg` and will download the model source and CIME into /path/to/my_ufs_sandbox.

To see more details regarding the checkout_externals script from the command line, type:

```
./manage_externals/checkout_externals --help
```

To confirm a successful download of all components, you can run `checkout_externals` with the status flag to show the status of the externals:

```
./manage_externals/checkout_externals -S
```

This should show a clean status for all externals, with no characters in the first two columns of output, as in this example:

```
Checking status of externals: model, stochastic_physics, fv3, ccpp/framework, atmos_
↪cubed_sphere, ccpp/physics, fms, nems, tests/produtil/nceplibs-pyprodutil, fv3gfs_
↪interface, nems_interface, cime,
    ./cime
    ./src/model
    ./src/model/FMS
    ./src/model/FV3
    ./src/model/FV3/atmos_cubed_sphere
    ./src/model/FV3/ccpp/framework
    ./src/model/FV3/ccpp/physics
    ./src/model/FV3/cime
    ./src/model/NEMS
    ./src/model/NEMS/cime/
    ./src/model/NEMS/tests/produtil/NCEPLIBS-pyprodutil
    ./src/model/stochastic_physics
```

You should now have a complete copy of the source code in your /path/to/my_ufs_sandbox.

If there were problems obtaining an external, you might instead see something like:

```
e-   ./src/model/FV3
```

This might happen if there was an unexpected interruption while downloading. First try rerunning `./manage_externals/checkout_externals`. If there is still a problem, try running with logging turned on using:

```
./manage_externals/checkout_externals --logging
```

Check the `manage_externals.log` file to see what errors are reported.

2.2 Model Configurations

The MR Weather App can be configured at four out-of-the-box resolutions with two different compsets, `GFSv15p2` or `GFSv16beta`. These compsets invoke physics suites that use or not an ocean-evolving parameterization depending on the initial data provided. See the Introduction for more information on the physics suites provided with the release and see the frequently-asked questions (*FAQ*) section for more information on compsets, physics suites, and initial datasets.

- Details of available component sets and resolutions are available from the `query_config` tool located in the `cime/scripts` directory

```
cd $SRCROOT/cime/scripts
./query_config --help
```

where `$SRCROOT` is the top directory of the ufs-mrweather-app.

2.2.1 Supported component sets

The components of the modeling system can be combined in numerous ways to carry out various scientific or software experiments. A particular mix of components, along with component-specific configuration and/or namelist settings is referred to as component set or "compset". The MR Weather App has a shorthand naming convention for component sets that are supported out-of-the-box.

To determine what MR Weather App compsets are available in the release, use the following command:

```
cd $SRCROOT/cime/scripts
./query_config --compsets
```

This should show a list of available compsets:

```
Active component: ufsatm
       --------------------------------------
       Compset Alias: Compset Long Name
       --------------------------------------
   GFSv15p2              : FCST_ufsatm%v15p2_SLND_SICE_SOCN_SROF_SGLC_SWAV
   GFSv16beta            : FCST_ufsatm%v16beta_SLND_SICE_SOCN_SROF_SGLC_SWAV
```

2.2.2 Supported grids

CIME has the flexibility to support numerous model resolutions. To see the grids that are currently supported, use the following command

```
cd $SRCROOT/cime/scripts
./query_config --grids
```

This should show the a list of available grids for this release.

```
=========================================
GRID naming convention
=========================================
The notation for the grid longname is
    a%name_l%name_oi%name_r%name_m%mask_g%name_w%name
where
    a% => atm, l% => lnd, oi% => ocn/ice, r% => river, m% => mask, g% => glc, w% =>␣
↪wav

Supported grid configurations are given via alias specification in
the file "config_grids.xml". Each grid alias can also be associated  with the
following optional attributes

 -------------------------------------------------------------
       default component grids:

 component         compset       value
 -------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
atm      SATM              null
lnd      SLND              null
ocnice   SOCN              null
rof      SROF              null
glc      SGLC              null
wav      SWAV              null
iac      SIAC              null
-------------------------------------------------------------

alias: C96
  non-default grids are: atm:C96

alias: C192
  non-default grids are: atm:C192

alias: C384
  non-default grids are: atm:C384

alias: C768
  non-default grids are: atm:C768
```

As can be seen, MR Weather App currently supports four grids with the following nominal resolutions

- C96 (~100km)
- C192 (~50km),
- C384 (~25km)
- C768 (~13km),

and all with 64 vertical levels.

2.3 Setup the environment

Four environment variables need to be set prior to running the CIME workflow:

```
export UFS_INPUT=/path/to/inputs
export UFS_SCRATCH=/path/to/outputs
export UFS_DRIVER=nems
export CIME_MODEL=ufs
```

`UFS_INPUT` should be set to the location of a folder where input data will be accessed. There should be a folder named `ufs_inputdata` underneath this folder. The folder `$UFS_INPUT/ufs_inputdata` should exist before running the CIME workflow. This is often a shared location on a platform so that all users on that platform can access data from the same location.

`UFS_SCRATCH` should be set to the location of a writeable folder where output will be written for each case. This is typically a user scratch space or temporary location with a large allocation available.

The following settings are recommended on the pre-configured platforms:

Table 2.1: Path settings for pre-configured platforms.

Platform	$UFS_INPUT	$UFS_SCRATCH
NCAR Cheyenne	$CESMDATAROOT	/glade/scratch/$USER
NOAA Hera	/scratch1/NCEPDEV/stmp2/CIME_UFS	<my-project-dir>/$USER
NOAA Jet	/lfs4/HFIP/hfv3gfs/ufs-release-v1.1/CIME_UFS	<my-project-dir>/$USER
NOAA Gaea	/lustre/f2/pdata/esrl/gsd/ufs/ufs-release-v1.1/CIME_UFS	<my-project-dir>/$USER

On platforms that are not pre-configured a script needs to be executed to define a set of environment variables related to the location of NCEPLIBS dependencies.

```
# SH or BASH shells
source $NCEPLIBS_DIR/bin/setenv_nceplibs.sh

# CSH or TCSH shells
source $NCEPLIBS_DIR/bin/setenv_nceplibs.csh
```

The recommended best practice to set the `$UFS_SCRATCH` and `$UFS_INPUT` environment variables and source the NCEPLIBS provided shell script `setenv_nceplibs.sh|.csh` is to add the above commands to a startup script such as `$HOME/.bashrc` (Bash shell) or `$HOME/.tcshrc` (Tcsh shell). These files are executed automatically when you start a new shell so that you do not need to re-define them during each login.

Important: On some platforms (in particular Stampede2) this practice is **required** to ensure the environment variables are properly set on compute nodes accessed by the workflow.

2.4 Create a case

The create_newcase command creates a case directory containing the scripts and XML files to configure a case (see below) for the requested resolution, component set, and machine. `create_newcase` has three required arguments: `--case`, `--compset` and `--res`. The `workflow` argument is optional, to select alternate workflow components (see below). The `project` argument is optional, to set the batch system project account (see below). (invoke `create_newcase --help` for help).

On machines where a project or account code is needed, you must either specify the `--project $PROJECT` argument in the `create_newcase` command, or set the `$PROJECT` variable in your shell environment. If this argument is not set, the default value in config_machines.xml for `$PROJECT` will be used. An error will be reported if the default project account is not accessable.

If running on a preconfigured or configurable machine, that machine will normally be recognized automatically and therefore it is not required to specify the `--machine` argument to create_newcase. Generic linux and macos systems will require the `--machine linux` / `--machine macos` argument to be used (see Section 5.2).

Invoke `create_newcase` as follows from the `cime/scripts` directory:

```
cd cime/scripts
./create_newcase --case CASENAME --compset COMPSET --res GRID --workflow WORKFLOW
```

where:

- `CASENAME` defines the name of your case (stored in the `$CASE` XML variable). This is a very important piece of metadata that will be used in filenames, internal metadata and directory paths. `create_newcase` will create the *case directory* with the same name as the `CASENAME`. If `CASENAME` is simply a name (not a path), the case directory is created in the `cime/scripts` directory where you executed create_newcase. If

CASENAME is a relative or absolute path, the case directory is created there and the name of the case will be the tail path. The full path to the case directory will be stored in the $CASEROOT XML variable.

- COMPSET is the component set and can be GFSv15p2 or GFSv16beta, which trigger supported Common Community Physics Package (CCPP) suites. If you would like to learn more about CCPP please consider reading the CCPP Overview.
- GRID is the model resolution, which can be C96, C192, C384 and C768.
- WORKFLOW is the workflow and can be set as ufs-mrweather or ufs-mrweather_wo_post. The ufs-mrweather includes both pre- and post-processing steps, while ufs-mrweather_wo_post includes only pre-processing step. In the current version of the MR Weather App, the pre-processing step need to be run to generate initial conditions for the UFS Weather Model.
- PROJECT is the project or account code needed to run batch jobs. You may either specify the --project $PROJECT argument in the create_newcase command, or set the $PROJECT variable in your shell environment.

Here is an example on NCAR machine Cheyenne with the $USER shell environment variable set to your Cheyenne login name:

```
cd cime/scripts
./create_newcase --case $UFS_SCRATCH/ufs-mrweather-app-workflow.c96 --compset␣
↪GFSv15p2 --res C96 --workflow ufs-mrweather
```

2.5 Setting up the case run script

Issuing the case.setup command creates scripts needed to run the model along with namelist user_nl_xxx files, where xxx denotes the set of components for the given case configuration such as ufsatm and cpl. Selected namelist entries can be customized by editing user_nl_xxx, see *FAQ*.

cd to the case directory or case root ($CASEROOT) $UFS_SCRATCH/ufs-mrweather-app-workflow.c96 as shown above:

```
cd /glade/scratch/$USER/cases/ufs-mrweather-app-workflow.c96
```

Before invoking case.setup, you could modify the env_mach_pes.xml file in the case directory using the xmlchange command as needed for the experiment (optional). (Note: To edit any of the env xml files, use the xmlchange command. xmlchange --help can be used for help.)

Please also be aware that you need to provide consistent layout, write_tasks_per_group and write_groups namelist options to the model when total number of PEs are changed.

Invoke the case.setup command.

```
./case.setup
```

Note: The CIME commands ./xmlquery, ./case.setup, ./case.build, ./case.submit examine and modify the CIME case, and so, are linked into the directory specified by --case when the ./create_newcase is run. They should be run from this case directory.

2.6 Build the executable using the case.build command

Modify build settings in `env_build.xml` (optional).

Run the build script.

```
./case.build
```

Users of the NCAR cheyenne system should consider using qcmd to compile UFS Weather Model on a compute node as follows:

```
qcmd -- ./case.build
```

The UFS Weather Model executable (named as `ufs.exe`) will appear in the directory given by the XML variable `$EXEROOT`, which can be queried using:

```
./xmlquery EXEROOT
```

2.7 Run the case

Modify runtime settings in `env_run.xml` (optional). Two settings you may want to change now are:

1. Run length: By default, the model is set to run for 5 days based on the `$STOP_N` and `$STOP_OPTION` variables:

   ```
   ./xmlquery STOP_OPTION,STOP_N
   ```

   These default settings can be useful in troubleshooting runtime problems before submitting for a longer time or a production runs. For example, following setting can be used to set the simulation lenght to 36-hours. Please, also be aware that `nyears`, `nmonths` and `nsteps` options for `STOP_OPTION` are not supported in the MR Weather App.

   ```
   ./xmlchange STOP_OPTION=nhours,STOP_N=36
   ```

2. You can set the `$DOUT_S` variable to FALSE to turn off short term archiving:

   ```
   ./xmlchange DOUT_S=FALSE
   ```

3. The default job wall clock time, which is set to 12-hours, can be changed for relatively short and low-resolution simulations. For example, following commands sets the job wall clock time to 30-minutes.

   ```
   ./xmlchange JOB_WALLCLOCK_TIME=00:30:00
   ./xmlchange USER_REQUESTED_WALLTIME=00:30:00
   ```

4. The default start date (2019-08-29, 00 UTC) can be also changed by following commands

   ```
   ./xmlchange RUN_STARTDATE=YYYY-MM-DD
   ./xmlchange START_TOD=AS_SECOND
   ```

   where:

   - `RUN_STARTDATE` is the start date and need to be given in YYYY-MM-DD format such as 2020-01-15
   - `START_TOD` is the time of day in seconds such as 12 UTC need to be given as 43200 seconds.

Submit the job to the batch queue using the `case.submit` command.

```
./case.submit
```

Based on the selected workflow (`ufs-mrweather` or `ufs-mrweather_wo_post`), the `case.submit` command submits a chain of jobs that their dependency is automatically set. For example, `ufs-mrweather` workflow submits a job array with three seperate jobs that will run in an order: pre-processing, simulation and post-processing. The first ten characters of the job names will be `chgres.ufs`, `run.ufs-mr`, and `gfs_post.u`, respectively.

When the jobs are complete, most output will *NOT* be written under the case directory, but instead under some other directories (defined by $UFS_SCRATCH). Review the following directories and files, whose locations can be found with `xmlquery` (note: `xmlquery` can be run with a list of comma separated names and no spaces):

```
./xmlquery RUNDIR,CASE,CASEROOT,DOUT_S,DOUT_S_ROOT
```

- `$RUNDIR`

  This directory is set in the `env_run.xml` file. This is the location where MR Weather App was run. Log files for each stage of the workflow can be found here.

  Table 2.2: Log files

Component	File Name	Look for...
chgres.ufs	chgres_cube.yymmdd-hhmmss.log	"DONE"
run.ufs-mr	ufs.log.<jobid>.yymmdd-hhmmss	"PROGRAM nems HAS ENDED"
gfs_post.ufs	oi.hhh	"PROGRAM UNIFIED_POST HAS ENDED"

- `$DOUT_S_ROOT/$CASE`

  `$DOUT_S_ROOT` refers to the short term archive path location on local disk. This path is used by the case.st_archive script when `$DOUT_S = TRUE`.

  `$DOUT_S_ROOT/$CASE` is the short term archive directory for this case. If `$DOUT_S` is FALSE, then no archive directory should exist. If `$DOUT_S` is TRUE, then log, history, and restart files should have been copied into a directory tree here.

- `$DOUT_S_ROOT/$CASE/logs`

  The log files should have been copied into this directory if the run completed successfully and the short-term archiver is turned on with `$DOUT_S = TRUE`. Otherwise, the log files are in the `$RUNDIR`.

- `$CASEROOT`

  There could be standard out and/or standard error files output from the batch system.

- `$CASEROOT/CaseDocs`

  The case namelist files are copied into this directory from the `$RUNDIR`.

CHAPTER

THREE

CODE REPOSITORIES AND DIRECTORY STRUCTURE

This chapter describes the code repositories that comprise the MR Weather App, without describing, in detail, any of the components.

3.1 Hierarchical Repository Structure

The umbrella repository for the MR Weather App is named ufs-mrweather-app and is available on GitHub at https://github.com/ufs-community/ufs-mrweather-app. An umbrella repository is defined as a repository that includes links, called externals, to additional repositories. The MR Weather App includes the `checkout_externals` tools along with a configuration file called `Externals.cfg`, which describes the external repositories associated with this umbrella (see Table 3.1).

Table 3.1: List of top-level repositories that comprise the MR Weather App.

Repository Description	Authoritative repository URL
Umbrella repository for the MR Weather App	https://github.com/ufs-community/ufs-mrweather-app
Umbrella repository for the UFS Weather Model	https://github.com/ufs-community/ufs-weather-model
CIME	https://github.com/ESMCI/cime
Layer required for CIME to build ufs-weather-model	https://github.com/ESCOMP/fv3gfs_interface
Layer required for CIME to build NEMS driver	https://github.com/ESCOMP/NEMS_interface
UPP	https://github.com/NOAA-EMC/EMC_post

The UFS MR Weather Model is itself an umbrella repository and contains a number of sub-repositories used by the model as documented here. The CIME repository contains the workflow and build system for the prognostic model. The two layer repositories provide interfaces to allow CIME to properly build the ufs-weather-model and the NEMS driver.

Note: Note that the prerequisite libraries (including NCEP Libraries) are not included in the MR Weather App repository. The source code for these components resides in the umbrella repositories NCEPLIBS and NCEPLIBS-external. The former has links to the chgres_cube preprocessor repository and to UPP.

These external components are already built on the preconfigured platforms listed here. However, they must be cloned and built on other platforms according to the instructions provided in the wiki pages of those repositories: <https://github.com/NOAA-EMC/NCEPLIBS/wiki>` and <https://github.com/NOAA-EMC/NCEPLIBS-external/wiki>.

The UPP code works in two ways with the MR Weather App. The code to create the UPP executable is part of the NCEPLIBS, and the executable is already installed in preconfigued platforms. For the purposes of enabling customization of UPP fields, as described in the *Inputs and Outputs chapter*, the UPP code is also included in the MR Weather App umbrella repository.

3.2 Directory Structure

The directory structure on disk for users of the MR Weather App depends on whether one is using a pre-configured platform. Users working on pre-configured platforms will only have the files associated with the ufs-mrweather-app in their disk space. The directory structure is set in configuration file `Externals.cfg`, which is in the top directory where the umbrella repository has been cloned. A listing of the directory structure is shown *here*.

The directory structures for the standalone UFS Weather Model and the UFS Weather Model included with the MR Weather App are equal in that they contain subdirectories for *FMS*, *FV3*, *NEMS* and stochastic_physics. However, in the MR Weather App, subdirectories are located under `src/model`. The MR Weather App also includes directories for CIME, such as the `src/model/FV3/cime` and `src/model/NEMS/cime` directories.

Users working outside of preconfigured platforms will have additional files on disk associated with the libraries, pre- and post-processing. The resulting directory structure is determined by the path settings in the NCEPLIBS `.gitmodules` file.

CHAPTER

FOUR

INPUTS AND OUTPUTS

This chapter provides an overview of the input and output files needed by the components of the MR Weather App (*chgres_cube*, the UFS *Weather Model*, and *UPP*). Links to more detailed documentation for each of the components are provided.

4.1 Input files

The MR Weather App requires numerous input files. The input files data format can be GRIB2, NEMSIO, or netCDF, and the input files must be staged by the user. *CIME* can run the end-to-end system and write output files to disk.

4.1.1 chgres_cube

When a user runs the MR Weather App as described in the quickstart guide, input data for chgres_cube is linked from a location on disk to your run directory via CIME. The data is stored in a hierarchical way in the `$DIN_LOC_IC` directory (see Section 4.3). A list of the input files for chgres_cube can be found here.

4.1.2 UFS Weather Model

The input files for the MR Weather Model are located one directory up from the chgres_cube input files in `$RUNDIR` (see Section 2.7). An extensive description of the input files for the MR Weather Model can be found in the UFS Weather Model Users Guide.

Note: Due to renaming/linking by CIME, the file names used in the MR Weather App differ from the names described in the UFS Weather Model User's Guide.

4.1.3 UPP input files

Documentation for the UPP input files can be found here.

4.2 Output files

The location of the output files written to disk is determined by CIME (see Section 2.7).

4.2.1 chgres_cube

The files output by chgres_cube reside in the `$DIN_LOC_IC` directory, and are linked by CIME to files that include the grid name and date in the same directory. For example:

```
sfc_data.tile[1-6].nc -> C96.2019-08-28_00.sfc_data.tile[1-6].nc
gfs_ctrl.nc -> C96.2019-08-28_00.gfs_ctrl.nc
gfs_data.tile1.nc -> C96.2019-08-28_00.gfs_data.tile1.nc
```

These output files are used as input for the UFS Weather Model.

Note: The same input directory could have multiple pre-processed input files for different dates and once the run date is changed, CIME is able to link the correct files with the names that model expects.

4.2.2 UFS Weather Model

The output files for the UFS Weather Model are described in the Users Guide.

4.2.3 UPP output files

Documentation for the UPP output files can be found here.

If you wish to modify the fields or levels that are output from the UPP, you will need to make modifications to files `postcntrl_gfs_f00.xml` (used to post-process model data at the 0-h forecast lead time) and/or `postcntrl_gfs.xml` (used to post-process model data at all other forecast lead times), which reside in the UPP repository distributed with the MR Weather App. Specifically, if the code was cloned in the directory `my_ufs_sandbox`, the files will be located in `my_ufs_sandbox/src/post/parm`. Please note that this process requires advanced knowledge of which fields can be output for the UFS Weather Model.

Use the directions in the UPP Users Guide for details on how to make modifications to these xml files and for remaking the flat text files that the UPP reads, which are `postxconfig-NT-GFS.txt` and `postxconfig-NT-GFS-F00.txt`. It is important that you do not rename these flat files or the CIME workflow will not use them.

Once you have created new flat text files reflecting your changes, you will need to copy or link these static files to the `/SourceMods/src.ufsatm` directory within the CIME case directory. When running your case, CIME will first look for the `postxconfig-NT-GFS.txt` or `postxconfig-NT-GFS-F00.txt` in this directory, depending on forecast hour. If they are not present, the workflow will use the default files in a pre-configured location.

You may then setup/build/run your case as usual and the UPP will use the new flat `*.txt` files.

4.3 Downloading and staging input data

A set of input files, including static (fix) data and raw initial conditions, are needed to run the MR Weather App. There are two variables that describe the location of the static and initial condition files: `$DIN_LOC_ROOT` is the directory where the static files are located and `$DIN_LOC_IC` is the directory where the initial conditions are located. By default, `$DIN_LOC_ROOT` is set to $UFS_INPUT/ufs_inputdata and `$DIN_LOC_IC` is set to `$DIN_LOC_ROOT/icfiles`. In this directory, the initial conditions are located in subdirectories named `YYYYMM/YYYYMMDD` (YYYY: year, MM: month, DD: day).

Variable `$DIN_LOC_ROOT` is already set in preconfigured platforms and points to a centralized location where the fix files are staged. Similarly, variable `$DIN_LOC_IC` is by default set to `$DIN_LOC_ROOT/icfiles` and points to the directory with initial conditions for the Hurricane Dorian initialization in 08-29-2019. In all other platforms, users can customize the location of the fix files by setting *$UFS_INPUT* to a writable directory and creating a subdirectory $UFS_INPUT/ufs_inputdata.

A customized location for `$DIN_LOC_IC` is necessary when users need to stage new initial condition files and do not have write permission to `$DIN_LOC_ROOT`. Users can customize `$DIN_LOC_IC` after creating the case using the commands below.

```
cd $CASEROOT
./xmlchange DIN_LOC_IC=/path/to/directory
```

4.3.1 Static files

The user does not need to stage the fix files manually because CIME retrieves the fix files from `$DIN_LOC_ROOT` (if available) or from a FTP data repository. When CIME retrieves the files from the ftp site, it places them in `$DIN_LOC_ROOT`.

4.3.2 Initial condition formats and source

The MR Weather App currently only supports the use of Global Forecast System (GFS) data as raw initial conditions (that is, MRF, AVN, ERA5 etc. are not supported). The GFS data can be provided in three formats: *NEMSIO*, *netCDF*, or *GRIB2*.

- **NEMSIO**

  These files cover the entire globe down to a horizontal resolution of 13 km and can be found at https://nomads.ncep.noaa.gov/pub/data/nccf/com/gfs/prod/. A small sample is also available at the FTP data repository https://ftp.emc.ncep.noaa.gov/EIB/UFS/.

- **NetCDF**

  These files cover the entire globe down to a horizontal resolution of 13 km and can be found at the FTP data repository https://ftp.emc.ncep.noaa.gov/EIB/UFS/.

- **GRIB2**

  These files cover the entire globe and resolutions of 0.5 and 1.0 degree are supported. There are both current and historic sources of GRIB2 data available. At the time of this writing, files for dates 05/15/2020 12 UTC or more recent are considered current, while files for preceding dates are considered historical. However, the cutoff date may change in the future. Here are the locations:

  – 0.5 deg current files are available at https://www.ncei.noaa.gov/thredds/catalog/model-gfs-g4-anl-files/catalog.html

   - 0.5 deg historical files are available at https://www.ncei.noaa.gov/thredds/catalog/model-gfs-g4-anl-files-old/catalog.html
   - 1.0 deg current files can be requested from https://www.ncei.noaa.gov/thredds/catalog/model-gfs-g3-anl-files/catalog.html
   - 1.0 deg historical files can be requested from https://www.ncei.noaa.gov/thredds/catalog/model-gfs-g3-anl-files-old/catalog.html

A small sample is also available at the FTP data repository https://ftp.emc.ncep.noaa.gov/EIB/UFS/.

4.3.3 Initial condition naming convention

The default naming convention for the initial condition files is described below. The user must stage the files on disk following this convention so they can be recognized by the MR Weather App workflow.

- **NEMSIO**
  - Two-dimensional surface variables `sfc.input.ic.nemsio`
  - Three-dimensional atmosphere state `atm.input.ic.nemsio`
- **NetCDF**
  - Two-dimensional surface variables `sfc.input.ic.nc`
  - Three-dimensional atmosphere state `atm.input.ic.nc`
- **GRIB2**
  - Surface variables and atmosphere state `atm.input.ic.grb2`

4.3.4 Default initial conditions

All supported CompSets use the Hurricane Dorian initialization of 08-29-2019. In preconfigured platforms, the 08-29-2019 initial conditions are pre-staged in `$DIN_LOC_IC`. Those are GRIB2 files with 0.5 deg resolution.

The default input data for the Hurricane Dorian initialization of 08-29-2019 is also available on the FTP data repository.

4.3.5 Running the App for different dates

If users want to run the MR Weather App for dates other than 08-29-2019, they need to make a change in the case to specify the desired data. This is done by setting the `RUN_STARTDATE` and `START_TOD` CIME options using `./xmlchange`.

CIME will look for the following directory containing initial conditions: `$DIN_LOC_IC/YYMMMM/YYYYMMDD`.

Starting with the v1.1.0 release, the MR Weather App workflow no longer auto-downloads datasets. The data must be present in the centralized location (for preconfigured platforms) or downloaded manually.

4.3.6 Staging initial conditions manually

If users want to run the MR Weather App with initial conditions other than what is currently available in preconfigured platforms, they need to stage the data manually. The data should be placed in `$DIN_LOC_IC`.

Note: The following example script, `get.sh` can be used as a reference to download the NEMSIO file from the NOMADS server for a sample date, which in this case is 24-12-2018. **Note that NEMSIO files in NOMADS are only available for the last 10-days.**

```
#!/bin/bash

# Command line arguments
if [ -z "$1" -o -z "$2" ]; then
   echo "Usage: $0 yyyymmdd hh"
   exit
fi
yyyymmdd=$1 #i.e. "20191224"
hh=$2 #i.e. "12"

# Get the data (do not need to edit anything after this point!)
yyyymm=$((yyyymmdd/100))
din_loc_ic=`./xmlquery DIN_LOC_IC --value`
mkdir -p $din_loc_ic/$yyyymm/$yyyymmdd
echo "Download files to $din_loc_ic/$yyyymm/$yyyymmdd ..."
cd $din_loc_ic/$yyyymm/$yyyymmdd
wget -c https://nomads.ncep.noaa.gov/pub/data/nccf/com/gfs/prod/gfs.$yyyymmdd/$hh/gfs.
↪t${hh}z.atmanl.nemsio
wget -c https://nomads.ncep.noaa.gov/pub/data/nccf/com/gfs/prod/gfs.$yyyymmdd/$hh/gfs.
↪t${hh}z.sfcanl.nemsio
cd -
```

Script `get.sh` should be placed in **$CASEROOT** and used as follows:

```
chmod 755 get.sh
./get.sh 20191224 12
```

After downloading the nemsio files, the downloaded files need to be linked to the names expected by the App:

```
ln -s gfs.t${hh}z.atmanl.nemsio atm.input.ic.nemsio
ln -s gfs.t${hh}z.sfcanl.nemsio sfc.input.ic.nemsio
```

For downloading files in GRIB2 format with 0.5 degree grid spacing, the same code `get.sh` can be used except the wget command should be replaced with the following line:

```
#For current files:
wget -c https://www.ncei.noaa.gov/thredds/catalog/model-gfs-g4-anl-files/$yyyymmdd/
↪gfsanl_4_${yyyymmdd}_${hh}00_000.grb2
#For historic files:
wget -c https://www.ncei.noaa.gov/thredds/catalog/model-gfs-g4-anl-files-old/
↪$yyyymmdd/gfsanl_4_${yyyymmdd}_${hh}00_000.grb2
```

For downloading files in GRIB2 format with 1.0 degree grid spacing, the same code `get.sh` can be used except the wget command should be replaced with the following line:

```
#For current files:
wget -c https://www.ncei.noaa.gov/thredds/catalog/model-gfs-g3-anl-files/$yyyymmdd/
↪gfsanl_3_${yyyymmdd}_${hh}00_000.grb2
```

(continues on next page)

(continued from previous page)

```
#For historical files:
wget -c https://www.ncei.noaa.gov/thredds/catalog/model-gfs-g3-anl-files-old/
↪$yyyymmdd/gfsanl_3_${yyyymmdd}_${hh}00_000.grb2
```

After downloading the file, the user must link the new file to the name expected by the App. For example,

```
ln -s gfsanl_3_20190829_0000_000.grb2 atm.input.ic.grb2
```

For downloading files in netCDF format, the wget commands in `get.sh` need to be changed to:

```
wget -c https://ftp.emc.ncep.noaa.gov/EIB/UFS/inputdata/$yyyymm/gfs.$yyyymmdd/$hh/gfs.
↪t${hh}z.atmf000.nc
wget -c https://ftp.emc.ncep.noaa.gov/EIB/UFS/inputdata/$yyyymm/gfs.$yyyymmdd/$hh/gfs.
↪t${hh}z.sfcf000.nc
```

Currently, only two sample netCDF files are available for testing at the FTP data repository. Similarly, the downloaded files need to be linked to the names expected by the App. For example,

```
ln -s gfs.t${hh}z.atmf000.nc atm.input.ic.nc
ln -s gfs.t${hh}z.sfcf000.nc sfc.input.ic.nc
```

4.3.7 Order of operations

If you want to download the input data manually, you should do it before you build the MR Weather App.

4.3.8 Coexistence of multiple files for the same date

Directory *$DIN_LOC_IC/YYMMMM/YYYYMMDD* can have GRIB2, NEMSIO, and netCDF files for a given initialization hour and can have files for multiple initialization hours (00, 06, 12, and 18 UTC).

If a directory has files in more than one format for the same initialization date and time, CIME will use the GRIB2 files. If the user wants to change this behavior so CIME uses the NEMSIO or netCDF files, the user should edit file `user_nl_ufsatm` and add

```
input_type = "gaussian_nemsio" for NEMSIO
input_type = "gaussian_netcdf" for netCDF
```

4.3.9 Best practices for conserving disk space and keeping files safe

Initial condition files are large and can occupy a significant amount of disk space. If various users will employ a common file system to conduct runs, it is recommended that these users share the same `$DIN_LOC_ROOT`. That way, if initial conditions are already on disk for a given date, they do not needed to be replicated.

The files in the subdirectories of `$DIN_LOC_ROOT` should be write-protected. This prevents these files from being accidentally modified or deleted. The directories in `$DIN_LOC_ROOT` should generally be group writable, so the directory can be shared among multiple users.

CHAPTER

FIVE

CONFIGURING A NEW PLATFORM

5.1 Software/Operating System Prerequisites

The following are the external system and software requirements for installing and running MR Weather Application.

- UNIX style operating system such as CNL, AIX, Linux, Mac
- Python 2.7 or 3
- Perl 5
- Git client (1.8 or greater)
- Fortran compiler with support for Fortran 2003
- C compiler
- MPI
- NCEPLIBS-external (includes ESMF)
- NCEPLIBS
- CMake 3.15 or newer

Note that *NCEPLIBS-external* and *NCEPLIBS* reside in separate GitHub repositories. NCEPLIBS-external is a collection of third-party libraries required to build NCEPLIBS, which contains the NCEP library source code and utilities required for chgres_cube, the UFS Weather Model, and UPP. NCEPLIBS-external must be installed before building the NCEPLIBS, and both are a prerequesite for porting CIME to a new platform. The *NCEPLIBS-external* and *NCEPLIBS* repositories each contain a wiki page with instructions. More details are in Section 5.2.

5.2 Generic MacOS or Linux platforms

CIME defines a generic build for MacOS using homebrewi and generic Linux machines. You must first install NCEPLIBS-external and NCEPLIBS following the instructions here.. Then you will need to set the environment variable NCEPLIBS_DIR pointing to the install location (/usr/local/ufs-release-v1). You will also need to define a root location for the model input and output data, again using environment variables. The following are suggestions:

- `UFS_INPUT $HOME/projects`
- `UFS_SCRATCH $HOME/projects/scratch`

Create these directories:

- `mkdir -p $HOME/projects/scratch`
- `mkdir -p $HOME/projects/ufs_inputdata`

You are now ready to build the ufs-mrweather-app as documented in the *Workflow Quick Start*. Use the optional `--machine` argument to create_newcase and create_test with value `macos` or `linux`.

5.3 Porting CIME to a new machine

This section describes the steps needed to port the CIME workflow to a new platform.

5.3.1 Add the new machine description to config_machines.xml

Edit the file **$CIMEROOT/config/ufs/machines/config_machines.xml** and add a new *<machine/>* entry under the root XML element. A good approach to this is to copy an existing *<machine/>* description and modify it to match the new machine to which you are porting CIME. An example entry looks like this:

```
<machine MACH="hera">
  <DESC>NOAA hera system</DESC>
  <NODENAME_REGEX>hfe</NODENAME_REGEX>
  <OS>LINUX</OS>
  <COMPILERS>intel</COMPILERS>
  <MPILIBS>impi</MPILIBS>
  <PROJECT>nems</PROJECT>
  <SAVE_TIMING_DIR/>
  <CIME_OUTPUT_ROOT>/scratch1/NCEPDEV/nems/$USER</CIME_OUTPUT_ROOT>
  <DIN_LOC_ROOT>/scratch1/NCEPDEV/nems/Rocky.Dunlap/INPUTDATA</DIN_LOC_ROOT>
  <DIN_LOC_ROOT_CLMFORC>/scratch1/NCEPDEV/nems/Rocky.Dunlap/INPUTDATA/atm/datm7</DIN_
↪LOC_ROOT_CLMFORC>
  <DOUT_S_ROOT>$CIME_OUTPUT_ROOT/archive/$CASE</DOUT_S_ROOT>
  <BASELINE_ROOT>/scratch1/NCEPDEV/nems/Rocky.Dunlap/BASELINES</BASELINE_ROOT>
  <CCSM_CPRNC>/home/Rocky.Dunlap/bin/cprnc</CCSM_CPRNC>
  <GMAKE>make</GMAKE>
  <GMAKE_J>8</GMAKE_J>
  <BATCH_SYSTEM>slurm</BATCH_SYSTEM>
  <SUPPORTED_BY>NCEP</SUPPORTED_BY>
  <MAX_TASKS_PER_NODE>80</MAX_TASKS_PER_NODE>
  <MAX_MPITASKS_PER_NODE>40</MAX_MPITASKS_PER_NODE>
  <PROJECT_REQUIRED>TRUE</PROJECT_REQUIRED>
  <mpirun mpilib="default">
    <executable>srun</executable>
    <arguments>
      <arg name="num_tasks">-n $TOTALPES</arg>
    </arguments>
  </mpirun>
  <mpirun mpilib="mpi-serial">
    <executable></executable>
  </mpirun>
  <module_system type="module">
    <init_path lang="sh">/apps/lmod/lmod/init/sh</init_path>
    <init_path lang="csh">/apps/lmod/lmod/init/csh</init_path>
    <init_path lang="python">/apps/lmod/lmod/init/env_modules_python.py</init_path>
    <cmd_path lang="sh">module</cmd_path>
    <cmd_path lang="csh">module</cmd_path>
    <cmd_path lang="python">/apps/lmod/lmod/libexec/lmod python</cmd_path>
    <modules compiler="intel">
      <command name="purge"/>
      <command name="load">intel/18.0.5.274</command>
```

(continues on next page)

(continued from previous page)

```
    </modules>
    <modules mpilib="impi">
      <command name="load">netcdf/4.7.0</command>
      <command name="load">impi/2018.0.4</command>
      <command name="use">/scratch1/BMC/gmtb/software/modulefiles/intel-18.0.5.274/
↪impi-2018.0.4</command>
      <command name="load">NCEPlibs/1.0.0alpha01</command>
    </modules>
    <modules>
      <command name="use">/scratch1/BMC/gmtb/software/modulefiles/generic</command>
      <command name="load">cmake/3.16.3</command>
    </modules>
  </module_system>
  <environment_variables comp_interface="nuopc">
    <env name="ESMF_RUNTIME_PROFILE">ON</env>
    <env name="ESMF_RUNTIME_PROFILE_OUTPUT">SUMMARY</env>
  </environment_variables>
</machine>
```

Many of the XML elements above are self-explanatory. For details about individual elements see the config_machines.xml file section in the CIME documentation.

The value of `CCSM_CPRNC` will be set in the step below after the "cprnc" is installed on the system.

When finished, verify that your **config_machines.xml** file conforms to its schema definition:

```
cd $CIMEROOT
xmllint --noout --schema config/xml_schemas/config_machines.xsd config/ufs/machines/
↪config_machines.xml
```

5.3.2 Add the batch system to config_batch.xml

Edit file **$CIMEROOT/config/ufs/machines/config_batch.xml** and add a *<batch_system/>* element describing the batch system on the new machine. Again, this can be done by copying an existing element and making any needed modifications. Here is an example batch description:

```
<batch_system MACH="hera" type="slurm">
  <batch_submit>sbatch</batch_submit>
  <submit_args>
    <arg flag="--time" name="$JOB_WALLCLOCK_TIME"/>
    <arg flag="-q" name="$JOB_QUEUE"/>
    <arg flag="--account" name="$PROJECT"/>
  </submit_args>
  <directives>
    <directive>--partition=hera</directive>
  </directives>
  <queues>
    <queue walltimemax="08:00:00" nodemin="1" nodemax="210">batch</queue>
    <queue default="true" walltimemax="00:30:00" nodemin="1" nodemax="210">debug</
↪queue>
  </queues>
</batch_system>
```

For more details see the config_batch.xml file description in the CIME documentation.

To verify correctness of the config_batch.xml file, use the command:

```
cd $CIMEROOT
xmllint --noout --schema config/xml_schemas/config_batch.xsd config/ufs/machines/
↪config_batch.xml
```

5.3.3 (Optional) Build and install the "cprnc" tool

The CIME testing system uses a tool called `cprnc` to compare netCDF files. This tool can either be built one time on a system and referenced from the **config_machines.xml** file or it will be built automatically by CIME if not found.

If you choose to build `cprnc` use these steps:

```
cd $CIMEROOT/tools/cprnc
CIMEROOT=../.. ../configure --macros-format=Makefile --mpilib=mpi-serial
CIMEROOT=../.. source ./.env_mach_specific.sh && make
```

You should now have a `cprnc` executable. Ideally, this executable will be moved to a shared location so that all users on the platform have access to the tool. Update **$CIMEROOT/config/ufs/machines/config_machines.xml** and set `CCSM_CPRNC` to the path of the `cprnc` executable.

5.3.4 Verify that the port is working by running a simple test

Once you have completed the above steps, run the following test to see if you are able to build and run a basic workflow with the UFS MR Weather App.

```
cd $CIMEROOT/scripts
./create_test SMS_Lh5.C96.GFSv15p2 --workflow ufs-mrweather --machine
↪$MACHINE
```

The **$MACHINE** is the name of the machine that you added to the **config_machines.xml**.

This will attempt to run the full end-to-end workflow including pre-processing, model forecast, and post-processing.

CHAPTER

SIX

TESTING

There are several tests available as part of the regression testing suite to ensure the system is installed correctly and works properly. The regression test also confirms that code upgrades do not have side effects on existing functionalities and ensures that the system still works after code changes are made. The regression test can only be run on Cheyenne, Orion, and Stampede.

Pre-existing baselines are not provided for this App. Users can run the tests without using a baseline (just to certify that the tests run to completion) or create their own baseline (to compare future runs against).

create_test is the CIME tool used to execute the regression tests. It can be used as an easy way to run a single basic test or an entire suite of tests. create_test runs a test suite in parallel for improved performance. It is the driver behind the automated nightly testing of cime-driven models.

More information about CIME testing can be found on CIME: Testing.

6.1 Test requirements

In order to run the tests, NCEPLIBS and NCEPLIBS-external need to be installed (see Chapter 5 for instructions). These libraries have been preinstalled on Cheyenne, but not on Stampede and Orion.

The code must have been downloaded before the regression tests can be run. This can be done with the following commands:

```
mkdir -p $myUFS_INPUT/ufs_inputdata/icfiles/201908/20190829  # Create subdirectory␣
↪for raw ICs
git clone https://github.com/ufs-community/ufs-mrweather-app.git -b ufs-v1.1.0 my_ufs_
↪sandbox
cd my_ufs_sandbox
./manage_externals/checkout_externals
```

Several environment variables need to be set before running the regression tests. The instructions below provide quick information on how to set up the environment variables (for complete information, users should refer to Chapter 2).

```
export myUFS_INPUT=my_directory       # Directory for staging input datasets
export UFS_SCRATCH=my_scratch_space   # Directory for output files
export PROJECT=your_compute_project   # Project you can use to conduct runs in your␣
↪platform
export UFS_DRIVER=nems                # Do not change
export CIME_MODEL=ufs                 # Do not change
```

The input data required for the tests needs to be on disk before the tests are submitted. It is already staged on Cheyenne since it is a preconfigured platform. On Orion and Stampede, data must be acquired from the ftp site and staged on disk before proceeding with the test. The instructions below provide quick information on how to stage data on disk (for complete information, users should refer to Chapter 4).

```
mkdir -p $myUFS_INPUT/ufs_inputdata/icfiles/201908/20190829  # Create subdirectory␣
↪for raw ICs
cd $myUFS_INPUT/ufs_inputdata/icfiles/201908/20190829
wget https://ftp.emc.ncep.noaa.gov/EIB/UFS/inputdata/201908/20190829/gfs_4_20190829_
↪0000_000.grb2
ln -s gfs_4_20190829_0000_000.grb2 atm.input.ic.grb2 # Link raw ICs to expected name
```

6.2 Testname syntax

Tests are named with the following forms, [ ]=optional:

```
TESTTYPE[_MODIFIERS].GRID.COMPSET[.MACHINE_COMPILER][.GROUP-TESTMODS]
```

where:

- `TESTTYPE` defines the general type of test, e.g. SMS. Following is the list of tests that are supported by Medium-Range Weather Application.
  - **SMS: Smoke startup test (default 5 days).**

    Do a 5 day initial test. (file suffix: base)
  - **ERS: Exact restart from startup (default 6 days + 5 days)**

    Do an 11 day initial test - write a restart at day 6. (file suffix: base)
    Do a 5 day restart test, starting from restart at day 6. (file suffix: rest)
    Compare component history files '.base' and '.rest' at day 11. They should be identical.
  - **PET: Modified threading OPENMP bit for bit test (default 5 days)**

    Do an initial run where all components are threaded by default. (file suffix: base) Do another initial run with NTHRDS=1 for all components. (file suffix: single_thread) Compare base and single_thread.
- `MODIFIERS` changes to the default settings for the test.
- `GRID` The model grid (can be an alias). Currently, `C96`, `C192`, `C384` and `C768` are supported.
- `COMPSET` alias of the compset, or long name, if no `--xml` arguments are used. It can be `GFSv15p2` or `GFSv16beta`.
- `MACHINE` This is optional; if this value is not supplied, create_test will probe the underlying machine.
- `COMPILER` If this value is not supplied, use the default compiler for `MACHINE`.
- `GROUP-TESTMODS` This is optional. This points to a directory with `user_nl_xxx` files or a `shell_commands` that can be used to make namelist and `XML` modifications prior to running a test.

6.3 Query list of supported tests

$CIMEROOT/scripts/query_testlists gathers descriptions of the tests and testlists available for UFS, the components, and projects. The available tests for Cheyenne:

```
prealpha   : SMS_Lh3.C96.GFSv15p2.cheyenne_intel
prealpha   : SMS_Lh3.C96.GFSv15p2.cheyenne_gnu
prealpha   : SMS_Lh3.C96.GFSv16beta.cheyenne_intel
prealpha   : SMS_Lh3.C96.GFSv16beta.cheyenne_gnu
prealpha   : SMS_Lh3_D.C96.GFSv15p2.cheyenne_intel
prealpha   : SMS_Lh3_D.C96.GFSv15p2.cheyenne_gnu
prealpha   : SMS_Lh3_D.C96.GFSv16beta.cheyenne_intel
prealpha   : SMS_Lh3_D.C96.GFSv16beta.cheyenne_gnu
prealpha   : ERS_Lh11.C96.GFSv15p2.cheyenne_intel
prealpha   : ERS_Lh11.C96.GFSv15p2.cheyenne_gnu
prealpha   : ERS_Lh11.C96.GFSv16beta.cheyenne_intel
prealpha   : ERS_Lh11.C96.GFSv16beta.cheyenne_gnu
prealpha   : PET_Lh11.C96.GFSv15p2.cheyenne_intel
prealpha   : PET_Lh11.C96.GFSv15p2.cheyenne_gnu
prealpha   : SMS_Lh3.C192.GFSv15p2.cheyenne_intel
prealpha   : SMS_Lh3.C192.GFSv15p2.cheyenne_gnu
prealpha   : SMS_Lh3.C192.GFSv16beta.cheyenne_intel
prealpha   : SMS_Lh3.C192.GFSv16beta.cheyenne_gnu
prealpha   : SMS_Lh3_D.C192.GFSv15p2.cheyenne_intel
prealpha   : SMS_Lh3_D.C192.GFSv15p2.cheyenne_gnu
prealpha   : SMS_Lh3_D.C192.GFSv16beta.cheyenne_intel
prealpha   : SMS_Lh3_D.C192.GFSv16beta.cheyenne_gnu
prealpha   : SMS_Lh3.C384.GFSv15p2.cheyenne_intel
prealpha   : SMS_Lh3.C384.GFSv15p2.cheyenne_gnu
prealpha   : SMS_Lh3.C384.GFSv16beta.cheyenne_intel
prealpha   : SMS_Lh3.C384.GFSv16beta.cheyenne_gnu
prealpha   : SMS_Lh3_D.C384.GFSv15p2.cheyenne_intel
prealpha   : SMS_Lh3_D.C384.GFSv15p2.cheyenne_gnu
prealpha   : SMS_Lh3_D.C384.GFSv16beta.cheyenne_intel
prealpha   : SMS_Lh3_D.C384.GFSv16beta.cheyenne_gnu
prealpha   : SMS_Lh3.C768.GFSv15p2.cheyenne_intel
prealpha   : SMS_Lh3.C768.GFSv15p2.cheyenne_gnu
prealpha   : SMS_Lh3.C768.GFSv16beta.cheyenne_intel
prealpha   : SMS_Lh3.C768.GFSv16beta.cheyenne_gnu
prealpha   : SMS_Lh3_D.C768.GFSv15p2.cheyenne_intel
prealpha   : SMS_Lh3_D.C768.GFSv15p2.cheyenne_gnu
prealpha   : SMS_Lh3_D.C768.GFSv16beta.cheyenne_intel
prealpha   : SMS_Lh3_D.C768.GFSv16beta.cheyenne_gnu
```

The results indicate that there are tests available on Cheyenne for two compilers (Intel and GNU). Furthermore, the results indicate that all tests are part of a `testlist` called prealpha. `Testlists` are lists that aggregate a number of tests under a single umbrella. All tests contained in a `testlist` can be run with single command when the `testlist` is passed as an argument to the create_test command.

The `--xml-{compiler,machine,category,testlist}` arguments can be used as in create_test (above) to focus the search. The 'category' descriptor of a test can be used to run a group of associated tests at the same time. The available categories, with the tests they encompass, can be listed by:

```
cd $SRCROOT/cime/scripts
./query_testlists --define-testtypes
```

The `--show-options` argument does the same, but displays the 'options' defined for the tests, such as queue, walltime, etc..

6.4 Using create_test

To run a SMS test:

```
cd $SRCROOT/cime/scripts
./create_test SMS_D_Lh5.C96.GFSv15p2 --workflow ufs-mrweather_wo_post --test-id try
```

This will build and run the test in `/glade/scratch/$USER/SMS_D_Lh5.C96.GFSv15p2.cheyenne_intel.try` and this directory is called as **CASEROOT**. The run directory is in **CASEROOT/run** and the build is in **CASEROOT/bld**.

In this case, the C96 resolution model case with CCPP suite version v15p2 is created and runs 5 hours (**Lh5**) without post-processing step.

To run a test with baseline comparisons against baseline name 'master':

```
cd $SRCROOT/cime/scripts
./create_test SMS_Lh5.C96.GFSv15p2 --workflow ufs-mrweather_wo_post --test-id try --
↪compare master --baseline-root $BASELINE_ROOT
```

To run a Exact restart test:

```
cd $SRCROOT/cime/scripts
./create_test ERS_Lh11.C96.GFSv15p2 --workflow ufs-mrweather_wo_post --test-id try
```

This will build and run the test that includes two runs, first an 11 hour initial run (cold start) with a restart written at hour 6 and then a restart run (warm start) starting from hour 6 and compare the outputs written at hour 11. The output of the runs must be same.

To run a threaded test:

```
cd $SRCROOT/cime/scripts
./create_test PET_Lh11.C96.GFSv15p2 --workflow ufs-mrweather_wo_post --test-id try
```

To run entire test suite:

```
cd $SRCROOT/cime/scripts
./create_test --xml-testlist ../../src/model/FV3/cime/cime_config/testlist.xml --xml-
↪machine MACHINE --generate GENERATE --baseline-root BASELINE_ROOT --workflow ufs-
↪mrweather_wo_post
```

This will run entire test suite on specified machine `MACHINE` such as Stampede2 and generates the baseline under `BASELINE_ROOT` directory with a name of `GENERATE`.

The commands to run the regression test on Cheyenne, Orion, and Stampede are below. You must replace the compute projects listed (using variable `PROJECT`) to a project you can use to run the tests. For Cheyenne:

```
qcmd -l walltime=3:00:00 -- "export UFS_DRIVER=nems; CIME_MODEL=ufs;␣
↪PROJECT=p48503002 ./create_test --xml-testlist ../../src/model/FV3/cime/cime_config/
↪testlist.xml --xml-machine cheyenne --workflow ufs-mrweather_wo_post  --xml-
↪category prealpha"
```

For Orion:

```
export UFS_DRIVER=nems; CIME_MODEL=ufs; PROJECT=gmtb ./create_test --xml-testlist ../.
↪./src/model/FV3/cime/cime_config/testlist.xml --xml-machine orion --generate␣
↪GENERATE --baseline-root BASELINE_ROOT --workflow ufs-mrweather_wo_post --xml-
↪compiler intel --xml-category prealpha
```

On Stampede, it is necessary to submit the tests divided in three `testlists` (*prealpha_p1*, *pre_alpha_p2*, and *prealpha_p3*) because there is a limit to the number of jobs a user can have in the queue at a given time. Users should submit each set of tests separately, and wait for all tests to finish before submitting the next set:

```
export UFS_DRIVER=nems; CIME_MODEL=ufs; PROJECT=tg854445 ./create_test --xml-testlist␣
↪../../src/model/FV3/cime/cime_config/testlist.xml --xml-machine stampede2-skx --
↪workflow ufs-mrweather_wo_post -j 4 --walltime 01:00:00 --xml-compiler intel --xml-
↪category prealpha_p1
export UFS_DRIVER=nems; CIME_MODEL=ufs; PROJECT=tg854445 ./create_test --xml-testlist␣
↪../../src/model/FV3/cime/cime_config/testlist.xml --xml-machine stampede2-skx --
↪workflow ufs-mrweather_wo_post -j 4 --walltime 01:00:00 --xml-compiler intel --xml-
↪category prealpha_p2
export UFS_DRIVER=nems; CIME_MODEL=ufs; PROJECT=tg854445 ./create_test --xml-testlist␣
↪../../src/model/FV3/cime/cime_config/testlist.xml --xml-machine stampede2-skx --
↪workflow ufs-mrweather_wo_post -j 4 --walltime 01:00:00 --xml-compiler intel --xml-
↪category prealpha_p3
```

The running status can be checked by the following command:

```
./cs.status
```

Test success is defined as no failures and no jobs left in pending (PEND) state.

CHAPTER

SEVEN

FAQ

7.1 How can I set required environment variables?

The best practice to set environment variables (`UFS_SCRATCH`, `UFS_INPUT`, `UFS_DRIVER`, `CIME_MODEL`, and `PROJECT`) is to put them into the `.bashrc` (Bash shell) or `.tcshrc` (Tcsh shell) files. These files are executed when a user opens a new shell or logs in to the server (including compute nodes) so that their environment is set correctly. In platforms that are not preconfigured, the NCEPLIBS-provided shell script (`setenv_nceplibs.sh|.csh`) should be sourced in the same way.

BASH (edit ~/.bashrc):

```
export UFS_INPUT=/path/to/inputs
export UFS_SCRATCH=/path/to/outputs
export PROJECT=your_compute_project
export UFS_DRIVER=nems
export CIME_MODEL=ufs
source /path/to/nceplibs/bin/setenv_nceplibs.sh
```

BASH (edit ~/.tcshrc):

```
setenv UFS_INPUT /path/to/inputs
setenv UFS_SCRATCH /path/to/outputs
setenv PROJECT your_compute_project
setenv UFS_DRIVER nems
setenv CIME_MODEL ufs
source /path/to/nceplibs/bin/setenv_nceplibs.csh
```

Note: The user might need to create the `~/.bashrc` or `~/.tcshrc` file.

7.2 How can I see/check the steps in my workflow?

A good way to see what `case.submit` will do, is to use the `preview_run` command, which will output the environment for your run along with the batch submit and mpirun commands.

```
cd $CASEROOT
./preview_run
```

7.3 How can I run an individual task in the existing workflow?

The CIME allows you to run a specific task in the workflow by supplying the `--only-job` parameter to the `case.submit` command.

The following example will run only the preprocessing utility `chgres_cube`:

```
cd $CASEROOT
./case.submit --only-job case.chgres
```

This will create the initial conditions for the model simulation using the raw input files that are provided by NOAA Operational Model Archive and Distribution System (NOMADS).

To run the simulation:

```
cd $CASEROOT
./case.submit --only-job case.run
```

If the user wants to define the first job submitted in a workflow, the `--job` parameter can be passed to the `case.submit` command.

```
cd $CASEROOT
./case.submit --job case.run
```

In this case, two dependent jobs will be submitted: model simulation and post-processing.

7.4 How can I change the wall clock time and queue for specific tasks in the workflow?

These can be done by using the `xmlchange` command.

For example, the following command can be used to set the job wall clock time to 10 minutes for `chgres_cube`

```
cd $CASEROOT
./xmlchange JOB_WALLCLOCK_TIME=00:10:00 --subgroup case.chgres
```

The following command will change the job queue to `bigmem` for `chgres_cube`:

```
cd $CASEROOT
./xmlchange JOB_QUEUE=bigmem --subgroup case.chgres
```

Note: Without the `--subgroup` option, the `xmlchange` command changes the job wall clock time for all submitted jobs.

7.5 How can I change the project account that will be used to submit jobs?

There are two ways to change project account that is used to submit job:

- Set `PROJECT` environment variable before creating case
- Use the `xmlchange` command to change the project account (please replace PROJECT ID with an appropriate project number).

```
cd $CASEROOT
./xmlchange PROJECT=[PROJECT ID]
```

Note: A PROJECT environment variable setting will take precident over the case XML setting.

7.6 How do I change the processor layout for the UFS Weather Model?

The total number of processor used by the UFS Weather Model can be modified by using `xmlchange` command and editing the `user_nl_ufsatm` file.

To query the default configuration of the processor layout:

```
cd $CASEROOT
./pelayout
```

and to change the default processor layout:

```
cd $CASEROOT
./xmlchange NTASKS_ATM=150
```

This will set the total number of processors to 150, but the model configuration files (`model_configure` and `input.nml`) must be changed to be consistent with the total number of processors set by the `xmlchange` command.

In this case, the following namelist options need to be modified accordingly:

- **layout**: Processor layout on each tile.
- **ntiles**: Number of tiles on the domain. For the cubed sphere, this should be 6, one tile for each face of the cubed sphere.
- **write_groups**: Number of group for I/O tasks.
- **write_tasks_per_group**: Number of I/O tasks for each group.

The number of tasks assigned to a domain for UFS Medium-Range Weather Model must be equal to:

$$NTASKS\_ATM = layout_x * layout_y * ntiles + write\_tasks\_per\_group * write\_groups$$

to have consistent model configuration with **NTASKS_ATM** defined above. `user_nl_ufsatm` can be changed as following:

```
!----------------------------------------------------------------------------------
! Users should add all user specific namelist changes below in the form of
!   namelist_var = new_namelist_value
! Note - that it does not matter what namelist group the namelist_var belongs to
!----------------------------------------------------------------------------------
layout = 3,8
write_groups = 1
write_tasks_per_group = 6
```

Note: The model resolution also needs to divide evenly with the layout pair. For the given configuration (C96 resolution), $96/3 = 32$ and $96/8 = 12$.

7.7 How do I change the number of OPENMP threads?

The user may need to change the number of threads to reduce memory consumption for each compute node. This is especially true for high-resolution cases, and is already set by CIME for C768. This can be done using the following command:

```
cd $CASEROOT
./xmlchange NTHRDS_ATM=4
./case.setup --reset
./case.build --clean-all
./case.build
```

Note: The model needs to be built again if threading is changed from 1. Setting **NTHRDS_ATM** does not require changes in the model configuration files. The job submission scripts handle it automatically and submit jobs using more compute nodes.

7.8 How do I restart the model?

To restart the model the `xmlchange` command can be used:

```
cd $CASEROOT
./xmlchange CONTINUE_RUN=TRUE
./case.submit
```

In this case, CIME makes the required changes to the model namelist files (`model_configure` and `input.nml`) and also copies the files from the `RESTART` to the `INPUT` directory.

Note: If there are restart files belonging to multiple time snapshots (i.e. with 20190829.060000., 20190829.120000. prefixes if written every 6-hours), CIME gets the latest one (the files with `20190829.120000.` prefix) automatically.

The restart interval can also be changed to a 6 hourly interval as follows:

```
cd $CASEROOT
./xmlchange REST_OPTION=nhours
./xmlchange REST_N=6
```

Note: The default value of the **restart_interval** namelist option is zero (0), and the model writes a single restart file at the end of the simulation.

The following example demonstrates the 48 hour model simulation split into an initial 24-hour simulation with a cold start plus an additional 24-hour simulation with warm start.

The initial 24 hours simulation:

```
cd $CASEROOT
./xmlchange STOP_OPTION=nhours
./xmlchange STOP_N=24
./case.submit
```

and restart the model for 24 hours simulation:

```
cd $CASEROOT
./xmlchange CONTINUE_RUN=TRUE
./case.submit
```

Note: The restart run length can be changed using the `xmlchange` command and setting `STOP_N` and `STOP_OPTION`.

The model outputs always start from 000 (e.g., sfcf000.nc, atmf000.nc), and don't depend on the model start time and method (warm or cold start).

7.9 How do I change a namelist option for chgres_cube or the model?

From the case directory running `./preview_namelists` will generate the namelists for the run. This is normally run by `case.submit`, but you can also run it from the command line after running the command `case.setup`. Run it once before editing `user_nl_ufsatm` and examine `input.nml` to see the default value, then edit `user_nl_ufsatm` and run it again to see the change.

Typical usage of `preview_namelists` is simply:

```
./preview_namelists
```

The `input.nml` will be generated under the directory CaseDocs,

```
ls CaseDocs
atm_in  config.nml  input.nml  itag.tmp  model_configure
```

To set model namelist options in CIME, edit the file `user_nl_ufsatm` in the case and add the change(s) as name-value pairs. For example:

```
!----------------------------------------------------------------------------------
! This file can be used to change namelist options for:
! - Chgres
```

(continues on next page)

(continued from previous page)

```
! - UFS MR-Weather Model
! - NCEP Post
!
! Users should add all user-specific namelist changes below in the form of
!  namelist_var = new_namelist_value
!
! To change the namelist variables that are defined as multiple times under
! different namelist groups
!  namelist_var@namelist_group = new_namelist_value
!
! Following is the list of namelist variables that need to be accessed by
! specifying the namelist groups:
!
! alpha@nam_physics_nml
! alpha@test_case_nml
! avg_max_length@atmos_model_nml
! avg_max_length@gfs_physics_nml
! debug@atmos_model_nml
! debug@gfs_physics_nml
! icliq_sw@gfs_physics_nml
! icliq_sw@nam_physics_nml
! iospec_ieee32@fms_nml
! iospec_ieee32@fms_io_nml
! ntiles@fv_core_nml
! ntiles@nest_nml
! read_all_pe@fms_io_nml
! read_all_pe@fms_nml
! regional@chgres
! regional@fv_core_nml
!---------------------------------------------------------------------------------
do_skeb = T
```

Then run `./case.submit`. This will update the namelist and submit the job.

If you want to review what you have done before you submit the case, you can run `./preview_namelists` and then examine the namelist(s) in the run directory or the case subdirectory `CaseDocs/`.

Some variables are tied to xml in the case and can only be changed via the `xmlchange` command. Attempting to change them by editing the file `user_nl_ufsatm` may generate an error. The parameters that need to be changed via `xmlchange` are defined in `namelist_definition_ufsatm.xml`.

```
cd src/model/FV3/cime/cime_config
cat namelist_definition_ufsatm.xml | grep "modify_via_xml"
<entry id="ccpp_suite" modify_via_xml="CCPP_SUITES">
<entry id="start_year" modify_via_xml="RUN_STARTDATE">
<entry id="start_month" modify_via_xml="RUN_STARTDATE">
<entry id="start_day" modify_via_xml="RUN_STARTDATE">
<entry id="start_hour" modify_via_xml="START_TOD">
<entry id="start_minute" modify_via_xml="START_TOD">
<entry id="start_second" modify_via_xml="START_TOD">
<entry id="nhours_fcst" modify_via_xml="STOP_N">
<entry id="restart_interval" modify_via_xml="REST_N”>
```

The changes are required to ensure consistency between the model configuration and the CIME.

Warning: The `user_nl_ufsatm` file is also used to control namelist options for chgres_cube and NCEP-Post. Different namelist groups in the model namelist and the pre-, post-processing tools could have the same namelist variable. In this case, just using the namelist variable causes failures in the automated namelist generation. The following is the list of namelist variables that needs to be used along with their group name.

- alpha@nam_physics_nml
- alpha@test_case_nml
- avg_max_length@atmos_model_nml
- avg_max_length@gfs_physics_nml
- debug@atmos_model_nml
- debug@gfs_physics_nml
- icliq_sw@gfs_physics_nml
- icliq_sw@nam_physics_nml
- iospec_ieee32@fms_nml
- iospec_ieee32@fms_io_nml
- ntiles@fv_core_nml
- ntiles@nest_nml
- read_all_pe@fms_io_nml
- read_all_pe@fms_nml
- regional@chgres
- regional@fv_core_nml

7.10 How do I turn on stochastic physics?

There are three types of stochastic physics supported with this release: SPPT, SHUM, and SKEB. They can be used together or separately, and their use is controlled by setting model namelist options DO_SPPT, DO_SHUM, DO_SKEB to true or false. These options are set to false by default for all supported compsets and physics suites.

In addition to the namelist variables that turn stochastic physics on or off, there are several variables that control the behavior of the physics. Those are explained in the Stochastic Physics User's Guide.

In order to set variables DO_SPPT, DO_SHUM, DO_SKEB to true in the model namelist, as well as to set the values of the variables that customize the stochastic physics, please see FAQ entry *How do I change a namelist option for chgres_cube or the model?*

7.11 Can I customize the UPP output?

Starting with v1.1.0, you may customize your output following the instructions in Section 4.2.3.

7.12 How do I find out which platforms are preconfigured for the MR Weather App?

Preconfigured machines are platforms that have machine specific files and settings scripts and should run the MR Weather Application **out-of-the-box** (other than potentially needing to download input files). Preconfigured platforms are usually listed by their common site-specific name.

To see the list of preconfigured, out of the box platforms, issue the following commands:

```
cd $SRCROOT/cime/scripts
./query_config --machines
```

The output will contain entries like the following:

```
cheyenne (current) : NCAR SGI platform, os is Linux, 36 pes/node, batch system is PBS
('      os              ', 'LINUX')
('      compilers       ', 'intel,gnu,pgi')
('      mpilibs         ', ['mpt', 'openmpi'])
('      pes/node        ', '36')
('      max_tasks/node  ', '36')
```

7.13 What are the compsets and physics suites supported in this release?

There are two compsets supported in this release: GFSv15p2 and GFSv16beta, corresponding to the physics suites associated with the operational GFS v15 model and with the developmental physics for the future implementation of GFS v16. However, there are four physics suites supported for this release: GFSv15p2, GFSv15p2_no_nsst, GFSv16beta, and GFSv16beta_no_nsst. The difference between a suite and its no_nsst counterpart is that the no_nsst suites do not include the Near Sea Surface Temperature (NSST) ocean parameterization. Instead, they employ a simple ocean scheme (sfc_ocean) that keeps the sea surface temperature constant throughout the forecast. Compset GFSv15p2 can use either the GFSv15p2 suite or the GFSv15p2_no_nsst suite. Similarly, Compset GFSv16beta can use either the GFSv16beta suite or the GFSv16beta_no_nsst suite. The choice is made based on the format of the initial conditions file. When GRIB2 format is chosen, the non_nsst suites are used. When NEMSIO or netCDF format is chosen, the suites with NSST are chosen. These differences are needed because the GRIB2 files do not have all the fields needed to initialize the operational NSST parameterization.

7.14 How can I change number of task used by chgres_cube or UPP (NCEP-Post)?

By default, CIME automatically sets number of tasks used by `chgres_cube` and NCEP-Post (*UPP*) based on the resolution of the created case using following logic:

- **chgres_cube**

  It requires that number of task used by chgres_cube need to be divided evenly with the number of tiles (6).

  - C96: closest number of task to tasks_per_node, which can be divided by 6
  - C192: closest number of task to tasks_per_node, which can be divided by 6
  - C384: closest number of task to 2 * tasks_per_node, which can be divided by 6
  - C768: closest number of task to 4 * tasks_per_node, which can be divided by 6

- **UPP**

  - C96: tasks_per_node
  - C192: tasks_per_node
  - C384: 2 * tasks_per_node
  - C768: 4 * tasks_per_node

The number of tasks will increase along with the increased horizontal resolution due to the memory consumption of the pre-processing tool and **tasks_per_node** is defined for the each platform using **MAX_MPITASKS_PER_NODE** element (i.e. 36 for NCAR Cheyenne and 48 for TACC Stampede2).

To change the values set automatically by CIME-CSS, the `xmlchange` command can be used:

```
cd $CASEROOT
./xmlchange task_count=72 --subgroup case.chgres
```

This command will change the number of tasks used by chgres_cube to 72. If the user wants to change the number of task for NCEP-Post, the subgroup option needs to set to `case.gfs_post`.

7.15 How can I run the MR Weather App for another date without overriding my previous run?

Before running the App for a second date, you should save your previous run in another directory by moving that directory to a different location.

From the case directory do:

```
RUNDIR = ` ./xmlquery RUNDIR --value`
mv $RUNDIR $RUNDIR.forecastdate
```

7.16 How do I diagnose a failure with a high-resolution run?

One possible source of failure with high-resolution runs is lack of memory. To diagnose if this is the problem, try a low resolution run first.

CHAPTER

EIGHT

ADDITIONAL INFORMATION ABOUT CIME

All compiler flags are defined in `cime/config/ufs/machines/config_compilers.xml`. This file contains all supported compilers and the specific compiler flags to be used for building with that compiler. `cime/config/ufs/machines/config_compilers.xml` contains compiler flags for each target compiler. In general, flags are not system dependent and are valid on all systems. However, there can be machine or OS (useful to generalize over eg MacOS or Cray systems) dependent flags.

All environment variables and module operations are defined in `cime/config/ufs/machines/config_machines.xml`. The environment variables are sorted by machine name and can be subset for compiler, mpilib, debug, etc.

The following describes how the xml files in the `cime/config/ufs/machines/` directory are utilized in the case directory that is created for you when you invoke **create_newcase**.

Note: It is important to point out that for a model component to be CIME CCS compliant, it needs to have a directory `cime_config/` that contains a `buildlib` script that tells CIME how to build that component and a `buildnml` script that tells CIME how to generate namelists for your target component configuration.

- CIMEs `case.setup script` reads the `config_compilers.xml` file and creates a `Macros.make` and `Macros.cmake` file in youe case directory.
- The `Macros.cmake` file is then used by the file `FV3/cime/cime_config/buildlib` to build your model component. `Macros.cmake` is included by file `configure_cime.cmake` and there the compiler flag names are translated to those used by the FV3GFS cmake build. If CCPP is used the ccpp_precompile script is called prior to calling the cmake for the model. Finally gmake is called and all the libraries are built, verbose records of the build are written to the atm.bldlog.timestamp file in the case EXEROOT.

When you create an MR Weather Application case, CIME will create a `$CASEROOT` directory for you. It will also create a `$CASEROOT/SourceMods/src.fv3gfs` directory where you can put in modified source code that you can use for your experiment. The CIME build will look in the `$CASEROOT/SourceMods/src.fv3gfs` directory for any source file matching the name of any source file in the build. (FMS, CCPP, stochastic_physics, FV3) If it finds a match it will use the file in `SourceMods/src.fv3gfs` instead of the matching file in your checked out code sandbox. If a file is removed from `SourceMods/src.fv3gfs` then the next build will again use the original file your checked out code base. For best interaction with the git repository it is recommended that you edit source files in the source tree and do not use the SourceMods mechanism, however keep in mind that source files changed in this way will affect *all* cases associated with the source tree.

CHAPTER

NINE

GLOSSARY

CCPP Model agnostic, vetted, collection of codes containing atmospheric physical parameterizations and suites for use in NWP along with a framework that connects the physics to host models

chgres_cube The preprocessing software used to create initial condition files to “coldstart” the forecast model. The initial conditions are created from either GFS GRIB2 or NEMSIO data.

CIME The Common Infrastructure for Modeling the Earth (CIME - pronounced “SEAM”) provides a Case Control System for configuring, compiling and executing Earth system models, data and stub model components, a driver and associated tools and libraries.

FMS The Flexible Modeling System (FMS) is a software framework for supporting the efficient development, construction, execution, and scientific interpretation of atmospheric, oceanic, and climate system models.

FV3 The GFDL Finite-Volume Cubed-Sphere Dynamical Core (FV3) is a scalable and flexible dynamical core capable of both hydrostatic and non-hydrostatic atmospheric simulations.

GRIB2 The second version of the World Meterological Organization’s (WMO) standard for distributing gridded data.

NCEP National Centers for Environmental Prediction, an arm of the National Weather Service.

NCEPLIBS The NCEP library source code and utilities required for chgres_cube, the UFS Weather Model, and UPP.

NCEPLIBS-external A collection of third-party libraries required to build NCEPLIBS, chgres_cube, the UFS Weather Model, and UPP.

NCL An interpreted language designed specifically for scientific data analysis and visualization. More information can be found at https://www.ncl.ucar.edu.

NEMS The NOAA Environmental Modeling System - a software infrastructure that supports NCEP/EMC’s forecast products.

NEMSIO A binary format for atmospheric model output on the native gaussian grid.

NetCDF A set of software libraries and machine-independent data formats that supports the creation, access, and sharing of array-oriented scientific data.

Stochastic physics A package of stochastic schemes used to represent model uncertainty: SKEB (Stochastic Kinetic Energy Backscatter), SPPT (Stochastically Perturbed Physics Tendencies), and SHUM (Specific Humidity)

Suite A collection of primary physics schemes and interstitial schemes that are known to work well together

UFS A Unified Forecast System (UFS) is a community-based, coupled comprehensive Earth system modeling system. The UFS numerical applications span local to global domains and predictive time scales from sub-hourly analyses to seasonal predictions. It is designed to support the Weather Enterprise and to be the source system for NOAA’s operational numerical weather prediction applications

UPP The Unified Post Processing System, developed at NCEP and used operationally for models maintained by NCEP. The UPP has the capability to post-process output from a variety of NWP models, including FV3.

Weather Model A prognostic model that can be used for short- and medium-range research and operational forecasts. It can be an atmosphere-only model or be an atmospheric model coupled with one or more additional components, such as a wave or ocean model.

INDEX