User's Guide for the NCEP Unified Post Processor (UPP) Version 4

Contents

Acknowledgments:
The adaptation of the original WRF Post Processor package and User's Guide (by Mike Baldwin of NSSL/CIMMS and Hui-Ya Chuang of NCEP/EMC) was done by Lígia Bernardet (NOAA/ESRL/DTC) in collaboration with Dusan Jovic (NCEP/EMC), Robert Rozumalski (COMET), Wesley Ebisuzaki (NWS/HQTR), and Louisa Nance (NCAR/RAL/DTC). Upgrades to WRF Post Processor versions 2.2 and higher were performed by Hui-Ya Chuang, Dusan Jovic and Mathew Pyle (NCEP/EMC). Transitioning of the documentation from the WRF Post Processor to the Unified Post Processor was performed by Nicole McKee (NCEP/EMC), Hui-ya Chuang (NCEP/EMC), and Jamie Wolff (NCAR/RAL/DTC). Implementation of the Community Unified Post Processor was performed by Tricia Slovacek, Kate Fossell, and Tracy Hertneky (NCAR/RAL/DTC).

NCEP Unified Post Processor (UPP)

UPP Introduction

The NCEP Unified Post Processor has replaced the WRF Post Processor (WPP). The UPP software package is based on WPP but has enhanced capabilities to post-process output from a variety of NWP models, including WRF-NMM, WRF-ARW, Nonhydrostatic Multi-scale Model on the B grid (NMMB), Global Forecast System (GFS), Climate Forecast System (CFS), and the global Finite-Volume Cubed Sphere dynamical core (FV3). At this time, community user support is only provided for the WRF-ARW and FV3 systems.

In addition to the option to output fields on the model's native vertical levels, UPP interpolates output from the model's native grids to National Weather Service (NWS) standard levels (pressure, height, etc.) and standard output grids (AWIPS, Lambert Conformal, polar-stereographic, etc.) in NWS and World Meteorological Organization (WMO) GRIB format.

UPP incorporates the Joint Center for Satellite Data Assimilation (JCSDA) Community Radiative Transfer Model (CRTM) to compute model derived brightness temperature (TB) for various instruments and channels. This additional feature enables the generation of a number of simulated satellite products including GOES and AMSRE products for WRF-NMM, Hurricane WRF (HWRF), WRF-ARW and GFS.

Software Installation/Getting Started

UPP Software Requirements

Before installing the UPP code, it is necessary to ensure that you have the required libraries available on your system. These libraries include:

- Unidata's NetCDF library

  https://www.unidata.ucar.edu/software/netcdf/docs/getting_and_building_netcdf.html

- The NCEP libraries for the UPP application

  https://github.com/NCAR/NCEPlibs

  - For instructions on building the NCEP libraries required for UPP, please refer to the README document in the NCEPlibs directory.

  Note: *These are specific versions of the NCEP libraries maintained by NCAR/DTC and other versions of NCEPlibs may not work.*

The UPP has some sample visualization scripts included to create graphics using:

- GrADS (http://cola.gmu.edu/grads/gadoc/gadoc.php)
- GEMPAK (http://www.unidata.ucar.edu/software/gempak/index.html)

Note:*These are not part of the UPP installation and need to be installed separately if one would like to use either plotting package.*

UPP has been tested on LINUX platforms (with PGI, Intel and GFORTRAN compilers).

Obtaining the UPP Code

The user can obtain the code in one of to ways:

1. The UPP package is now available on Github. To create a local copy of the remote UPP repository on your computer and get the CRTM submodules:

   git clone -b "release-tag-name" --recurse-submodules https://github.com/NOAA-EMC/EMC_post UPPV4.1

   where, "*release-tag-name*" is the release tag you wish to clone (e.g. for UPPV4.1 use the release tag dtc_post_v4.1.0).

This will clone the specified release of the ***EMC_post*** repository into a directory called ***UPPV4.1***.

1. Download the release tarfile from the UPP website:

   https://dtcenter.org/community-code/unified-post-processor-upp/download

   Once the tar file is obtained, ***gunzip*** and ***untar*** the file.

   tar -xzvf UPPV4.1.tar.gz

   This command will create a directory called UPPV4.1.

Note: *Always obtain the latest version of the code if you are not trying to continue a pre-existing project.*

The remainder of this documentation assumes the top directory of the UPP is called UPPV4.1

UPP Directory Structure

Under the main directory of ***UPPV4.1*** reside the following relevant subdirectories (***** indicates directories that are created after the configuration step):

exec*: Contains the ***unipost*** executable after successful compilation

include*: Source include modules built/used during compilation of UPP

lib*: Libraries built/used by UPP that are separate from NCEPlibs

parm: Contains parameter files, which can be modified by the user to control how the post processing is performed.

scripts: Contains sample run scripts to process ***wrfout*** and ***fv3*** history files.
- **run_unipost**: run ***unipost***.
- **run_unipostandgempak**: run ***unipost*** and GEMPAK to plot various fields.
- **run_unipostandgrads**: run ***unipost*** and GrADS to plot various fields.
- **run_unipost_frames**: run ***unipost*** on a single file containing multiple forecast times. (WRF only)
- **run_unipost_gracet**: run ***unipost*** on forecast files with non-zero minutes/seconds. (WRF only)
- **run_unipost_minutes**: run ***unipost*** for sub-hourly forecast files. (WRF only)
- **run_unipostandgrads_global**: run ***unipost*** and GrADS for global ***wrfout*** files; results in single GRIB file. (WRF only)

sorc: Contains source codes for:

- **arch**: Machine dependent configuration build scripts used to construct *configure.upp*
- **comlibs**: Contains source code subdirectories for the UPP libraries not included in NCEPlibs
  - **crtm2**: Community Radiative Transfer Model library
  - **wrfmpi_stubs**: Contains some ***C*** and ***FORTRAN*** codes to generate ***libmpi.a*** library used to replace MPI calls for serial compilation.
  - **xml**: XML support for the GRIB2 parameter file
- **ncep_post.fd**: Source code for ***unipost***

Installing the UPP Code

Before installing UPP, the following environment variables must be set:

setenv NETCDF /path/to/netcdf
setenv NCEPLIBS_DIR /path/to/NCEPlibs

To reference the netCDF libraries, the configure script checks for an environment variable (***NETCDF***) first, then the system default (***/user/local/netcdf***), and then a user supplied link (***./netcdf_links***). If none of these resolve a path, the user will be prompted by the configure script to supply a path. To reference the NCEP libraries, the configure script checks for an environment variable (***NCEPLIBS_DIR***).

Type configure, and provide the required info. For example:

./configure

You will be given a list of choices for your computer.
(Please note that at this time, the option to build serially does not work)

Choices for LINUX operating systems are as follows:
1. Linux x86_64, PGI compiler (serial)
2. Linux x86_64, PGI compiler (dmpar)
3. Linux x86_64, Intel compiler (serial)
4. Linux x86_64, Intel compiler (dmpar)
5. Linux x86_64, Intel compiler, SGI MPT (serial)
6. Linux x86_64, Intel compiler, SGI MPT (dmpar)
7. Linux x86_64, gfortran compiler (serial)
8. Linux x86_64, gfortran compiler (dmpar)

Choose one of the ***dmpar*** configure options listed. If the serial option is chosen during configuration, an error staement will be printed. Check the ***configure.upp*** file created and edit for compile options/paths, if necessary. For debug flag settings, the configure script can be run with a –d switch or flag.

To compile UPP, enter the following command:

./compile >& compile_upp.log &

When compiling with distributed memory (serial) this command should create 2 (3) UPP libraries in ***UPPV4.1/lib/*** (***libCRTM.a***, (***libmpi.a***), ***libxmlparse.a***) and the UPP executable in ***UPPV4.1/exec/*** (***unipost.exe***).

To remove all built files, as well as the ***configure.upp***, type:

./clean

This action is recommended if a mistake is made during the installation process or a change is made to the configuration or build environment. There is also a ***clean –a*** option which will revert back to a pre-install configuration.

*Note: For building UPPV4.1 on operational NCEP machines (hera/jet/wcoss), just type **./compile machine_name** (e.g. **./compile hera**). This is an option for UPPV4.1 only and will not work for any previous release versions.*

UPP Functionalities

The UPP,

- is compatible with WRF v3.7 and higher for Ferrier physics (UPPV3.0+) or WRF v3.5 and higer (UPPV2.1).
- can be used to post-process WRF-ARW, WRF-NMM, NMMB, GFS, CFS, and FV3 forecasts (community support only provided for WRF-ARW and global FV3 forecasts).
- can ingest WRF history files (wrfout*) in netCDF format.
- can ingest FV3 history files (e.g. dyn*/phy* or gfs*) in binarynemsiompiio and netCDF format.

Unipost Functionalities

- Interpolates the forecasts from the model's native vertical coordinate to NWS standard output levels (e.g., pressure, height) and computes mean sea level pressure. If the requested parameter is on a model's native level, then no vertical interpolation is performed.
  - Computes diagnostic output quantities (e.g., convective available potential energy, helicity, relative humidity). A full list of fields that can be generated by ***unipost*** is provided for GRIB1 and GRIB2.
  - Outputs the results in NWS and WMO standard GRIB1 or GRIB2 format (for GRIB documentation, see http://www.nco.ncep.noaa.gov/pmb/docs/).
    - FV3 only available for GRIB2 output.
  - Destaggers the WRF-ARW forecasts from a C-grid to an A-grid.
  - Except for new capabilities of post processing GFS/CFS and additions of many new variables, UPP uses the same algorithms to derive most existing variables as were used in WPP. The only three exceptions/changes from the WPP are:
    - Computes RH w.r.t. ice for GFS, but w.r.t. water for all other supported models. WPP computed RH w.r.t. water only.
    - The height and wind speed at the maximum wind level is computed by assuming the wind speed varies quadratically in height in the location of the maximum wind level. The WPP defined maximum wind level at the level with the maximum wind speed among all model levels. The static tropopause level is obtained by finding the lowest level that has a temperature lapse rate of less than 2 K/km over a 2 km depth above it. The WPP defined the tropopause by finding the lowest level that has a mean temperature lapse rate of 2 K/km over three model layers.

Setting up the WRF or FV3 model to interface with UPP

The unipost program is currently set up to read a large number of fields from the WRF and FV3 model history files. This configuration stems from NCEP's need to generate all of its required operational products. When using the netCDF or NEMS binary read, this program is configured such that it will run successfully even if an expected input field is missing from the WRF or FV3 history file as long as this field is not required to produce a requested output field. If the pre-requisites for a requested output field are missing from the WRF or FV3 history file, unipost will abort at run time.

Take care not to remove fields from the wrfout or gfs files which may be needed for diagnostic purposes by the UPP package. For example, if isobaric state fields are requested, but the pressure fields on model interfaces (i.e. P_HYD for WRF-ARW) are not available in the history file, unipost will abort at run time.

In general, the default fields available in the wrfout or gfs files are sufficient to run UPP. The fields written to the WRF history file are controlled by the settings in the Registry . For WRF, see the Registry.EM, Registry.EM_COMMON files in the Registry subdirectory of the main WRFV4 directory.
Note: *It is necessary to re-compile the WRF model source code after modifying the Registry file.*

UPP is written to process a single forecast hour, therefore, having a single forecast per output file is optimal. However, for WRF based forecasts, UPP can be run across multiple forecast times in a single output file to extract a specified forecast hour.

UPP Control File Overview

The user interacts with unipost through the control file by specifying the fields and levels desired for output. These files are different depending on whether GRIB1 or GRIB2 output is requested.

Each control file is composed of a header and a body. The header specifies the output file information. The body allows the user to select which fields and levels to process.

GRIB1 control file

Note: *This section pertains to outputting GRIB1 format (WRF only).*

The header of the ***parm/wrf_ cntrl.parm*** file contains the following variables:

- **KGTYPE**: defines output grid type, which should always be 255.
- **IMDLTY**: identifies the process ID for AWIPS.
- **DATSET**: defines the prefix used for the output file name. Currently set to "WRFPRS". Note: the run_* scripts assume "WRFPRS" is used for WRF runs.

The body of the ***wrf_ cntrl.parm*** file is composed of a series of line pairs similar to the following:

```
(PRESS ON MDL SFCS ) SCAL=( 3.0)
L=(11000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000)
```

where,

- The top line specifies the variable (e.g. PRESS) to process, the level type (e.g. ON MDL SFCS) a user is interested in, and the degree of accuracy to be retained (SCAL=3.0) in the GRIB output.
  - SCAL defines the precision of the data written out to the GRIB format. Positive values denote decimal scaling (maintain that number of significant digits), while negative values describe binary scaling (precise to 2^{SCAL}; i.e., SCAL=-3.0 gives output precise to the nearest 1/8). Because ***copygb*** is unable to handle binary precision at this time, negative numbers are discouraged.
  - A list of all possible output fields for ***unipost*** is provided in the GRIB1 ouput table. This table provides the full name of the variable in the first column and an abbreviated name in the second column. The abbreviated names are used in the control file. Note that the variable names also contain the type of level on which they are output. For instance, temperature is available on "model surface" and "pressure surface".
- The second line specifies the levels on which the variable is to be posted. In this case, 0 indicates no output at this level and 1 indicates output the variable specified on the top line at the level specified by the position of the digit and the type of level defined for this variable. For flight/wind energy fields, a 2 may be specified, such that 2 requests AGL and 1 requests MSL.

Example: The sample control file ***parm/wrf_ cntrl.parm*** has the following entry for surface dew point temperature:

(SURFACE DEWPOINT) SCAL=(4.0)
L=(00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000)

Based on this entry, surface dew point temperature will not be output by unipost. To add this field to the output, modify the entry to read:

(SURFACE DEWPOINT) SCAL=(4.0)
L=(10000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000)

GRIB2 control file

Note: *This section describes the control file for outputting GRIB2 format.*

For outputting GRIB2 format, a preprocessing step is required by the user to convert the xml file ***parm/postcntrl.xml*** to a flat text file ***postxconfig-NT.txt***. This new flat file is quicker to process than the old xml file. First, the user will need to edit the ***postcntrl.xml*** file to declare which fields and levels are to be output from unipost. The header of the ***parm/postcntrl.xml*** includes a number of variables describing the output

file information. Of particular interest is the <datset> parameter which specifies the prefix for the output filename.
The body of the ***postcntrl.xml*** file is composed of a series of parameter blocks similar to the following:

<param>
<shortname>TMP_ON_ISOBARIC_SFC</shortname>
<pname>TMP</pname>
<level>10000. 25000. 50000. 70000. 85000. 100000.</level>
<scale>3.0</scale>
</param>

where,

- The <shortname> is the character name describing the product or field.
- The <pname> is the field type abbreviation used by the grib2 libraries.
- The <level> is a list of levels desired to be oUPP_Users_guide_V4.1.lyxutput (included only for fields available on vertical coordinate levels).
- The <scale> indicates the the grib precison packing.

As an optional step to ensure that the user-edited xml files are error free, XML stylesheets (***parm/EMC_POST_CTRL_S*** and ***EMC_POST_Avblflds_Schema.xsd***) can be used to validate both the ***postcntrl.xml*** and ***post_avblflds.xml*** files, respectively. Confirmation of validation will be given (e.g. postcntrl.xml validates) or otherwise return errors if it does not match the schema. To run the validation:

xmllint --noout --schema EMC_POST_CTRL_Schema.xsd postcntrl.xml
xmllint --noout --schema EMC_POST_Avblflds_Schema.xsd post_avblflds.xml

Once the xmls are validated, the user will need to generate the flat file. Edit the ***parm/makefile_comm*** if necessary to point to another directory or xml. The makefile will call the perl program ***parm/POSTXMLPreprocessor.pl*** to generate the post flat file ***postxconfig-NT.txt***. Generate the flat file:

make

Controlling which variables unipost outputs

To output a field, the body of the control file needs to contain an entry for the appropriate variable and output for this variable must be turned on for at least one level (see next section: "*Controlling which levels*

unipost outputs"). If an entry for a particular field is not yet available in the control file, a section may be added to the control file with the appropriate entries for that field.

Controlling which levels unipost outputs

The second line of each pair determines which levels unipost will output. For GRIB1, output on a given level is turned off by a "0" or turned on by a "1" in the ***wrf_cntrl.parm***. For GRIB2, the desired levels are listed in the <level> list in the ***postcntrl.xml***.

- For isobaric output, 46 levels are possible, from 2 to 1000 hPa (*2, 5, 7, 10, 20, 30, 50, 70 mb and then every 25 mb from 75 to 1000 mb*). The complete list of levels is specified in ***sorc/ncep_post.fd/CTLBLK.f***.
  - Modify specification of variable LSMDEF to change the number of pressure levels: LSMDEF=47
  - Modify specification of SPLDEF array to change the values of pressure levels: (/200.,500.,700.,1000.,2000.,3000. &,5000.,7000.,7500.,10000.,12500.,15000.,17500.,20000., ... /)
- For model-level output, all model levels are possible, from the highest to the lowest.
- When using the Noah LSM, the soil layers are 0-10 cm, 10-40 cm, 40-100 cm, and 100-200 cm.
- When using the RUC LSM, the soil levels are 0 cm, 1 cm, 4 cm, 10 cm, 30 cm, 60 cm, 100 cm, 160 cm, and 300 cm. For the RUC LSM, it is also necessary to include the five additional output levels in the relevant control file to output 9 instead of 4 levels. (For the old RUC LSM, there are only 6 layers and if using this, you will need to change "RUC LSM" from 9 to 6 in the WRFPOST.f routine.)
- When using Pliem-Xiu LSM, there are two layers: 0-1 cm, 1-100 cm
- For low, mid, and high cloud layers, the layers are $\geq$642 hPa, $\geq$350 hPa, and $<$350 hPa, respectively.
- For PBL layer averages, the levels correspond to 6 layers with a thickness of 30 hPa each.
- For flight level, the levels are 30 m, 50 m, 80 m, 100 m, 305 m, 457 m, 610 m, 914 m, 1524 m, 1829 m, 2134 m, 2743 m, 3658 m, 4572 m, 6000 m, 7010 m.
- For AGL radar reflectivity, the levels are 4000 and 1000 m (see Appendix A for details).
- For surface or shelter-level output, only the first position of the line needs to be turned on.

Running UPP

Seven scripts for running the UPP package are included in the tar file:
run_unipost
run_unipostandgrads
run_unipostandgempak
run_unipost_frames
run_unipost_gracet
run_unipost_minute
run_unipostandgrads_global

Before running any of the above listed scripts, perform the following instructions:

1. ***cd*** to your ***DOMAINPATH*** directory. This is your working directory for the run.
2. Make a directory to put the UPP results.
   mkdir postprd
3. Make a directory for staging a copy of the desired control file.
   mkdir parm

4. Copy over the relevant control file to your working directory to customize ***unipost***.

   For Grib1, copy the default ***UPPV4.1/parm/wrf_cntrl.parm*** file.

   For Grib2, copy the default ***UPPV4.1/parm/postxconfig-NT-WRF.txt*** for WRF or ***UPPV4.1/parm/postxconf NT-GFS.txt*** for FV3.

5. For Grib1, edit the ***wrf_cntrl.parm*** file to reflect the fields and levels you want unipost to output.

   For Grib2, edit the ***postcntrl.xml*** file and re-make the ***postxconfig-NT.txt*** file.

6. Copy over the (***UPPV4.1/scripts/run_unipost****) script of your choice to the ***postprd/***.

7. Edit the run script as outlined below. Once these directories are set up and the edits outlined below are complete, the scripts can be run interactively from the ***postprd*** directory by simply typing the script name on the command line.

Overview of the scripts to run the UPP

Note: *It is recommended that the user refer to the* ***run_unipost**** *scripts in the* ***scripts/*** *while reading this overview.*

All user modified variables are contained at the top of the ***run_unipost**** script in one user-edit section, along with a brief description. Descriptions below follow the ***run_unipost*** script.

1. Set up basic path variables:

   TOP_DIR : Top level directory for source code (***UPPV4.1***)

   DOMAINPATH : Working directory for this run

   UNIPOST_HOME : Location of the ***UPPV4.1*** build directory

   POSTEXEC : Location of the ***UPPV4.1*** executables

   modelDataPath : Location of the model output data files to be processed (e.g. "***wrfprd/***" for WRF-based runs).

   paramFile : Name and location of grib1 control file (***wrf_cntrl.parm***) that lists desired fields for GRIB1 output. Template in ***UPPV4.1/parm/***

   txtCntrlFile : Name and location of ***postxconfig-NT.txt*** file that lists desired fields for GRIB2 format. This file is generated by the user following the steps listed above in the GRIB2 Control File section.

Note: *For WRF, the scripts are configured such that* ***unipost*** *expects the WRF history files (****wrfout**** files) to be in* ***wrfprd/****, the* ***wrf_cntrl.parm*** *(Grib1) or* ***postxconfig-NT.txt*** *(Grib2) file to be in* ***parm/*** *and the postprocessor working directory to be called* ***postprd/****, all under* ***DOMAINPATH****. A similar sctructure is expected for FV3.*

This set up is for user convenience to have a script ready to run, paths may be modified but be sure to check the run script to make sure settings are correct.

1. Specify dynamic core being run ("ARW" or "FV3")

   dyncore : What model core is used ("ARW" or "FV3")

2. Specify the format for the input model files and output UPP files.

   inFormat : Format of the model data

   arw – "netcdf"

   fv3 – "binarynemsiompiio or netcdf"

outFormat : Format of output from UPP

grib (WRF only)

grib2

3. Specify the forecast cycles to be post-processed

   startdate : Forecast start date (YYYYMMDDHH)

   fhr : First forecast hour to be post-processed

   lastfhr : Last forecast hour to be post-processed

   incrementhr : Increment (in hours) between forecast files

   * Do not set to 0 or the script will loop continuously *

4. Set up how many domains will be post-processed (used by WRF only)

   domain_list : List of domains for run (e.g. "d01 d02")

5. Set/uncomment the run command for your system. (i.e. serial, mpirun, etc).Need to add how to obtain the code via github download

   RUN_COMMAND : System run command for serial or parallel runs

   - The default execution command in the distributed scripts is for a single processor:
     ./unipost.exe > unipost_${domain}.${fhr}.out 2>&1
   - To run unipost using mpi (dmpar compilation), the command line should be:
     >> LINUX-MPI systems: ***mpirun -np N unipost.exe > outpost 2>&1***
     (***Note:*** *on some systems a host file also needs to be specified:* – ***machinefile "host"***)
     >> IBM: ***mpirun.lsf unipost.exe < itag > outpost***
     >> SGI MPT: ***mpiexec_mpt unipost.exe < itag > outpost***

6. Set naming convention for prefix and extension of output file name

   - ***comsp*** is the initial string of the output file name (by default it is not set and the prefix of the output file will be the string set in ***wrf_cntrl.parm DATSET***, if set it will concatenate the setting to the front of the string specified in ***wrf_cntrl.parm DATSET***
   - ***tmmark*** is used for the file extension (in ***run_unipost***, *tmmark=tm00*, if not set, it is set to .GrbF)

Since V3.0, the itag that will be read in by ***unipost.exe*** from stdin (unit 5) is generated automatically in the ***run_unipost*** script based on the user-defined options above. It should not be necessary to edit this. For description purposes, the namelist (***itag***) contains 5 (6) lines for WRF (FV3):

1. Name of the WRF or FV3 (pressure level) output file to be posted.
2. Format of WRF or FV3 model output (netcdf, binarynemsiompiio).
3. Format of UPP output (GRIB1 or GRIB2)
4. Forecast valid time (not model start time) in WRF or FV3 format (the forecast time desired to be post-processed).
5. Dynamic core used (NCAR or GFS).
6. Name of the FV3 (surface) output file to be post-processed.

7. Name of Grib2 control file (postxconfig-NT.txt)

Note: *If the third line (i.e., UPP output type) is not set, UPP will default the output file format to "grib1". Please note that output for FV3 only supports GRIB2.*

If scripts ***run_unipostandgrads*** or ***run_unipostandgempak*** are used, additional steps are taken to create image files (see Visualization section below).

Upon a successful run, ***unipost*** will generate output files ***WRFPRS_d****nn.hh* (***GFSPRS****.hh*), in the post-processor working directory, where "*nn*" refers to the domain id and "*hh*" denotes the forecast hour. In addition, the script ***run_unipostandgrads*** will produce a suite of png images named ***variablehh_GrADS.png***, and the script ***run_unipostandgempak*** will produce a suite of gif images named ***variablehh.gif***.

If the run did not complete successfully, a log file in the post-processor working directory called ***unipost_dnn.hh.out*** (***unipost.hh.out***), where "*nn*" is the domain id and "*hh*" is the forecast hour, may be consulted for further information.

Note: FV3 output is on a Guassian grid. To interpolate to a lat/lon or other projection, use wgrib2 (see *Examples of wgrib2* below).

Examples of copygb (GRIB1 only)

Copygb can be used to horizontally interpolate forecasts from the native grid to a standard AWIPS or user-defined grid (for information on AWIPS grids, see http://www.nco.ncep.noaa.gov/pmb/docs/on388/tableb.html). It is a flexible program that can accept several command line options specifying details of how the horizontal interpolation from the native grid to the output grid should be performed. Output from copygb is in NWS and WMO standard GRIB1 format (for GRIB documentation, see http://www.nco.ncep.noaa.gov/pmb/docs/).

Copygb is available on Github and can be obtained by cloning the repository:

git clone https://github.com/NCAR/copygb
This will create a local repository called copygb. Instructions on how to build copygb, along with examples

for running the executable can be found in the ***README*** document.

Examples of wgrib2 (GRIB2 only)

Wgrib2 is a versatile program that has the ability to convert grib2 files from one grid to another for various user-defined grids as well as pre-defined NCEP grids. Complete documentation on grid specification with examples of re-gridding for all available grid definitions can be found at:
https://www.cpc.ncep.noaa.gov/products/wesley/wgrib2/new_grid.html

Since unipost output from FV3GFS is on a Gaussian grid, wgrib2 can be used for interpolation to a lat-lon, or other user defined grid. An example of how operations uses wgrib2 to interpolate to a lat-lon grid is given below.

```
wgrib2 INPUT_FILE -set_grib_type same -new_grid_winds earth |
-new_grid_interpolation bilinear |
-if ":(CRAIN|CICEP|CFRZR|CSNOW|ICSEV):" -new_grid_interpolation neighbor -fi |
-set_bitmap 1 -set_grib_max_bits 16 |
-if ":(APCP|ACPCP|PRATE|CPRAT):" -set_grib_max_bits 25 -fi |
-if ":(APCP|ACPCP|PRATE|CPRAT|DZDT):" -new_grid_interpolation budget -fi |
-new_grid "latlon 0:1440:0.25 90:721:-0.25" OUTPUT_FILE_0p25
```

where,

Line 1: Define the input file name, set the same grib packing as input file, and define the wind orientation to be relative to the earth (U-wind eastward, V-wind northward)

Line 2: Use a bilinear interpolation method for most variables (default method)
Line 3: Use nearest neighbor interpolation method for the specified variables
Line 4: Use bitmap for complex packing and set scaling to not exceed a set number of bits
Line 5: Same as line 4, but different settings for specific variables
Line 6: Use the budget interpolation method for the specified variables
Line 7: Define the grid specifications for an example 0.25 degree lat-lon grid, and define the output file name

Note: *At this time, wgrib2 is not distributed within the UPP. Users may download and install from http://www.cpc.ncep.noaa.gov/products/wesley/wgrib2/.*

Visualization with UPP

GEMPAK

The GEMPAK utility ***nagrib*** is able to decode GRIB files whose navigation is on any non-staggered grid. Hence, GEMPAK is able to decode GRIB files generated by the UPP package and plot horizontal fields or vertical cross sections.

A sample script named ***run_unipostandgempak***, which is included in the scripts directory of the tar file, can be used to run ***unipost*** and plot the following fields using GEMPAK:

- ***Sfcmap_dnn_hh.gif***: mean SLP and 6 hourly precipitation
- ***PrecipType_dnn_hh.gif***: precipitation type (just snow and rain)
- ***850mbRH_dnn_hh.gif***: 850 mb relative humidity
- ***850mbTempandWind_dnn_hh.gif***: 850 mb temperature and wind vectors
- ***500mbHandVort_dnn_hh.gif***: 500 mb geopotential height and vorticity
- ***250mbWindandH_dnn_hh.gif***: 250 mb wind speed isotacs and geopotential height

This script can be modified to customize fields for output. GEMPAK has an online users guide at:
https://www.unidata.ucar.edu/software/gempak/#doc.

In order to use the script ***run_unipostandgempak***, it is necessary to set the environment variable ***GEMEXEC*** to the path of the GEMPAK executables. For example,

setenv GEMEXEC /usr/local/gempak/bin

GrADS

The GrADS utilities ***grib2ctl.pl*** (***g2ctl.pl***) and ***gribmap*** are able to decode GRIB1 (GRIB2) files whose navigation is on any non-staggered grid. These utilities and instructions on how to use them to generate GrADS control files are available from:
http://www.cpc.ncep.noaa.gov/products/wesley/grib2ctl.html (GRIB1)
http://www.cpc.ncep.noaa.gov/products/wesley/g2ctl.html (GRIB2).

The GrADS package is available from: http://cola.gmu.edu/grads/gadoc/gadoc.php.
GrADS has an online User's Guide at: http://cola.gmu.edu/grads/gadoc/users.html
A list of basic commands for GrADS can be found at: http://cola.gmu.edu/grads/gadoc/reference_card.pdf.

A sample script named ***run_unipostandgrads***, which is included in the scripts directory of the Unified Post Processing package, can be used to run ***unipost*** and plot the following fields using GrADS:

- ***Sfcmaphh_dnn_GRADS.png***: mean SLP and 6-hour accumulated precipitation.
- ***850mbRHhh_dnn_GRADS.png***: 850 mb relative humidity

- ***850mbTempandWindhh_dnn_GRADS.png***: 850 mb temperature and wind vectors
- ***500mbHandVorthh_dnn_GRADS.png***: 500 mb geopotential heights and absolute vorticity
- ***250mbWindandHhh_dnn_GRADS.png***: 250 mb wind speed isotacs and geopotential heights

In order to use the script ***run_unipostandgrads***, it is necessary to:

1. Set the environmental variable GADDIR to the path of the GrADS fonts and auxiliary files. For example:

   setenv GADDIR /usr/local/grads/data

2. Add the location of the GrADS executables to the PATH. For example:

   setenv PATH /usr/local/grads/bin:$PATH

3. Link script ***cbar.gs*** to the post-processor working directory. (This scripts is provided in UPP package, and the ***run_unipostandgrads*** script makes a link from ***scripts/*** to ***postprd/***.) To generate the plots above, GrADS script ***cbar.gs*** is invoked. This script can also be obtained from the GrADS library of scripts at http://cola.gmu.edu/grads/gadoc/library.html.

Fields produced by unipost

The 3 tables described below contain documentation regarding the fields that are available for output by UPP for both GRIB1 and GRIB2.

Grib1 Table:

https://dtcenter.org/sites/default/files/community-code/upp-grib1-table_2.pdf

This table lists basic and derived fields currently produced by unipost for grib1. The abbreviated names listed in the second column of each table describe how the fields should be entered in the ***wrf_cntrl.parm***.

Grib2 Table:

https://dtcenter.org/sites/default/files/community-code/upp-grib2-table_0.pdf

This table lists basic and derived fields currently produced by unipost for grib2. The abbreviated names listed in the second column of each table describe how the fields should be entered in the ***postcntrl.xml***.

Grib2 Table (Extended):

https://dtcenter.org/sites/default/files/community-code/upp-grib2-extended-table_1.pdf

This table lists basic and derived fields currently produced by ***unipost*** for grib2 and includes the parameter information which UPP uses for identifying fields for GRIB2.

External References

For CRTM documentation, refer to: https://ftp.emc.ncep.noaa.gov/jcsda/CRTM/CRTM_User_Guide.pdf.

Appendix A: UPPV3.1+ Reflectivity field descriptions

Reflectivities are filled/computed depending on the model core and microphysics options.

UPP uses model derived reflectivity (REFL_10CM from WRF) for model runs using the Thompson microphysics option (mp=8). Other combinations use algorithms within UPP code.

Work is underway to provide more user flexibility when selecting reflectivity computations. For more information on model computed reflectivity, e.g. REFL_10CM, please see model documentation.

Relevant routines for reflectivity. Some or all of these may need to be modified if the user desires to change where/how reflectivity is processed. It is recommended that the user have knowledge of the model output and reflectivity computations before trying to modify the UPP code. Email upp-help@ucar.edu for further questions.

INITPOST* - Separate routines for each different model core (e.g. ARW, FV3, etc.). Reads model fields, e.g. REFL_10CM, REFD_MAX

MDLFLD.f - Computes DBZ or fills DBZ arrays with model computed Reflectivity. - Fills 3-D model level reflectivity array (UPP ID: 250) - Fills 2-D composite reflectivity array (UPP ID: 252)

MDL2AGL.f - Interpolates relevant DBZ array to AGL reflectivity (UPP ID: 253) - Outputs model computed maximum hourly reflectivity (REFD_MAX; UPP ID: 421)

MDL2P.f - Interpolates relevant DBZ array to pressure levels (UPP ID: 251)

Appendix B: Adding a new variable to the UPP code

**** NOTE ****

This document provides general procedures and an example of how to add a new variable to the UPP code. Please keep in mind it may not be an exhaustive step-by-step depending on your particular situation. While we can provide general assistance for adding a new variable, users should be aware that this requires good knowledge of Fortran and thorough understanding of the code.
We encourage users to contact us at upp-help@ucar.edu to make us aware of modifications you are making. In some cases, if we determine the changes you are making may be relevant for operational and/or community purposes, we will be interested in incorporating your changes into the code base for support and future release. We would then work with you to make this possible.

General Content Overview

1. Allocate the field: ALLOCATE.f

   This file is the instantiation or allocation of the variable. *Note that the variables are defined based on the parallel processing capability of UPP - use an example from the file.*

2. Deallocate the field: DEALLOCATE.f

   All good programmers give back their resources when they are done. Please update this routine to return your resource to the system.

3. Declare the new variable: VRBLS2D_mod.f, VRBLS3D_mod.f or VRBLS4D_mod.f

   The variable is declared in one of these module defining files depending on its dimension.

4. Define field for grib1: RQSTFLD.f

   This file contains a list of all possible fields to be output by UPP, corresponding key-word character string user places in wrf_cntrl.parm file, UPP ID for internal code, grib IDs.

5. Read model output: INITPOST.F

   This file is used for reading the model output files.

6. Determine routine for filling variable: e.g SURFCE.f, MDLFLD.f, MDL2P.f, etc.

   This is the place that you will fill the array with the data and call gribit to output the field.

7. Define table/grib2 parameters/ for grib2 output: params_grib2_tbl_new

   This table contains the necessary parameter information for grib2 fields.

8. Define the field for grib2 output: post_avlbflds.xml

   This file is used for defining all available grib2 fields.

9. Define control file entry for output: postcntrl.xml & postxconfig-NT.txt

   These files are used for controlling which fields are output by UPP for grib2.

10. Define output control file: wrf_cntrl.parm

    This file is used for controlling which fields are output by UPP for grib1.

Example Procedure: Steps for adding the new variable 'ACLHF'

- This example illustrates a new variable from the WRF output that will be read into UPP and directly output into the grib output files (i.e. no additional computations/calculations are needed for the field).
- Note that while grib1 procedures are provided, we have moved to grib2 almost exclusively. As such support for grib1 additions is limited.
- Additions to each of the routines are highlighted in green.
- Locations of ***routines*** are in /UPPV4.1/sorc/ncep_post.fd unless specified otherwise.
- A sample wrfout file for the following procedures is available by request. Please submit request to the UPP helpdesk at *upp-help@ucar.edu*.
  - This data is the 6-hr forecast of a WRF initialization of 2009-12-17_12:00:00

New variable to add: ACLHF

```
float ACLHF(Time, south_north, west_east) ;
      ACLHF:FieldType = 104 ;
      ACLHF:MemoryOrder = "XY " ;
      ACLHF:description = "ACCUMULATED UPWARD LATENT HEAT FLUX AT THE SURFACE" ;
      ACLHF:units = "J m-2" ;
      ACLHF:stagger = "" ;
      ACLHF:coordinates = "XLONG XLAT" ;
```

1. Allocate the new variable in ***ALLOCATE_ALL.f*** This file is the instantiation or allocation of the variable. Note that the variables are defined based on the parallel processing capability of UPP - use an example from the file.

   User Procedure:

   - Add in VRBLS2D section as:
     `allocate(aclhf(im,jsta_2l:jend_2u))`

2. De-allocate the variable to give the resources back in ***DEALLOCATE.f*** All good programmers give back their resources when they are done. Please update this routine to return your resource to the system.

   User procedure:

   - Add as:
     `deallocate(aclhf)`

3. Declare the new variable in the appropriate file depending on its dimensions; ***VRBLS2D_mod.f***, ***VRBLS3D_mod.f*** or ***VRBLS4D_mod.f***

   User procedure:

   - **ACLHF** is a 2-dimensional field, so declare it in ***VRBLS2D_mod.f***
   - Add to the end of the first section of allocations as:
     `ACLHF(:,:)`

4. List the new variable in ***RQSTFLD.f*** which includes a list of all possible fields to be output by UPP, as well as the corresponding key-word character string the user places in ***wrf_cntrl.parm*** file, the UPP ID for internal code, and grib IDs. Be sure to pick a unique identifier that is not already used for the new variable. The unique identifier or index are typically assigned in groups - hopefully a community area will be added in the future - or a defined method to avoid overwriting others values. Right now we are using 900's for community contributions.

   Example entry:

```
!  HWRF addition for v_flux as pass through variable:
DATA IFILV(901),AVBL(901),IQ(901),IS(901),AVBLGRB2(901) &
&                    /1,'MODEL SFC V WIND STR',125,001, &
&                    'V_FLX ON surface'/
```

   Where:

   - **IFILV** Identifies field as MASS/VELOCITY point
   - **AVBL** is the model output character string variable name (grib1)
   - **IQ** is the GRIB PDS OCTET 9 (table 2) - Indicator of parameter and units
   - **IS** is the GRIB PDS OCTET 10 (table 3&3a) - Indicator of type of level or layer
   - **AVBLGRB2** is the model output character string variable name (grib2)
   - **A UNIQUE** array location UPP uses to store this variable in parallel arrays (e.g. 901)

   User procedure:

   - A latent heat flux variable (LHTFL) was found in the GRIB1 parameter tables, so add a new unused parameter number (237) using Table 130 to define the new field.
     http://www.nco.ncep.noaa.gov/pmb/docs/on388/table2.html
   - Used level type surface, which is 001
     http://www.nco.ncep.noaa.gov/pmb/docs/on388/table3.html
   - Add as:

```
DATA IFILV(950),AVBL(950),IQ(950),IS(950),AVBLGRB2(950) &
&                    /1,'ACC SFC LAT HEAT FX ',237,001, &
&                    'ACC LHTFL ON surface '/ !Table 130
```

5. Read the model output field from the wrfout file by adding the new variable into ***INITPOST.F*** This file is used for reading the WRF-ARW model output files in netcdf format.

   User procedure:

   - Add using the 2D variable **SNDEPAC** (snowfall accumulation), which is also a 2D surface based accumulation field, as a template by following it through the routine.
   - Add to top section of the routine in 'use vrbls2d' to initiate the new variable as:

```
aclhf
```

   - Read in the new variable as:

```
VarName='ACLHF'
call getVariable(fileName,DateStr,DataHandle,VarName,DUMMY, &
IM,1,JM,1,IM,JS,JE,1)
do j = jsta_2l, jend_2u
```

```
      do i = 1, im
         ACLHF ( i, j ) = dummy ( i, j )
      end do
   end do
   !  print*,'ACLHF at ',ii,jj,' = ',ACLHF(ii,jj)
```

6. Determine the correct routine to add the new variable to (e.g. ***SURFCE.f***, ***MDLFLD.f***, ***MDL2P.f***, etc). You will need to determine the correct routine to add your field into; this is the place that you will fill the array with the data and call gribit to output the field. The correct routine will depend on what your field is. For example, if you have a new diagnostic called foo, and you want it interpolated to pressure levels, you would need to add it to ***MDL2P.f***. If foo was only a surface variable, you would add it to ***SURFCE.f***. If you wanted foo on native model levels, you would add it to ***MDLFLD.f***. If you're not sure which routine to add the new variable to, choose a similar variable as a template.

   Note: This is also where you would add any calculations needed for your new variable, should it be required.

   User procedure:

   - Treat ACLHF like a surface field (***SURFCE.f***)
   - Using the variable **SNDEPAC** (accumulated depth of snowfall) as a template which is also an accumulated field that is just being read through and output, similar to what we want.
   - Add in top section in 'use vrbls2d, only' to initiate the new variable as:

     ```
     aclhf
     ```

   - Add in main section using the template variable as a guide.
     - Note that ID(02), which is the ID for table version number, is added and set to 130. This is the table that we are adding the new variable to.
     - The block of code in `blue` is for metadata for the accumulation field being added in this example and the user does not need to edit it and it is not always needed. For example, for an instantaneous field, you would not need that block.

     ```
     !  ACCUM UPWARD LATENT HEAT FLUX AT SURFACE
     IF (IGET(950).GT.0) THEN
        ID(1:25) = 0
        ID(02) = 130
        ITPREC = NINT(TPREC)
     !mp
        IF(ITPREC .NE. 0) THEN
           IFINCR = MOD(IFHR,ITPREC)
           IF(IFMIN .GE. 1)IFINCR = MOD(IFHR*60+IFMIN,ITPREC*60)
        ELSE
           IFINCR = 0
        ENDIF
     !mp
        ID(18) = 0
        ID(19) = IFHR
        IF(IFMIN .GE. 1)ID(19)=IFHR*60+IFMIN
        ID(20) = 4
        IF (IFINCR.EQ.0) THEN
           ID(18) = IFHR-ITPREC
     ```

```
      ELSE
         ID(18) = IFHR-IFINCR
         IF(IFMIN .GE. 1)ID(18)=IFHR*60+IFMIN-IFINCR
      ENDIF
      IF (ID(18).LT.0) ID(18) = 0
      if(grib=='grib1') then
         DO J=JSTA,JEND
            DO I=1,IM
               GRID1(I,J) = ACLHF(I,J)
            ENDDO
         ENDDO
         CALL GRIBIT(IGET(950),LVLS(1,IGET(950)), GRID1,IM,JM)
      elseif(grib=='grib2') then
         cfld=cfld+1
         fld_info(cfld)%ifld=IAVBLFLD(IGET(950))
         fld_info(cfld)%ntrange=1
         fld_info(cfld)%tinvstat=IFHR-ID(18)
!$omp parallel do private(i,j,jj)
         do j=1,jend-jsta+1
         jj = jsta+j-1
            do i=1,im
               datapd(i,j,cfld) = ACLHF(i,jj)
            enddo
         enddo
      endif
   ENDIF
```

7. For grib2 output, add the new variable to /UPPV4.1/parm/***params_grib2_tbl_new***. For all current UPP output fields, this table lists, in order, the:

   - Discipline (https://www.nco.ncep.noaa.gov/pmb/docs/grib2/grib2_doc/grib2_table0-0.shtml)
   - Category (https://www.nco.ncep.noaa.gov/pmb/docs/grib2/grib2_doc/grib2_table4-1.shtml)
   - Parameter Number (https://www.nco.ncep.noaa.gov/pmb/docs/grib2/grib2_doc/grib2_table4-2.shtml)
   - Table information (0 for parameters from the WMO table; 1 for parameters from the local NCEP table)
   - Abbreviated Variable Name (from the parameters table)

   User procedure:

   - Since there is already a latent heat flux (LHTFL) parameter in this table, create a new Latent Heat Flux parameter so as to not overwrite the current one, just in case you want both to be output
   - Latent heat flux is a meteorological field (discipline=0)
   - Latent heat flux is a temperature product (category=0)
   - Pick an unused parameter number from the table defined by discipline=0 and category=0 (Table 4.2-0-0: https://www.nco.ncep.noaa.gov/pmb/docs/grib2/grib2_doc/grib2_table4-2-0-0.shtml). In this case, the unused parameter number 205 was chosen.

- Add using the NCEP local table (table=1)
- Choose an abbreviated parameter name to describe your field (e.g. ACLHF)
- Add as:
  ```
  0 0 205 1 ACLHF
  ```

8. Add the new variable to the /UPPV4.1/parm/***post_ avblflds.xml***, which lists all fields available for output in GRIB2 format.

   - Post_avblfldidx: the unique array number given in the RQSTFLD.f routine.
   - Shortname: name describing the variable and level type
   - Pname: the abbreviation for your variable
   - Table info: table used if not standard WMO
   - Fixed_sfc1_type: level type
   - Scale: precision of data written out to grib2 file

   User procedure:

   - Add as:
     ```
     <param>
        <post_avblfldidx>950</post_avblfldidx>
        <shortname>ACC_LATENT_HEAT_FLUX_ON_SURFACE</shortname>
        <pname>ACLHF</pname>
        <table_info>NCEP</table_info>
        <fixed_sfc1_type>surface</fixed_sfc1_type>
        <scale>4.0</scale>
     </param>
     ```

9. Add the new variable to the /UPPV4.1/parm/***postcntrl.xml file***, which lists all fields and levels you wish to output for GRIB2. Remake the /UPPV4.1/parm/***postxconfig-NT.txt*** file, which contains the information from the xml that UPP reads.

   - See the User's guide on steps for creating the text control file

   User procedure:

   - Add as:
     ```
     <param>
        <shortname>ACC_LATENT_HEAT_FLUX_ON_SURFACE</shortname>
        <pname>ACLHF</pname>
        <scale>4.0</scale>
     </param>
     ```

10. Add the new variable to the /UPPV4.1/parm/***wrf_ cntrl.parm*** file, which lists all fields and levels you wish to output for GRIB1.

    User procedure:

- Add as:

```
(ACC SFC LAT HEAT FX ) SCAL=( 4.0)
L=(10000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000
```

** Note the first entry is turned from 0 to 1. This turns on the first level.

11. Run clean on the code and recompile the code to include the changes before running your UPP run script.

User procedure:

```
>> ./clean -a
>> ./configure
>> ./compile >& compile.log &
```

12. Assuming the modified code compiled successfully and you were able to produce grib output, you can check the grib file for your new variable.

GRIB2 output of the new variable from this example procedure (using the wgrib2 utility if available on your system).

- The new variable will not be defined by the variable name. Instead it will be defined using the grib2 parameter information you entered into ***params_grib2_tbl_new*** from step 7 of this procedure.

```
456:43204412:vt=2009121718:surface:6 hour fcst:var discipline=0 center=7 local_table=1
parmcat=0 parm=205:
ndata=121002:undef=0:mean=1.97108e+06:min=-1.12e+06:max=2.406e+07
grid_template=30:winds(grid):
Lambert Conformal:  (402 x 301) input WE:SN output WE:SN res 8
Lat1 14.807213 Lon1 231.818604 LoV 258.040009
LatD 38.270000 Latin1 38.270000 Latin2 38.270000
LatSP 0.000000 LonSP 0.000000
North Pole (402 x 301) Dx 15000.000000 m Dy 15000.000000 m mode 8
```

GRIB1 output of the new variable from this example procedure (using the wgrib utility if available on your system).

- The new variable will not be defined by the variable name. Instead it is defined by `kpds5=237` (grib1 ID) and `kpds6=1` (level type) you gave it in ***RQSTFLD.f*** in step 4.
- For this particular variable, the accumulation period is shown, due to the addition of the metadata block in the ***SURFCE.f*** routine.

```
rec 319:59903982:date 2009121712 var237 kpds5=237 kpds6=1 kpds7=0 levels=(0,0) grid=255
sfc 0-24hr acc:
var237=undefined
timerange 0 P1 6 P2 0 TimeU 1 nx 402 ny 301 GDS grid 3 num_in_ave 0 missing 0 center
7 subcenter 0 process 125 Table 130 scan:  WE:SN winds(grid)
Lambert Conf:  Lat1 14.807000 Lon1 231.819000 Lov 258.040000
Latin1 38.270000 Latin2 38.270000 LatSP 0.000000 LonSP 0.000000
```

```
North Pole (402 x 301) Dx 15.000000 Dy 15.000000 scan 64 mode 136
min/max data -1.1217e+06 2.40583e+07 num bits 12 BDS_Ref -112.17 DecScale -4 BinScale
0
```

Acknowledgement
If significant help was provided via the UPP helpdesk for work resulting in a publication, please acknowledge the Developmental Testbed Center Mesoscale Modeling Team.
For referencing this document please use:
UPP Users' Guide V4.1, 23 pp. [available online at http://www.dtcenter.org/upp/users/docs/user_guide/V4.1/upp_users_g
]