Chapter 1: Introduction to the UEMS

Chapter Contents:

1.1 The UEMS: What is it?

The National Weather Service (NWS) Science and Training Resource Center's (STRC) Unified Environmental Modeling System (UEMS) is a complete, full-physics, state-of-the-science numerical weather prediction (NWP) package that incorporates the NOAA (NEMS)* and WRF systems into a single user-friendly, end-to-end forecasting system. All the capability of the NCEP NEMS and NCAR WRF models are retained within the UEMS; however, the installation, configuration, and execution of each is dramatically simplified to encourage its use within the operational, private, and university forecasting and research communities.

Nearly every element of an operational NWP system has been integrated into the UEMS, including the acquisition and processing of initialization data, model execution, output data processing, and file migration and archiving. Even tools for the display of output are provided. Real-time forecasting operations are enhanced through the use of an automated process that incorporates various fail-over options as well as the synchronous post-processing and distribution of forecast files. The UEMS can run on either a stand-alone workstation or a cluster of Linux computers with minimal effort.

Additional information about the UEMS, including much of the same information contained in this guide, is located on the NWS Science Operations Officer (SOO) Science and Training Resource Center (STRC) web site:

http://strc.comet.ucar.edu

** Oops - The "Unified" part of the UEMS has yet to be implemented but the sales & marketing in my head suggested that I make this announcement now in advance of the promotional campaign, should there be one. In any case, "Unified" will be included in the "UEMS" in the future, should there be one.*

1.2 Why you should care

The UEMS package was developed to promote the use of local numerical weather prediction models within the US National Weather Service (NWS) Weather Forecast Offices (WFOs) to achieve the following goals established by the SOO Science and Training Resource Coordinator (SOO STRC):

1. To improve the knowledge and use of NWP systems and issues at a local level.

2. To advance the forecasting process through an improved understanding of atmospheric processes and the use of non-traditional forecast tools.
3. To increase participation among the WFOs and other groups in developing and running NWP systems to examine local forecast problems.

Running the UEMS will serve to provide (*At least this is the plan*):

1. NWP guidance to NWS WFOs and River Forecast Centers (RFCs) at temporal and spatial scales not available from operational data sources.
2. A powerful tool for studying local forecast problems and historically significant weather events.
3. An alternative to the configuration and physics of operational systems.
4. A means to develop and test new diagnostic forecast techniques.
5. A method of training forecasters on NWP-related issues.
6. More hair on your head and less on your back*, and that is a good thing!*

1.3 Summary of UEMS features

a. The UEMS is a complete, full-physics, NWP package that incorporates dynamical cores from both the NCAR ARW and NCEP models into a single end-to-end forecasting system.

b. The system is easy to install and configure. Users can run simulations within 30 minutes of installation. And with such fantastic documentation, why shouldn't it be easy?

c. Yes, the rumors are correct! No compilers are necessary for running the UEMS. The system includes pre-compiled binaries optimized for 64-bit Linux systems running in distributed memory Linux environments. The MPICH executables are also included for running on local clusters across multiple workstations.

d. The installation tool does just about everything the user could want, including the creation of a user account (if necessary), installation of the package, and configuration of the system.

e. An auto-updating capability is integrated into the UEMS. When an update or patch becomes available, it is downloaded and installed automatically. And this time, it (almost) works!

f. The UEMS includes a preconfigured WRF ARW core benchmark case so that you can test the performance of your system.

g. The system is designed to give users flexibility in configuring and running NWP simulations, whether it is for local research or real-time forecasting applications.

h. The UEMS allows for the acquisition of many different initialization datasets. And just like the developer, the system is semi-intelligent in that it can determine which datasets are available for ingestion at a given time.

i. The system can reduce the likelihood of missed forecasts during real-time operations by incorporating multiple "fail-over" options that include alternate servers, datasets, or

initialization forecast hour. Should there be a problem with a simulation, it can also send e-mail to users.

j. All configuration parameters have been organized and documented in easy to read files that contain default settings for each dynamical core.

k. The system can be configured to automatically calculate an appropriate time step for a given horizontal grid spacing.

l. The UEMS supports the automated processing of forecast files concurrent with a model run, thus allowing users to view forecast fields while the model is still running.

m. The post-processor supports a wide variety of display software including AWIPS, BUFKIT, GrADS, GEMPAK, NAWIPS, and netCDF.

n. The EMSUPP, a modified version of the NCAR/NCEP Unified Post Processor (UPP), can processes forecast fields on 81 different pressure levels from 10 to 1025mb in GRIB 2.

o. Simulation output files can be exported to remote systems via SFTP, FTP, rsync, and SCP.

1.4 How much computer power is required to run the UEMS??

The answer to this question depends upon whether you will be executing the UEMS for research or real-time forecasting purposes. For real-time use, you need as much computer power as you can afford, with a premium placed on fast, multi-CPU Linux systems with at least 32GB of physical memory. The amount of memory should be commensurate with CPU performance. That's just the nature of NWP, as the chances are that you will always want to run the model at higher resolutions over a more extensive computational domain with the most accurate (and expensive) physics and dynamics. For research purposes, you will still want all the best for your runs; however, you will not need to make as many compromises since the speed of the machine is not as critical as you are not up against any deadlines to get a forecast completed. Finally, if you plan on running nested simulations, consider increasing the amount of physical memory even further.

The binaries compiled for the UEMS will run on any INTEL or AMD Linux (non-BSD) system running a minimum kernel version of 2.6 or later. Also, the processors must support SSE instructions so older AMD and INTEL processors may not be a viable option.

While the minimum amount of physical memory needed is ~8Gb, it is strongly recommended that machines have a minimum of 16Gb for real-time modeling to avoid paging and swapping issues. If you have less than 4Gb of memory, then consider increasing your system resources. Besides, memory is relatively inexpensive.

1.5 What support is available for the UEMS?

The STRC UEMS modeling package is entirely supported by the NWS Science and Training Resource Coordinator (SOO STRC). The US NWS and other NOAA agencies may request help from the developer directly. Support is also available to other government agencies upon request. Non-governmental agencies, such as non-profit groups, academic institutions, and the commercial sector

are also welcome to use the UEMS; however, support will be limited and up to the discretion of the SOO STRC. This means that you can ask for help, but it may take a while to get a response, so be persistent and patient -- it will eventually pay off.

Please keep in mind that a single person conducts *all* UEMS activities, including testing, package design, development, support, marketing and promotional appearances, documentation upkeep and poetry "slams", research, computer maintenance, real-time data server upkeep, website development (or lack thereof), DVD burning, labeling, stamp licking, and mailing. So be kind and understand that nothing gets done as quickly as it should, and most things not at all.

1.6 What If I have a brilliant idea for the UEMS?

The UEMS package is always under development, so please feel free to send the SOO STRC your suggestions for improvement; however, it is strongly suggested that you read the last paragraph in the previous section before doing so. It wouldn't hurt to read the preface section either.

Chapter 2: Installation of the UEMS

Chapter Contents:

2.1 Introduction: Because you have to start somewhere

Installation of the UEMS requires the use of uems_install.pl, which is designed to tackle nearly all the challenges you might encounter during the installation of this system. It is highly recommended that you use the most current version, as your life will become less complicated and more fulfilled if you do. *And that is a statement only a few modeling systems can make!*

"Where might I get this uems_install.pl utility thing?"

All NOAA-affiliated current and wannabe users of the UEMS can request the routine from the SOO Science and Training Resource Coordinator (SOO STRC). All non-NOAA users, i.e., everyone else, must register for the UEMS on the SOO/STRC site:

http://strc.comet.ucar.edu/software/uems

Registration allows users to receive notification of updates as they become available, and also allows the developer to advance his world geography skills when a request from Mauritius arrives.

After you register, there may be a delay in receiving response, since the information is checked for some semblance of legitimacy and to avoid advertisements for "dating services" from being sent to others. Once the UEMS concierge can confirm your registration, which typically takes as little as a few minutes, but as long a week if the lone support person is out of the office or comatose, you will receive a second email message with the routine attached. Enjoy!

2.2 System requirements

Below are a few of the system requirements for installing and running the UEMS. They are really more recommendations than requirements, but their purpose is to provide a good foundation on which to begin living the "UEMS lifestyle."

a. A relatively current Linux distribution

The UEMS is tested on Red Hat Enterprise, Fedora, CentOS, SuSe (sometimes), and Ubuntu (occasionally) distributions. Other Linux variants will probably work just fine; however, it is too difficult to keep up with all the releases and updates. Additionally, there is typically a lag before the developer can install a new distribution for testing, so just stick with what works, and we'll both be happier for it.

b. The UEMS user must be using a Tcsh or Bash shell

Anyone using the UEMS must be running a **tcsh** or **bash** shell; otherwise, horrible things can occur, such as your simulation failing to run.

c. 58 Gb of available disk space

This requirement pertains to the installation of the UEMS only. Of course, running an NWP model can use up a significant amount of disk space as you dump data files every minute for 144 hours, so this requirement should only be considered as a minimum.

d. A minimum of 16Gb of system memory

Again, this is just a bare minimum. The chances are this amount will need to be increased as you become more empowered by the UEMS. Eventually, all this power will go to your head, and you will need a personal supercomputer. Remember, with more memory comes more power! Your child's education fund can just wait.

2.3 Installing the UEMS

The default behavior of the utility is to download and unpack the package files from UEMS World Headquarters and then do the configuration for use on your system. There is no need to include any special flags or arguments. Everything is built into uems_install.pl, including a sense of dedication and purpose. All you need to do is follow the guidelines provided below. Most importantly, *think positive thoughts and don't show any fear!* The install utility can taste your fear.

The UEMS can be installed as root or a regular user. Installation as root user requires you specify a non-root user to assign ownership but also allows you to install the system anywhere you want.

2.3.1 The basics, or the stuff you must know

You can do a fresh install whether or not you have an existing release on your system. The novice user must embrace the most basic of all installation flags:

```
% uems_install.pl --install [release version]
```

As indicated by the [square brackets], the version number is optional, since the default is the most current release. If you are looking to install a previous release, say release "18.12.15", then you must specify the release number as an argument to "—install." For example:

```
% uems_install.pl --install 18.12.15
```

The above example is for demonstration purposes only, since release 18.12.15 probably doesn't exist. So don't try it at home. There is additional information on determining what releases are available in Section 2.4.

The downloading of package files from the UEMS servers may require a considerable amount of time, depending upon the speed and reliability of your network connection. Be patient, as your effort will be rewarded. If there is an interruption during the process, don't despair, as you can continue from where you left off by passing the "--continue" flag the next time:

```
% uems_install.pl --install --continue
```

2.3.2 During the installation

Regardless of the release you choose, the process will include the following:

- A greeting, because you are important to me
- Prompt you for the installation directory (Default: /usr1)
- Check to make sure the directory exists and whether you have write permission
- Determine whether an existing installation resides at that location, and if so:
  - Get the version of an existing installation
  - Ask whether you want to rename the existing installation to <uems>.<release> or have it whacked.
- If installing as root user:
  - A prompt for the name of the user to assign ownership of the package
  - An attempt to create a new account and home directory if the user does not exist
  - A check that the user's login shell is either tcsh or bash
  - A prompt for a password if a new user was created

Finally,

- Install from the specified source
- Complete the post-install configuration
- Congratulate you on another wise decision

During the installation, useful information will be printed to the screen, so don't leave the room, even if you have to go.

2.3.3 Local network installation

If you don't have direct access to the UEMS servers, you can still install the system. This method requires that the package files be manually downloaded and placed in a local directory. Remember the disk space requirements when deciding where to put the files, and then follow these steps:

Step a. Create a temporary directory for the package files. This directory can be called anything, provided that you have at least 60GB of space. For this example, the directory will be named "**/usr1/repository**".

```
# mkdir /usr1/repository
```

Step b. Go to a UEMS server in your browser to view the available full releases:

http://ems1.comet.ucar.edu/scums/releases
http://ems2.comet.ucar.edu/scums/releases
http://ems3.comet.ucar.edu/scums/releases

Index of /<scums>/releases

Name	Last modified	Size	Description
18.9.1/	30-Feb-2018 16:10	-	
18.53.4/	31-Dec-2018 24:61	-	

Determine the release you want to install, which should be the most current if you know what's good for you. The releases are identified by such silly names as "18.9.1" or "18.53.4," because the UEMS developer was born without an imagination.

Step c. Create a directory under "/usr1/repository/" with the name of the release to be downloaded. For example:

```
# mkdir /usr1/repository/18.53.4
```

Step d. Open the desired release on the server:

http://ems1.comet.ucar.edu/scums /releases/18.53.4

Download the package files to "/usr1/repository/18.53.4" directory. There may be quite a few files, some of which are large, so while you are waiting you can take up waterfall kayaking, investing in bitcoin, or some other worthwhile activity. Trust me; you have the time.

Step e. Once the tarfiles are downloaded, and your injuries have healed, run the uems_install.pl routine as follows (Replacing the location in the example below with the actual location on your system):

```
# ./uems_install.pl --install --repodir /usr1/repository
```

Step f. Congratulate yourself on another risky yet rewarding skill mastered – the UEMS installation, not the waterfall kayaking.

2.3.4 Checking for success

Before using the system, you must ensure the environment variables are correctly set. These variables are defined in

uems/etc/EMS.cshrc	(T|Cshrc shell users),
uems/etc/EMS.profile	(Bash shell users),
uems/etc/modulefiles/UEMSwrkstn	(Module users)

which is either sourced (T|Cshrc shell), executed (Bash shell), or loaded (Module users) upon login. To set the environment, you need to edit your startup (login) file, located in the top level of your home directory (~) and add the following lines:

For T|Cshrc shell users, edit ~/.cshrc

```
# Set UEMS environment variables
#
if (-f  <path>/uems/etc/EMS.cshrc) then
        source <path>/uems/etc/EMS.cshrc
endif
```

For Bash shell users, edit ~/.bashrc or ~/.bash_profile

```
# Set UEMS environment variables
#
if [-f <path>/uems/etc/EMS.profile]; then
        .   <path>/uems/etc/EMS.profile
fi
```

For module users, edit ~/.cshrc or ~/.bashrc or ~/.bash_profile

```
# Set UEMS environment variables
#
module load  UEMSwrkstn
```

Following a carefree installation and the inspection of your login files, you can now *log out and return* as the UEMS user. Make sure your environment is correct by attempting the following commands:

```
%   cd $UEMS
```

Wherein you will be located at the top level of the UEMS

You should also try:

```
% cd $EMS_RUNS
```

You are now located at the top of the run-time directory.

If the above tests are successful, try running the "**sysinfo**" command provided with your system (and it is yours now):

```
% sysinfo
```

You will see a summary of computer system configuration that includes the Linux distribution along with other information such as your blood pressure (just kidding). Please make sure that the following values are correct:

System Information for moots.comet.ucar.edu

System Date	: Tue May 8 15:45:31 2018-	*If your date is the same, you have problems!*
System Hostname	: moots.comet.ucar.edu	*Make sure the hostname is correct*
System Address	: 104.200.21.121	*Make sure the IP is populated and correct*

Sockets	: #	The number of physical CPUs or mice that you would see if you opened up the computer case
Cores per Socket	: #	The number of cores on each physical socket CPU, the stuff you can't see
Total Cores	: #	This value is sockets **x** cores per socket

If the number of Sockets or the Cores per Socket is incorrect, you will have to change the values in the EMS.cshrc, EMS.profile, or UEMSwrkstn file. If everything appears OK, then your installation is complete, and you are ready to become a modeler. If not, just give your not-so-local UEMS person a call, or send a message of encouragement along with some baked goods. He needs them both.

The final step in verifying the installation is running a benchmark simulation. Check out Appendix B for the gory details.

2.4 Useful flags for every occasion

Many command-line flags and options can be passed to the uems_install.pl utility. This section attempts to provide some guidance for the available options; however, just like most everything else with the UEMS, the documentation is probably lacking some important details. Regardless, it contains enough information to meet most of your installation wants and desires.

And that's really all you want out of life, isn't it?

Flag: **--install [release version|list|listall|listgeog]**

When can you use this flag: **Installing or just thinking about it.**

What this option can and can't do for you:

Passing "**--install**", without any other arguments or additional flags, indicates that you want to do a fresh installation of the UEMS. The default is to install the most current release; however, this behavior may be overridden by including a release number as an argument to "**--install**", in which case the specified release will be used.

Then, when using "**--install**,"

```
% uems_install.pl --install [additional options]
```

Or

```
% uems_install.pl --install <release> [additional options]
```

Alternatively, you can pass "**list**", "**listall**", or "**listgeog**" as an argument. Including "**list**" will provide a listing of all the available releases on the UEMS servers. Passing "**listall**" will include a summary of all the individual package files for each available release. The "**listgeog**" argument will provide a listing of the WRF static terrestrial datasets included in the installation.

```
% uems_install.pl --install list
```

Or

```
% uems_install.pl --install listall
```

Or

```
% uems_install.pl --install listgeog
```

If you request a fresh install but have an existing installation on the system, you have the option of renaming the previous installation ("uems.<release number>") or deleting it from your machine. If you elect to save (rename) the previous installation, any computational domains that reside in the previous "uems.<release number>/runs" directory will be transferred to the new "<uems>/runs" directory. This default behavior is nullified by passing the "**--noruns**" flag (see below). If you have domains located elsewhere that you want imported, see the "**--import <dir>**" flag.

Flag: **-- addpack <nawips|source|workshop|xgeogs>**

When can you use this flag: **After you have completed an installation**

What this option can and can't do for you:

Passing "**--addpack**" allows you to install non-default UEMS packages. These packages are excluded because they are not integral to running the UEMS and can be very large. Removing these packages from the default installation reduces delays and issues related to slow or dropped network connections, which also reduces the number of support requests to the developer.

The argument to "**--addpack**" is a character string that is used to match against the packages available for post-installation. These packages include:

String ID	Package Description
nawips	The N-AWIPS (GEMPAK) graphics package with pre-built binaries
source	The source code used to build the UEMS (some exclusions)
workshop	Tutorial and training package (very large)
xgeog	Auxiliary terrestrial datasets (gtopo,nlcd2006,nlcd2011)

You can pass the "**--addpack**" flag multiple times or you can include list separated by a comma (,).

For example, if you currently have UEMS V18.9.8 installed, either:

```
% uems_install.pl --addpack  nawips  --addpack source
```

Or

```
% uems_install.pl --addpack --package  nawips,source
```

The **source** package includes all the files used to build the UEMS binaries provided as part of a release. It does not include the base WRF source code, which is available from the WRF Model Users Site, but it does include the UEMS modified files and configuration files used in building the UEMS version of the code.

The **workshop** package includes classroom exercises and training produced as part of a 2007 workshop. While still somewhat useful, the tarfiles are relatively large and contain materials in need of updating, which will eventually get done. If you don't plan on using this package, then there is no reason for it to be installed.

The **nawips** data display package used to be part of the default installation; however, due to its size and suspected limited user interest, it was pulled from the default installation party.

The **xgeog** package consists of three additional WRF terrestrial datasets that are not part of the default installation:

String ID	Package Description
gtopo	Global USGS terrain elevation dataset (Pre WRF V2.8.1)
nlcd2006	US CONUS 40-category 2006 NLCD/MODIS landuse classification
nlcd2011	US CONUS 40-category 2011 NLCD/MODIS landuse classification

These datasets are either non-essential for running a simulation or were replaced with a more current version. These non-essential datasets may be requested individually or collectively ("**xgeog**") from the UEMS servers.

Flag: **-- emshome <path to location of UEMS installation>**

When can you use this flag: **Installing & Updating**

What this option can and can't do for you: **Hopefully, not much**

Include the "**--emshome**" flag if you want to specify either the location of an existing UEMS installation for updates or the future home of a fresh installation when passing the "**--install**" flag. For the most part, passing this flag is unnecessary; as the uems_install.pl routine will figure out most everything it needs and pester you for any additional information. So unless you have a good reason for using it, don't.

Flag: **--import <path to domain directories>[/<domain>]**

When can you use this flag: **Anytime**

What this option can and can't do for you: **import-export stuff**

The "**--import=<path to domain directories>"** flag allows you to import existing domains from a previous UEMS installation. This option is intended for importing either specific domains from a current $UEMS/runs directory, or a group of domains within a non-EMS directory (somewhere else on your system).

For example:

```
% uems_install.pl --install --import=/usr1/mysavedfiles/uems/runs
```

The above command will import any computational domains from "/usr1/mysavedfiles/uems/runs/" into a new installation. If you have been living right, all will go as planned and your configuration settings preserved. *Beginning to sweat a bit now aren't you?*

If you wish to import only a single domain from a directory containing multiple domains, then add the domain name to the end of the string:

```
% uems_install.pl --install --import=/save/uems/runs/bigstorm
```

In which case the "**bigstorm**" computational domain is imported to "<uems>/runs/" and processed following new installation.

You may pass multiple instances of "**--import**" if you have multiple domains to import:

```
% uems_install.pl --install --import=/saved/runs/bigstorm  --import=saved/runs/go-storm-go
```

Wherein both the "**bigstorm**" and "**go-storm-go**" domains are imported into "<uems>/runs/" and processed following installation.

If all you wish to do is import computational domains from your current UEMS installation ("<uems>/runs/"), then it is not necessary to include the "**--import**" flag as they are included by default. If you do not want the current domains under "<uems>/runs/" to be imported, then pass the "**--noruns**"

flag. Including "**--noruns**" has no impact on the "**--import**" flag. The "**--import**" flag is used to import directories moved to a location for safe keeping, which is why you need to include the <path to the domain directories> as an argument; otherwise, the almighty uems_install.pl does not have a clue as to what you want, and this sentence becomes even longer.

Finally, if you are a bit tentative about all this "importing" business, you can always wait until after the installation dust has settled and migrate (copy) any existing domain directories to the new location. Then, from each directory run:

```
% ems_domain --localize --update
```

Flag: **--continue**

When can you use this flag: **Installing**

What this option can and can't do for you: **Make your dreams come true**

Pass the "**--continue**" flag if you wish to continue the download and installation process from a previous failed attempt. The most likely situation for the use of this flag is following a failed connection to the UEMS servers in which only a partial download completed.

When the "**--continue**" flag is passed then uems_install.pl will look for the incomplete "uems" installation, which contains a "releases" directory under which all the downloaded tarfiles are located. The downloading and unpacking will continue from the last successfully downloaded package tarfile. None of the previously downloaded files will re-installed unless you include the "**--force**" flag.

If you fail to include the "**--continue**" flag, all is not lost. During the process, you will be prompted as how to proceed with the installation and handling of the existing partial installation. At that time you can choose to continue with the previously aborted installation. So the "**--continue**" flag is not so much a necessity as much as it is a convenience, just like a second thumb on your right hand.

Flag: **--nogeog**

When can you use this flag: **Installing**

What this option can and can't do for you:

When passing the "**--nogeog**" flag, you are telling the install routine not to include the large terrestrial datasets. Remember that you need these data to run a simulation, but if you already have them locally, you may not want to download them again. You can just copy or move the files over to the new UEMS installation.

However, there are times when these data change, and you need the updated files. *Remember, with the UEMS, stuff just happens. Just don't let it happen to you.*

If you are doing a fresh install but want to keep the terrestrial dataset files from a previous installation, do not include the "--**scour**" flag as that will result in the deletion of the files before you have a chance to

move them to a new installation.

Flag: **--xgeog [matching terrestrial dataset string]**

When can you use this flag: **Anytime**

What this option can and can't do for you:

Pass the "**--xgeog**" flag if you want to install any of the additional WRF terrestrial datasets that are not part of the default installation. There are a few datasets available that are either non-essential for running a simulation or replaced by a more current version. These non-essential datasets are still available from the UEMS servers, but you must specifically request them, which may be done either during a full install or afterward.

To see all the terrestrial datasets available, including those deemed "non-essential":

```
% uems_install.pl –install listgeog
```

The argument to "**--xgeog**" is a character string that will be used to match against the **wrf.geog_*** tarfiles on the UEMS servers. If the string matches multiple tarfiles, then all matching files will be downloaded and installed.

You can pass the "**--xgeog**" flag multiple times or you can include a list of tarfiles separated by a comma (,). Yes, you can even install terrestrial dataset packages previously installed as part of the default installation.

For example:

```
% uems_install.pl --install --xgeog gtopo_30s --xgeog nlcd2006
```

Or

```
% uems_install.pl --install --xgeog gtopo_30s,nlcd2006
```

Flag: **--nolocal**

When can you use this flag: **Installing & updating**

What this option can and can't do for you:

Passing the "**--nolocal**" flag overrides the default behavior of looking for existing UEMS package files in the "<uems>/updates" or "<uems>/release" directories. Even if package tarfiles exist locally, just look the other way, and download new files from a remote location. The old ones smelled funny and were attracting fruit flies anyway.

You would only use this flag if you believed the locally stored tarfiles are corrupted, and you didn't feel like deleting them first.

Flag: **--emshost <ems1|ems2|ems3>**

When can you use this flag: **Installing & Updating**

What this option can and can't do for you:

Including the "**--emshost**" flag instructs the installation utility to contact a specific UEMS host for information on update and installation packages. The default behavior is to select one of the official UEMS servers randomly, but you can request a specific UEMS server or a different one should there be any, which there aren't.

Flag: **--noruns**

When can you use this flag: **Installing**

What this option can and can't do for you:

When including the "**--noruns**" flag along with "**--install**", you are telling the routine **not** to import the computational domains from the existing "<uems>/runs/" directory to the new installation, which is the default behavior. Passing "**--noruns**" **does not affect** the "**--import**" option, so go ahead and use them together, or not at all, with complete reckless abandon. If you pass "**--noruns**" and there is no UEMS currently on the system, let me know what happens.

Flag: **--relsdir <path/some directory>**

When can you use this flag: **Installing**

What this option can and can't do for you:

Passing the "**--relsdir** <path/some directory>" flag will override the default location where the UEMS release package tarfiles are downloaded. When installing the system, the tarfiles are placed in the "<uems>/release/<release number>" directory unless you pass the "**--relsdir <something>**" option. It is best not to use this flag unless you have a good reason to change the default directory.

Flag: **--dvd <path to dvd rom drive>**

When can you use this flag: **Installing**

What this option can and can't do for you:

Use the "**--dvd**" option only if you are installing from a DVD (DVDs are still available for the "Hey you kids, get off my lawn" types). This flag is used only if you needed to copy the uems_install.pl routine from the DVD to another location because you could not mount the drive with executable permission; otherwise, you can ignore this flag. See Section 2.3.3 for guidance on DVD installation.

Flag: **--nounpack**

When can you use this flag: **Installing & Updating**

What this option can and can't do for you:

Pass the "**--nounpack**" flag if you do not want to install the downloaded UEMS package tarfiles (the default is to install them), whether for a new installation or an update. Just let them sit in the "<uems>/release|updates/<release number>" directory until they've cooled off, because they're "too hot to handle."

FYI - This flag should be named "--noinstall", but I thought that might be confused with the "**--install**" flag; however, you can always try "--noinstall" and see how far it gets you.

Flag: **--curl** and **--wget**

When can you use this flag: **Installing & Updating**

What this option can and can't do for you:

Passing the "**--curl**" or "**--wget**" flag instructs the installation routine to use that utility for HTTP communication to the server(s) guarding the package tarfiles. The default is to use whichever one is available on your system or randomly select between the two, but you have the option to say otherwise if one method be failing you.

Flag: **--scour**

When can you use this flag: **Installing**

What this option can and can't do for you:

Passing the "**--scour**" flag results in the removal of a previous UEMS installation from your system. The default behavior, i.e., not passing "**--scour**", is for the utility to ask you what to do if it encounters another installation, and provides a menu of choices. So if you like to order "off the menu" then pass the "**--scour**" flag.

Flag: **--repodir <path to release directories>**

When can you use this flag: **Installing & Updating**

What this option can and can't do for you:

Use the "**--repodir**" flag when you want to grab release or update packages from a local source rather than downloading the tarfiles from the UEMS servers. The argument to **"--repodir**" is a path to the directory containing one or more UEMS releases or update packages:

```
% ems_domain --update|install [X.Y.Z]  --repodir=<path to release directories >
```

The path, <path to release directories>, leads to a directory where the individual release subdirectories are located (<release>). These subdirectories contain the tarfiles for the UEMS release identified by the directory name (**X.Y.Z**). For example, on your local system, there exists a directory containing UEMS releases named "/usr/local/uems/releases/." Located beneath "/usr/local/uems/releases/" are subdirectories, each containing the package tarfiles for that release or update. The name of each release subdirectory represents the release number, such as:

```
% ls /usr/local/uems/[releases|updates]
    15.54.0  18.9.8  19.47.1
```

Then, when using "**--repodir**",

```
% uems_install.pl --install|update --repodir /usr/local/uems/releases|updates
```

The install routine will read the contents of any directories below <path to release directories> and select the most current update ("**--update**") or release ("**--install**").

In practice, it is best to maintain a separate release and update repository since updating requires the sequential installation of all update tarfiles from the currently installed release to the most recent available.

Flag: **--debug**

When can you use this flag: **Installing & Updating**

What this option can and can't do for you: **Deep Probing**

Include the "**--debug**" flag if you want uems_install.pl to print out all sorts of information about what is going on inside its brain while it's attempting to figure out what's going on inside of yours.

2.5 Updating the UEMS with passion

Eventually, you will want to update your old, crusty, buggy, and yet well-loved UEMS with a newer and shinier model that contains new bugs and that new UEMS smell. When that moment arrives, you can call upon your installation tool (yeah, it is a tool now) to tackle the job. It will someday be known as the "Swiss Army knife" of installation tools, but for now, it is just the leather awl of installation tools.

Just as with the installation processes, the great UEMS unwashed should embrace the most basic of updating commands:

```
%  uems_install.pl --update  [release version|list|listall]  [additional options]
```

Note that the release version is optional, as indicated by the [square brackets], since the default is the most

current release. It is recommended that *you not include the update release* and allow the uems_install.pl to do its magic. If you have been a slacker and found yourself a few dozen updates behind, no problem, as the uems_install.pl will figure out what updates you need to bring your system up to date. With any (a lot of) luck, the utility will download and install each missing update until you are current and just as brilliant as the day you installed your first UEMS. At least that's the plan.

Alternative arguments to "**--update**" are "**list**" and "**listall**." Passing "**list**" will provide a listing of all the current releases available on the UEMS servers for which you are eligible. Including "**listall**" will include a summary of the individual package files for each release listed by "**list**."

If you have any domains located in your "<uems>/runs" directory, then they will also be updated with new configuration files and your existing configuration settings will be retained (no, really, this time I promise!). At the risk of a complete failure, if you do not wish to update your domains, then include "**--norefesh**" flag. The update process does not include re-localizing the domains under "<ems>/runs/" by default, but this step is accomplished by including the "**--localize**" flag.

As you may or may not recall, when you installed the system, an entry is placed in a crontab file that automatically downloads and installs updates. If you fancy this auto-update feature, then enable the crontab entry by removing the comment, "#", from the line. If automation makes you nervous, Luddite, then keep it disabled and fly manually while following the guidance provided.

Flag: **--allyes**

When can you use this flag: **Updating**

What this option can and can't do for you:

Passing the "**--allyes**" flag tells the uems_install.pl routine, "Yes, yes, I love you and trust you completely." While the developer has never heard those words himself, here is your chance to say them without the need to verbalize. The only time you will need the "**--allyes**" flag is when doing automated updates since, without it, the update will hang while waiting for you to respond to some silly rhetorical question.

Flag: **--norefresh**

When can you use this flag: **Updating**

What this option can and can't do for you:

This option is only valid during a UEMS update ("**--update**") and ignored when doing a clean installation ("**--install**").

Include the "**--norefresh**" flag if you do not want to update the existing configuration files within each domain directory under "<uems>/runs/." This means that all existing domains will be left untouched during an update and will contain the same configuration files and localization as before. Of course, they may no longer work, but you'll take comfort in knowing that you have some control over your life.

If you have a change of heart or come to find that your life ruined by such shortsightedness, you can always update and localize afterward with the "**ems_domain**" utility:

```
% ems_domain --update --localize
```

But we are getting a few chapters ahead of ourselves.

Flag: **--force**

When can you use this flag: **Updating**

What this option can and can't do for you: **Quite possibly nothing**

Pass the "**--force**" to reinstall an update. You might want to use this flag if something went horribly wrong during an update and you need to it again; however, including the flag while running "**uems_install.pl --update --force**" multiple times in succession suggests that you might have a problem. At least that's what they told the developer.

The utility of this flag has diminished with recent updates to uems_install.pl and has possibly been rendered useless; however, I keep it in this documentation as a sentimental reminder of simpler Workstation Eta/Workstation WRF/WRF EMS/NEWREMS times.

Flag: **--updtdir** **<path/some directory>**

When can you use this flag: **Updating**

What this option can and can't do for you:

Passing the "**--updtdir** <path/some directory>" flag will override the default location where the downloaded UEMS update package files will reside. When updating the system, the package tarfiles are placed in the "<uems>/update/<release number>" directory unless you pass the "**--updtdir**" flag. The default is the best option unless you have a good reason to change.

And there is no reason to change. You are awesome just the way you are!

Chapter 3: A self-guided tour through the UEMS

Chapter Contents:

3.1 The UEMS at first glance

At this point, you should have successfully installed the UEMS on your computer, because if you haven't, then this chapter will be of little use to you. If the UEMS is not installed, go back to Chapter 2 and return after you have completed that task.

At first glance, it appears that the UEMS consists of numerous directories, subdirectories, and files that all require constant attention by the user. The reality is that while there is a myriad of directories and files, users, for the most part, will not have to modify the contents. The UEMS is designed so that both the novice and well-seasoned NWP aficionado could feel comfortable with the system. Nearly everything comes pre-configured for most applications so you can get down to business. However, there are always exceptions, and thus, this chapter provides an overview of the system if you feel it necessary to dig deeper into its bowels.

3.2 A tour from the top

Environment variables can be used to navigate the UEMS, many of which will be explained in this chapter. The most important variable is $UEMS, which defines the top level of the system installation. As you recall from Chapter 2, one test of a successful installation is the ability to change to the $UEMS directory, i.e., "cd $UEMS". Go ahead and try it now, as this is where your self-guided tour will begin.

From the top level of the UEMS you will see the following directories and files:

bin The **bin/** directory contains the main 64-bit WRF binaries installed with the UEMS. These binaries are specially modified and compiled for use with the system. If you are unable to appreciate the gift that the developer has given to you and desire to compile the WRF binaries locally, then that can be accommodated. Please check out the contents of the uems/util/UEMSbuild directory and Appendix H; otherwise, you will fail.

conf The **conf/** directory contains default configuration files for the run-time routines and GRIB info files. Whenever you create a new domain, these files are copied into the local <domain>/conf/<routine> directory. If you continue to investigate **conf/,** you will find the following subdirectories:

ems_auto	Contains the default configuration files for the ems_autorun.pl routine
ems_post	Contains the default configuration files for the ems_post.pl routine
ems_prep	Contains the lone configuration file for the ems_prep.pl routine
ems_run	Contains the default configuration files for the ems_run.pl routine
gribinfo	Contains the configuration files for all the supported initialization datasets. There's a lot of good reading in those files.

data The **data/** directory is the location for the large terrestrial datasets, default model configuration files, default namelist files, and tables and text files used with the system. There are quite a few files and directories beneath **data/**; however, you probably won't need to concern yourself with the contents unless you are trying to make changes to the system defaults.

Here is a brief description of each subdirectory and its contents:

domains	Contains a sample WRF ARW domain - used with ems_domain.pl
geog	Contains the various terrestrial datasets used with the WRF
tables	Contains the default UEMS configuration files and tables

docs The **docs/** directory contains assorted pieces of documentation on the WRF and UEMS. Not all of the information is relevant to the system, but users may find some value in these resources.

domwiz The **domwiz/** directory contains the Java libraries and binaries for running the WRF Domain Wizard (DWIZ).

logs The **logs/** directory contains the installation and Domain Wizard (dwiz) log files. It is also the default location for the ems_autorun.log file. The log files from the individual domain simulations are not located in "uems/logs," but can be found in "uems/runs/<domain>/log".

runs The **runs/** directory is where all the magic happens. This is where newly created domains will reside after a successful localization with the Domain Wizard or the "**ems_domain**" utility. It is also where all simulations are run.

strc The **strc/** directory and subdirectories contain all the Perl routines and modules used to power the UEMS. It is unlikely that you will ever need to edit the files in this directory. These files are also most likely to be updated with each new UEMS release.

util The **util/** directory contains many of the utility programs and packages provided with the UEMS that help to put the "**S**" in "UEM**S**." Hint, the "**S**" is for "**S**ystem". Preconfigured domains used for running the benchmark simulations reside here along with NAWIPS and GrADS visualization packages. There are also additional files and directories that may be of interest to you, so it is worth spending some time investigating all that "**util/**" has to offer.

Here is a brief description of the util/ subdirectory contents:

bin	Various precompiled non-UEMS developed utilities (Section 3.6.1)
benchmark	The ARW core benchmark case. Information and guidance for running the benchmark is provided in Appendix B.
grads	GrADS binaries, scripts, and data files
hdfview	A data viewer for netCDF 4 (HDF 5) formatted files
mpich2	MPICH2 binaries and documentation
nawips	NAWIPS model data manipulation and display package (optional)
ncview	NcView binaries, scripts, and data files
workshop	Data, HTML files, and documentation for UEMS lab exercises (optional)
build	Source code and build scripts for compiling the UEMS from scratch. (optional)
extras	Additional support utilities

3.3 The EMS.cshrc, EMS.profile, and UEMSwrkstn files

The environment variables must be properly set when using the UEMS. These variables are defined in the uems/etc/EMS.cshrc and EMS.profile files, for Tcsh or Bash users respectively, and are normally set upon login (See "Checking for success" in Chapter 2.3.4). An alternative to these shell scripts is the etc/modulefiles/UEMSwrkstn file, which can also be used to set the environment with the command:

```
% module load UEMSwrkstn
```

Provided the file is in your $MODULEPATH:

```
setenv MODULEPATH ${MODULEPATH}:/<path>/uems/etc/modulefiles
```

User's may edit the file as necessary, but for the most part, the UEMS install tool does an adequate job during the configuration process. Regardless, make sure you are happy with the settings.

Here is a summary of the environment variables you can modify should you feel empowered:

UEMS_LOCAL

Setting the UEMS_LOCAL environment variable instructs the system to reassign the default location of the EMS_RUN directory (uems/runs/) to another location. You might set UEMS_LOCAL = 1 if the

system is being controlled a "UEMS administrator" with multiple users creating domains and running simulations, such as within a classroom setting.

If UEMS_LOCAL is set, then the new **default** location for the EMS_RUN and EMS_LOGS directories become $HOME/uems/runs and $HOME/uems/logs, respectively. If these locations are not acceptable, then modify EMS_RUN and EMS_LOGS in the environment settings. It will be obvious.

If these directories do not exist, an attempt will be made to create them, but it's probably safer to create them manually.

SOCKETS and CORES

The SOCKETS environment variable defines the number of physical processors, or *sockets*, that reside on your system. SOCKETS *does not* define the total number of processors (SOCKETS * CORES), nor does it define the processors used when running a simulation. The SOCKETS variable is simply the number of actual CPUs that you could touch if you were to open up the computer case and move the large heat sinks. Most stand-alone workstations have either 1 or 2 sockets, although some exotic and expensive mainboards can support 8 or more, but you probably don't have that much computational bling.

The CORES environment variable defines the number of cores contained on each socket CPU. A description of the processor type in the /proc/cpuinfo file usually provides a clue as to the number of cores on each socket such as "dual core", "Duo", "quad core", or "6-Core". If you read a "tri-core" or "albacore," then you have other problems. If you really don't know how many cores there are on each socket CPU then try running the UEMS "sysinfo" utility; otherwise, look up your CPU on the Internet thing.

The SOCKETS and CORES environment variables are used by the UEMS to define the *total number of processors available on your local system* (SOCKETS * CORES), the value of which is stored by the OMP_NUM_THREADS environment variable. It is important to have these values correct as many of the binaries are compiled to run on multiple processors and the system needs to know the number of total cores available. Note that the UEMS can easily be configured for distributed computing across multiple Linux systems, but only for the "emsupp," WRF "real.exe" and the ARW core executables. Other binaries are compiled for distributed computing on the local host while still others run in shared memory. You cannot, for example, distribute the processing of initialization data across a cluster of workstations. This minor limitation was introduced to simplify UEMS development since the advantage in exporting computational load to other systems was minimal.

Also note that setting either SOCKETS or CORES greater than the true values for your system will result in degradation of UEMS performance. You can do it, and the Linux kernel will comply by creating the requested number of threads, but your performance will suffer, and you don't want that to happen.

Finally, don't get all giddy with excitement if the UEMS installation tool indicates that you have twice as many cores on your computer than you think it does. If hyper-threading is turned ON in the BIOS then the system can incorrectly report the number processors and cores 2-fold. In that event, you will have to manually assign the correct values to SOCKETS and CORES in EMS.cshrc (or EMS.profile), as your system performance will be compromised. You are better off turning hyper-threading OFF in your BIOS but that is up to you. Please read Section 3.4 for an explanation - multiple times.

MAILX

The MAILX environment variable defines the mail routine to use when sending informative messages or pearls of nonsensical wisdom, which tends to happen from time to time. It's just like a fortune cookie that provides lottery numbers. Note that the email recipients are defined in the ems_autorun.conf file located beneath the individual run-time domain directories. Additional information about the file is provided in Chapter 10, if you make it that far.

MPICH_HOME

MPICH_HOME defines location of the MPICH executables on your system. By default, the UEMS will use the pre-compiled executables located under uems/util/mpich2, which in most cases is fine. However, if you would like to use a locally compiled version of MPICH, or the package provided with your Linux distribution, you may do so by specifying the location of MPICH below. If MPICH_HOME is commented out or blank the UEMS provided package will be used.

DWIZ users - If using an alternate MPICH installation, you will need to create a link from

$MPICH_HOME/bin/mpiexec.gf to $MPICH_HOME/bin/mpiexec.gforker:

% cd $MPICH_HOME/bin

% ln -s mpiexec.gforker mpiexec.gf

EMS_NAWIPS

By default, the NAWIPS setting is commented out. You may remove the "#" if you wish to use the NAWIPS display package interactively; that is, from the command line, for generating images or displaying your model output. You *do not have to set the NAWIPS environment variable for generating BUFKIT files* or images using the included script_drvr.csh routine as it will be done automatically.

One consequence of including the NAWIPS environment is that it can increase the number of user-defined environment variables to a value greater than 343. Why is the number "343" important? Well, because there is an internal check in MPICH2 for the number of environment variables. If a user has more than 350 (343+ 7 that MPICH2 includes) the MPICH2 routines will fail to start. I don't know why. I didn't write MPICH2, but it is there.

HTTP_PROXY

The HTTP_PROXY environment variable is used when your local network has a proxy server that accepts HTTP requests on some port other then 1080. Symptoms of this problem include an inability to connect to outside data sources via HTTP when running ems_prep or ems_autorun. For most users this is not a problem, in which case HTTP_PROXY is left blank. If you know this is the case for your system then set the HTTP_PROXY to:

HTTP_PROXY = <IP address of server>:<port used>

For example, if proxy server 192.168.150.10 used port 80:

HTTP_PROXY = 192.168.150.10:80

If the proxy server requires a user name and password

HTTP_PROXY = <user>:<passwd>@<IP address>:<port used>

3.4 Environment variables you hate to love

There are additional environment settings defined in EMS.cshrc and EMS.profile that are used by the UEMS routines and utilities; only a few of which that may be of value to you. Those variables of importance are identified as follows:

$UEMS	The top level of the UEMS installation (uems)
$EMS_BIN	The primary directory where the UEMS binaries reside (uems/bin)
$EMS_DATA	The directory where the static datasets reside (uems/data)
$EMS_STRC	The location of the UEMS STRC Perl modules and routines (uems/strc)
$EMS_RUN	The directory where all your computational domains are located (uems/runs)
$EMS_UTIL	The utility (closet) directory (uems/util)
$EMS_CONF	The location of the default configuration and GRIB information files (uems/conf)
$EMS_LOGS	Location of the general UEMS log files (uems/logs)
$EMS_DOCS	Location of the UEMS and other WRF documentation (uems/docs)

3.5 The UEMS and hyper-threading – Just say no!

I'll probably have to repeat this diversion somewhere else in the guide, but this is as good a place as any to cover this topic. In regards to the use of hyper-threading with the UEMS, *I do not recommend using the additional threads available when hyper-threading is turned on in the BIOS.* The reason for this statement is that you are much more likely to see degradation in performance rather than improvement. Yes, there are a limited number of conditions under which you might see some improvement depending upon your processors, the amount of cache, your domain configuration and the decomposition; however, you are far more likely to see degradation in the performance.

Keep in mind that when you use hyper-threading, you are only increasing the number of threads available on a processor. *You are not increasing your computational resources*, such as the total number of processors available. If the number of threads created exceeds the number of available processors, then multiple threads will be assigned to a single processor. Consequently, each thread will have to share the resources allocated to that processor, such as memory and cache. This is only beneficial if the total amount of resources needed by the multiple threads is less than that available on the processor and there is no sharing of resources between threads. It is the sharing of resources, specifically cache, is what increases the overhead, and degrades performance.

As an example, if you have a dual CPU, quad core system that has a total of eight processors available. If

you specify that all eight processors be included in a simulation, the WRF model will decompose the domain into eight patches, each of which is assigned by the kernel to a processor as a single thread. Since there is a single thread assigned to each processor, there is no sharing of available core resources. The model/system handles all the communication between the patches.

If hyper-threading is turned on and you specify 16 processors be used, the model will decompose your domain into 16 patches and the kernel will create 16 threads. The problem is that since you only have eight processors there will be two threads assigned to each core. Remember that the threads will likely have to share resources on the processor, specifically cache, and it's the overhead involved in the sharing between threads that leads to the degradation in performance.

3.6 The run-time routines

The run-time routines would be the heart and soul of the EMS if it had a soul. These tools are used to acquire and process initialization data, run simulations, and post-process the model output files. They are also used to automate the process when running a real-time forecast system. Be kind to the run-time routines and they will be kind to you.

All the routines and ancillary modules are located below the uems/strc/Ubin/ directory; however, *they must never be run from this location*. Instead, when you create a computational domain (Chapter 5), symbolic links will be made from your domain directory to the files in uems/strc. When you look in the top level of your domain directory, you will see that the "**.pl**" has been removed in the link name so that "<command>" points to "<command>.pl" in uems/strc/Ubin/. Taking this approach makes it easier for the UEMS to keep things tidy. And a tidy system is a happy system.

Each of the run-time routines will be discussed ad nauseam in subsequent chapters of this guide; however, now is a great time to introduce them and provide you with an opportunity to say "Hello."

The primary (big) four:

ems_prep Used to identify, acquire and process the external datasets for use as initial and boundary condition information. It also sets the timing of your simulation.

ems_run Checks the model configuration, ingests the output files from ems_prep, creates the initial and lateral boundary conditions, and then executes the simulation.

ems_post Processes all the simulation output and exports the data files to exotic locations of your dreams.

ems_autorun Automates the process of running ems_prep, ems_run, and ems_post in succession for the purpose of creating a real-time simulation experience.

Two additional utilities that will see occasional use but are not part of the "big four." They do not have links from the domain directories, but you can probably handle that fact of life as well as you handled others.

ems_clean

The **ems_clean** routine is used to return your run-time directory to a specified state. This process

includes the removal of unnecessary files and directories and recreating symbolic links. The routine is automatically run whenever you execute one of the "big four" above, but may also be run manually from within a domain directory with the cleaning power controlled by the **"--level #**" flag, i.e.,

USAGE: % ems_clean --level 3

The level of cleaning ranges from simple removal of log files and resetting links (level 0) to relocalizing your computational domain and resetting the configuration files to the default (level 6). Most users will employ level 1, 3, or 4. Running "ems_clean --help" provides a more detailed explanation of the power that is ems_clean.

ems_autopost

The **ems_autopost** routine is run from within ems_run to start the processing of model output concurrent with a simulation. You will never start ems_autopost directly from the command line and you may not be aware it exists, but it does, just like the Keebler Elves and the next release of the UEMS. This routine is most frequently used when running ems_autorun for real-time forecasting. More details on ems_autopost can be found in Chapter 11.

3.7 Additional utilities you would have missed had they not been listed here

The UEMS includes a few additional utilities to enhance your local modeling experience, and then there are some that are simply useless. Below is a summary of those utilities and where they reside.

3.7.1 A few additional UEMS utilities

The following utilities are located in the **uems/strc/Ubin** directory, and should be in your $PATH following login in so there is no need to point to them directly.

benchtest

The purpose of this routine is to estimate the amount of time required to run a simulation over a given domain. To accomplish this task, a 12 hour simulation will be run over a user defined number of iterations, with the average, maximum, and minimum run times printed to the screen. Due to overhead and other on-going system processes, there usually is some run-to-run variability in the amount of time required to complete a simulation, so this routine will provide some guidance on what to expect.

USAGE: % benchtest <benchmark domain> <number iterations>

cnet

The primary purpose of **cnet** is to determine whether a system is reachable from the machine on which it is being run. The program takes either a host name or IP address and then skillfully checks whether the system is resolvable and currently reachable. If the answer to these questions is "Yes"

then cnet spits out the hostname, IP, and network interface used to reach the remote system.

Returned values of 0 (host, ip, iface, [ssh]) indicate that there was a problem, which may be better illuminated by passing the '-v' flag.

USAGE: % [--verbose] [--nossh] hostname1 ... hostnameN

Option	Description
--nossh	Do not conduct passwordless SSH check
--**v|V|verbose**	Provide some verbosermation (I made a new word!)
--**help**	Print out this menu again in case you missed it the first time

ems_domain

The purpose of the ems_domain routine is to create and/or localize a computational domain for use with the UEMS. It may be used as an alternative to the domain wizard if the DW GUI is not available; however, using ems_domain requires that you understand the basics about model domain configuration and grid navigation since there are no pretty depictions of your domain.

After executing ems_domain, your domain will be ready to run simulations, with or without you.

USAGE: % ems_domain [--info [domain] or --create <domain> or --localize [domain] or --import <directory>] [Other Options]
Or
USAGE: % ems_domain -h for the entire list of options

gribnav

The **gribnav** utility is one of many tools that may be used to check the navigation of a GRIB file. This utility will provide the projection type, grid dimensions and spacing, true and standard Lat/Lon, the pole and corner points of the domain defined in a GRIB file. It should work with most all of the UEMS grid navigation options, although feel free to point out any “short comings” to receive bonus points and valuable prizes.

USAGE: % gribnav <GRIB 1 or GRIB 2 file>

mpitest

The mpitest routine tests routine the viability of the UEMS when running on a single system or multiple nodes within a cluster. There are no mandatory flags with mpitest, although the "--nodes" flag comes highly recommened unless you want to run the test on localhost with a single core. When running the test with miltiple nodes make sure that passwordless SSH is configured between all nodes, including the localhost and itself. That's just the way MPI works.

Additionally, MPI must be installed in the same location on each node, which is defined by the

"$EMS_MPI" environment variable. You provide a list of hostnames/IPs that you plan on using to execute a simulation and mpicheck will do a few basic tests to determine whether your master plan for world domination has any chance of succeeding:

USAGE: % mpitest [Other flags] --nodes node1[:#cpus],node2[:#cpus],...,nodeN[:#cpus]

Where “node#” are replaced with the hostnames of the machines in the cluster. If you are running the UEMS on a single workstation, then don’t worry about it unless you are just looking for some fun and want to be disappointed.

ncpus

The primary purpose of **ncpus** is to determine the number of available processors for the machine on which it is being run The routine simply mines local system information, then does some magic and spits out a number. Whether it is the correct number depends completely upon the skill of the author to distill the available information into something meaningful, such as "42"

A value of 0 (cores) indicates there was a problem, which may be better illuminated by passing the '--v' flag.

USAGE: % ncpus [--v | --V | --verbose] [--sockets]

Option	Description
--**sockets**	List the number of sockets, cores per socket, total cores, and total threads
--**v|V|verbose**	Print out additional information for debugging
--**help**	Print out this menu again in case you missed it the first time

sysinfo

Running the sysinfo utility provides a summary of your computer system, including the hostname, IP address, Linux distribution and kernel, the number and type of processors on the machine, as well as some additional information. Here is an example of the information provided:

USAGE: % rozumal@seven-> sysinfo

```
Starting UEMS routine sysinfo (V15.44.12) on seven at Thu Dec 4 20:30:33 2015 UTC

     *  Gathering information for localhost seven
        ------------------------------------------------------------------

          System Information for seven

              System Date           : Thu Dec  4 20:30:34 2015 UTC
              System Hostname       : seven
              System Address        : 128.117.107.12

              System OS             : Linux
              OS Kernel             : 2.6.32-431.1.2.0.1.el6.x86_64
              Kernel Type           : x86_64
              Linux Distribution    : CentOS release 6.7 (Final)

          Network Interface Information for seven
```

```
    Network Interface     : eth0
    Interface Address     : 128.117.107.12
    Address Resolves to   : Nothing
    Interface State       : Up

    Network Interface     : eth1
    Interface Address     : None Assigned
    Address Resolves to   : Nothing
    Interface State       : Up

    Network Interface     : lo
    Interface Address     : 127.0.0.1
    Address Resolves to   : Nothing
    Interface State       : Up

    Network Interface     : virbr0-nic
    Interface Address     : None Assigned
    Address Resolves to   : Nothing
    Interface State       : Inactive

Processor and Memory Information for seven

    CPU Name              : Intel(R) Xeon(R) CPU X5670 @ 2.93GHz
    CPU Instructions      : nehalem
    CPU Type              : 64-bit
    CPU Speed             : 1600 MHz

    UEMS Determined Processor Count
        Physical CPUs     : 2
        Cores per CPU     : 6
        Total Processors  : 12

    EMS.cshrc Specified Processor Count
        Physical CPUs     : 2
        Cores per CPU     : 6
        Total Processors  : 12

    Hyper-Threading       : Off

    System Memory         : 23.5 Gbytes

UEMS User Information for rozumal on seven

    User  ID              : 8010
    Group ID              : 1750
    Home Directory        : /home/comet/rozumal
    Home Directory Mount  : NFS
    User Shell            : /bin/tcsh
    Shell Installed       : Yes
    Shell Login Files     : .cshrc
    EMS.cshrc Sourced     : .cshrc
    EMS.cshrc Port Range  : None Defined

UEMS Installation Information for seven

    UEMS Release           : 19.4.2, WRF4.0.3
    UEMS Home Directory    : /usr1/uems
    UEMS Home Mount        : Local
    UEMS User ID           : 8010
    UEMS Group ID          : 1750
    UEMS Binaries          : x64

    UEMS Run Directory   : /usr1/uems/runs
    UEMS Run Dir Mount     : Local
    UEMS Run Dir User ID   : 8010
    UEMS Run Dir Group ID  : 1750

    Run Dir Avail Space   : 809.31 Gb
    Run Dir Space Used    : 53%

    UEMS Util Directory    : /usr1/uems/util
```

```
Your awesome UEMS sysinfo party is complete - Thu Dec 4 20:30:34 2015 UTC

The UEMS Metaphysician says: "Think Globally, Model Locally!"
```

3.7.2 Non UEMS-developed tools and utilities

Here is a summary of the "value added" utilities provided with the UEMS, some of which are used by the run-time routines. It's in your interest to not rename or remove them, but a few may be of use to you:

Located in the **uems/util/bin/** directory:

cpuid	Provides information of about your CPU
int2nc	Converts WRF intermediate format files into netCDF files
g1print	Prints out information in GRIB 1 files for creating a Vtable
g2print	Prints out information in GRIB 2 files for creating a Vtable
ncview	A netCDF file visualization tool
rdwrfin	Prints out a summary of WRF intermediate format files
rdwrfnc	Prints out a summary of WRF netCDF format files
cnvgrib	Converts between GRIB 1 and GRIB 2 formats (GRIB 1 <-> GRIB 2)
wgrib2	A very, very special GRIB 2 file interrogation tool (with extra UEMS love added)

Chapter 4: Just Getting Started? Read this Chapter!

Chapter Contents:

4.1 Just the basics, please

This chapter will be short and sweet, because there is nothing like a jumble of nonsensical, incomprehensible verbiage to persuade potential users to turn off their computers in favor of needlepoint. Besides, you will get enough blather reading the remaining chapters, if you make it that far. This is the time for parental guidance and handholding, which we all need from time to time whether we want to admit it or not.

If you want to learn the essential steps in running the UEMS, then read this chapter. If you are a well-seasoned user, you can skip this information and move on something else, such as macramé.

4.2 Help! Help! - How do I use this thing?

Now that you have successfully installed the system and hopefully run the benchmark case (**Appendix B**), it is time to set up and complete a simulation with your own domain. The UEMS comes preconfigured for success, so you do not have to change any settings initially, although there are plenty of options with which to experiment as you become more comfortable with the system. For now, we shall stick to the basics.

Running the UEMS is foolproof, provided that you follow these five simple steps:

Step 1. Create a computational domain with the Domain Wizard GUI

Start the Domain Wizard (DW) by executing "**dwiz**" from the command line, following which the GUI will magically appear. All the guidance you need for running the Domain Wizard is found in **Chapter 5,** although you can probably muddle through the creation of a domain even if you are an "*Instructions? I don't need no stinking instructions*" type of person. It is reasonably intuitive and minimal issues will be encountered provided you remember to:

a. Don't bite off more grid points than you can chew. The creation the perfect domain requires that you balance the areal coverage and grid spacing of your domain (and thus the number of grid points) with the processors and memory on your computer system. Failure to understand your limitations will only lead to colossal failure. Trust me, I know colossal failure.

b. Be sure to localize your domain by pushing the "Localize" button at the top of the localization window. A lot of users miss this critical step.

Review Chapter 5 if you have any questions. After Step 1 is successfully completed, you can move on to Step 2. That's the simple logic and beauty behind the UEMS "5 Step System."

Step 2. Prepare to run a simulation

While running the DW, a new domain was created and placed in "uems/runs/<domain name>". Beneath your newly minted domain you will find some links and sub-directories with which you may be unfamiliar. An explanation of these files and directories is provided in Chapter 6, but you can't be bothered to read ahead right now as you are on a mission towards success.

The UEMS default configuration is sufficient for running simulations with a grid spacing between 6 and 25km. If everything went as planned during the installation, you do not need to edit any files at this point. Nonetheless, if you feel a bit queasy by placing so much trust in the system, then there are two configuration files that you can review to make sure everything is correct, because stuff sometimes happens.

The "**conf/ems_run/run_ncpus.conf**" file contains the settings for the number of processors to be used during a simulation and the decomposition of the domain. Make sure that these values are correctly specified for your computer. This number should be the total number of processors available, which is the number of physical CPUs on your system multiplied by the number of cores per CPU. Look for:

REAL_NODECPUS = local:#
WRFM_NODECPUS = local:#

Where the "**#**" is be the number of processors on your system. If the current value is incorrect, you can change it to an appropriate value. *You are strongly encouraged to heed the advice provided regarding the number of processors to use.* As a general rule, you should use ***fewer*** than the total number of processors available on your system. You want to leave some resources free for system processes and IO.

You also are encouraged to read about and possibly modify the **DECOMP** and **NUMTILES** parameters. In doing so, you will become familiar with these parameters and all the magic powers they possess.

The second file you might want to examine is "**conf/ems_run/run_physics_cumulus.conf**," which contains the physics settings for your simulation. Initially, the only parameter that you might consider changing is **CU_PHYSICS** if your grid spacing is *less than 8 km*. If this is the case, then you can change:

CU_PHYSICS = 11, 0

To

CU_PHYSICS = 0, 0

But you don't have to make this change. It's just a recommendation.

While inspecting the physics configuration, you are encouraged to familiarize yourself with the other parameters and options available in this file, since you will revisit them again and again as

you evolve into a modeler. This fate is inevitable so don't even try to resist.
Once step 2 is completed, you can move on to Step 3.

Step 3. Processing your initialization data

This step assumes that you have a network connection to the outside world to access the initialization datasets. Alternatively, you can use data files (GRIB) that already reside on your system (**Chapter 7**); however, this example assumes you have an outside connection.

If you have good bandwidth, you can run the following command:

```
% ems_prep --dset gfs --length 24
```

Or, if you have limited bandwidth, then run:

```
% ems_prep --dset gfspt --length 24
```

Running either command instructs ems_prep (Chapter 7) to download and process the first 24 hours of forecast files from the most recent Global Forecast System (GFS) operational run. These files will serve as the initial and boundary condition data for a 24-hour simulation ("--length 24"). Any sub- (nested) domains are included by passing the "--domains" (Chapter 7) option, but we are not there yet. We're taking baby steps here.

You might want to take note of the information being printed to the screen and become familiar with the steps in this process, such as the sites from which the data are obtained. *And yes, you really are awesome!*

Once this step is completed, you can move on to Step 4, but not before.

Step 4. Running your simulation

This next step is easy, simply run:

```
% ems_run
```

Doing so instructs the system to create the initial and lateral boundary condition files and then start the simulation. Some semi-valuable information is printed to the screen, so don't become distracted by anything going on outside your window like a passing fire truck or a tornado. You are a modeler now and all the interesting weather occurs only on your computer. Besides, real modelers don't have windows. So if you have windows, then please cover them with posters featuring Star Trek, Albert Einstein, puppies, or something emblazoned with the word "Decisions." *This should be Step 4b – "Cover windows."*

Running the simulation may take some time depending on the size of your domain, grid spacing used, physics options selected and the performance of your computer system. For your modeling education, *decreasing the grid spacing by ½, say from 16 to 8km, over the same computational area, will result in an 8-fold increase in time required to run the simulation*. Those are just the

sobering facts. Now get back to covering your windows.

Once your simulation has completed, you can move on to Step 5.

Step 5. Processing the simulation output files

Congratulations on a successful simulation! If your run was not successful, then you should not be reading these words, so please return to Step 4 and figure out what went wrong.

Following the simulation, all your output files are located in the "wrfprd/" directory. These data are in netCDF format and may be viewed directly with any utility that reads netCDF. Lucky for you the UEMS provides you with a couple such tools. For example, if you want a summary of the file contents:

```
% rdwrfnc –m <netCDF file>
```

Or to view the various fields in the file:

```
% ncview <netCDF file>
```

Now, if leaving the output files in netCDF format is not in your final plans, or even if you have no plans, you can use "**ems_post**" to process the data into another format, generate images, or export the files to another system. There are many possibilities available, and it's really up to you to choose. Much more information on ems_post is available in Chapter 9; however, for now you can take comfort in converting your netCDF files into GRIB 2 format by running:

```
% ems_post
```

Once your processing has completed, you can finally show your friends and/or significant other what you *really* have been doing alone in a room with the windows blocked out.

4.3 Additional information you might want to know

Once you become comfortable with running each step above, you can simplify the entire process by using the **ems_autorun** routine, which is discussed in Chapter 10. This routine is primarily designed for real-time forecasting applications but can be used for case studies as well. Additionally, if you decide to use ems_autorun for real-time forecasting, you might also want to consider using the autopost option detailed in Chapter 11.

Good luck, and may the power of your awesomeness always be with you!

Chapter 5: Just Because You Can - Mastering Your Model Domain

Chapter Contents:

5.1 Mastering the basics

Before you head off into the wonderful world of numerical weather prediction, you will need to identify and create a computational domain. This process constitutes your first two baby steps to becoming a modeler; specifically, 1) Defining the areal coverage and navigation of your domain(s), and 2) Localizing the domain to create the necessary terrestrial datasets. The creation and localization of a computational domain can be achieved either through use of the Domain Wizard (DW) GUI or pseudo-manually by running the "ems_domain" from the command line. If you are relatively new to this process, then you are probably best off starting with the DW, the guidance for which is provided in Section 5.2.

There are a few important items to keep in mind when creating a computational domain:

a. The region over which you are interested in running a simulation will determine the map projection used. If you are using the WRF ARW core, it is recommended that you use:

   i. Lambert Conformal projection for mid latitudes
   ii. Polar Stereographic projection for high latitudes
   iii. Mercator or Lat-Lon projections for low latitudes

b. It is slightly easier to create a global domain with the ems_domain.pl utility than by using the DW, but you can still use the DW.

c. You must complete a successful localization before proceed with your modeling fun.

After identifying an areal coverage and grid spacing for your computational domain(s), you will need to complete a localization. The localization of your domain is handled by the WPS "geogrid" routine. The purpose of geogrid is to take the specified areal coverage of the primary domain and any sub-domains and then interpolate the available terrestrial data fields to your grids. In addition to computing the latitude, longitude and map scale factors at every grid point, geogrid will interpolate soil categories, land use category, terrain height, annual mean deep soil temperature, monthly vegetation fraction, monthly albedo, maximum snow albedo, and slope category for use in your simulation.

The global data for these fields are provided with the UEMS package. Several of the datasets are available in only one resolution, but others are made available in resolutions of 30", 2', 5', and 10' (here, " denotes arc seconds and ' denotes arc minutes). The default for both the DW and ems_domain is to select the resolution that is best suited for your domain.

New or additional datasets can be interpolated to a domain through the use of the table file, "GEOGRID.TBL". The GEOGRID.TBL file defines each of the fields that are produced by geogrid. It describes the interpolation methods to be used for a field as well as the location on the file system where the dataset for that field is located. The various GEOGRID.TBL files are located in the uems/data/tables/wps directory. The output from geogrid is written in WRF I/O API format. Geogrid can also be made to write its output in NetCDF for easy visualization using external software packages, including ncview. (*Some of the above text was liberated and adapted from the official ARW users guide, so the authors should get some props*)

5.2 Be the "Wizard of Your Domain" with the Domain Wizard

WRF Domain Wizard is the graphical user interface (GUI) for the WRF Preprocessing System (WPS). It enables users to visually define the areal coverage, grid spacing, and map projection for a computational domain and any nested, or sub-domains. It also allows users to localize a domain and view terrestrial data, such as the land-sea mask, terrain height, land use type, and vegetation index. Sounds kinda cool, doesn't it? Well, it's just lucky for you that the UEMS includes a precompiled version of the DW along with all the Java libraries you need to "run with the Wizard."

5.2.1 Starting the Domain Wizard

If the installation of the system went smoothly, then you can start the DW GUI from the command line:

% dwiz (Domain Wizard GUI)

And the Domain Wizard GUI will appear looking something like this:

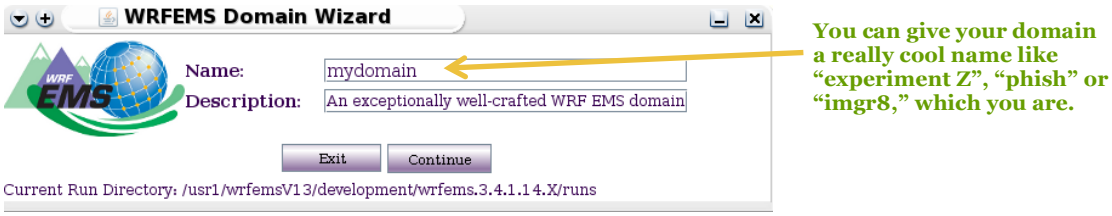
The startup window gives you an opportunity to select from creating a new domain or modifying an existing domain. The "Current Run Directory" is defined by the $EMS_RUN environment variable, so if the default location is not to your liking you will have to "Exit" Domain Wizard, setenv EMS_RUN <your directory>, and start Domain Wizard again. You do not need to change the default location though.

5.2.2 Opening an existing domain

If you want to open an existing domain, select the appropriately named "Open Existing Domain" radio button and then press "Continue." These carefully choreographed actions will take you to the "Open Domain" window. When completed, highlight your domain in the left-hand panel, look at the pretty preview in the right-hand panel, and click "Next" to load the domain. See, it's relatively easy!

5.2.3 The "Name Your Domain Challenge" window

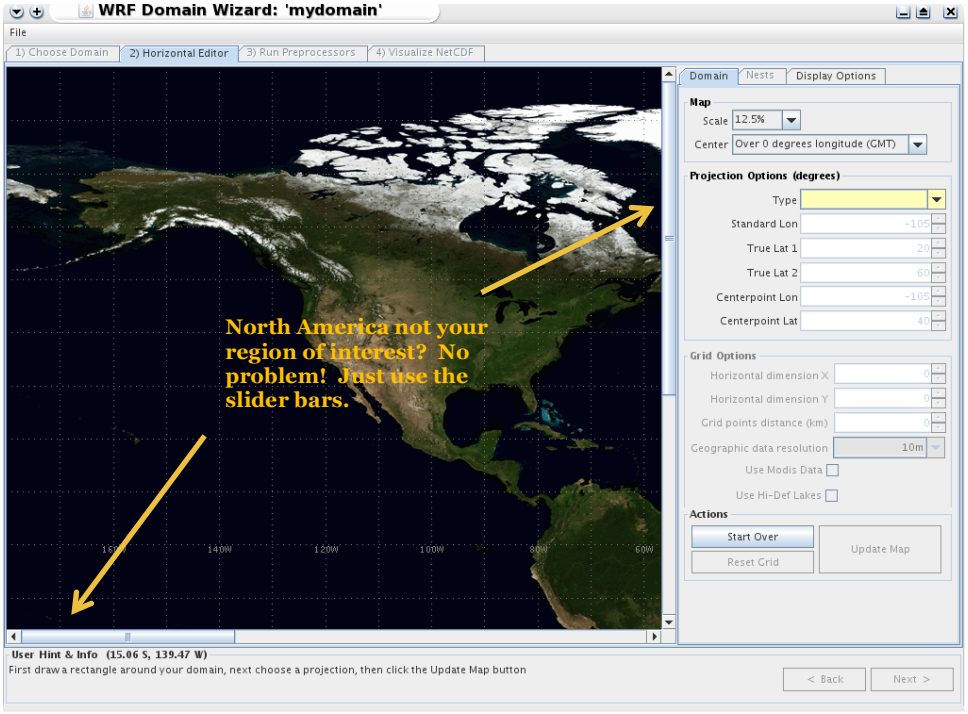
If you choose to create a new domain by selecting the button entitled "Create New Domain," the "Continue" button will move you on to the "Name Your Domain Challenge" window! This window allows you to give a name to your domain. The name must not include spaces, and consist of only lower-case characters (if you happen to forget these rules, the Domain Wizard is happy to force them upon you). Something snappy and clever is highly recommended, similar to that shown below. But don't be a copycat either.



Once you have given your domain a name, such as “splat” or “tofu” or “ilove2model,” click the “Continue” button to move on to the “Horizontal Editor” window. Go ahead and click it – Now.

5.2.4 The “Horizontal Editor” window

The Horizontal Editor window allows you to define a model domain by drawing a bounding box on a map (left side), and then to edit the map projection variables (right side). Always present is the standard menu bar at the top of the window. You can exit the application or query the version information by selecting options found under the “Actions” and “Help” buttons of the menu bar. At the bottom of the GUI window, a User Hints & Information text area sometimes suggests steps to be taken, and at other times summarizes the success status of the step that has recently completed. The UEMS minions are working on a 3D holographic version for the UEMS 27.9 release, but until then, here is an image depicting what the window will look like on your system:



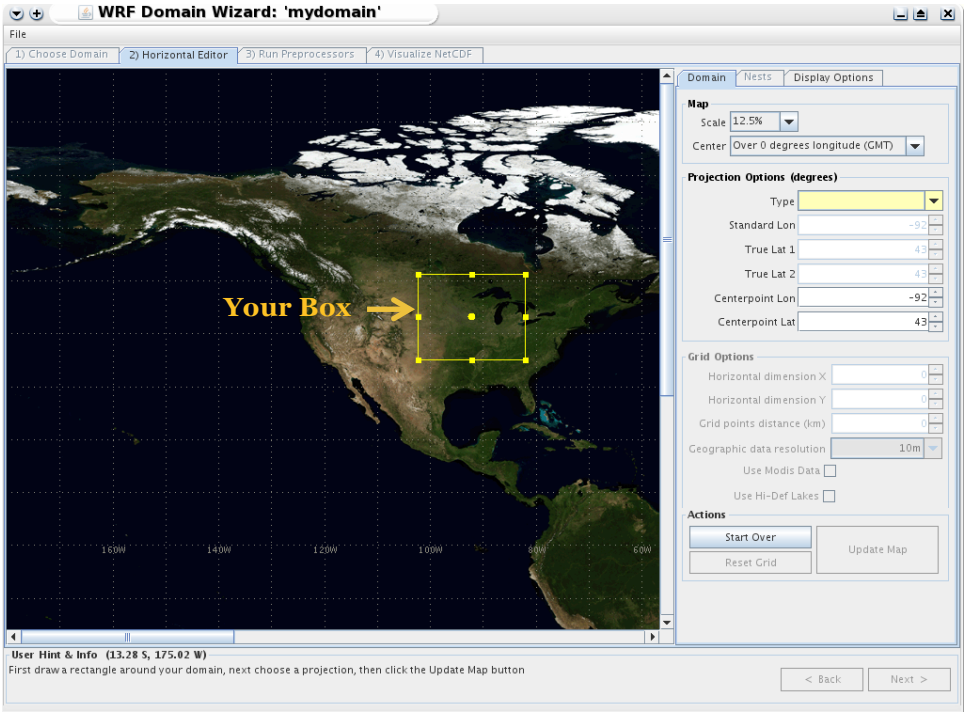
The initial map image above shows a Cylindrical Equidistant projection of the world centered over North America. The region can be increased or decreased in size by changing the “Scale” in the upper right of the window. The entire global map is not displayed all at once, but the panel has sliding scroll bars on the bottom and right sides to reposition the image within the panel. So, if you are located say, in Mauritius, and North America is not the center of your modeling universe, you can use the scroll bars to make things right.

5.2.5 Defining your primary domain

Your "Primary Domain," which is sometimes referred to as the "Mother of all domains" (MOAD), is the foundation of your simulations. It is the bedrock upon which you will cultivate the fruits of your modeling labor. Because if anyone can grow something on bedrock, you can! With this in mind, you can now move ahead to crafting your primary domain.

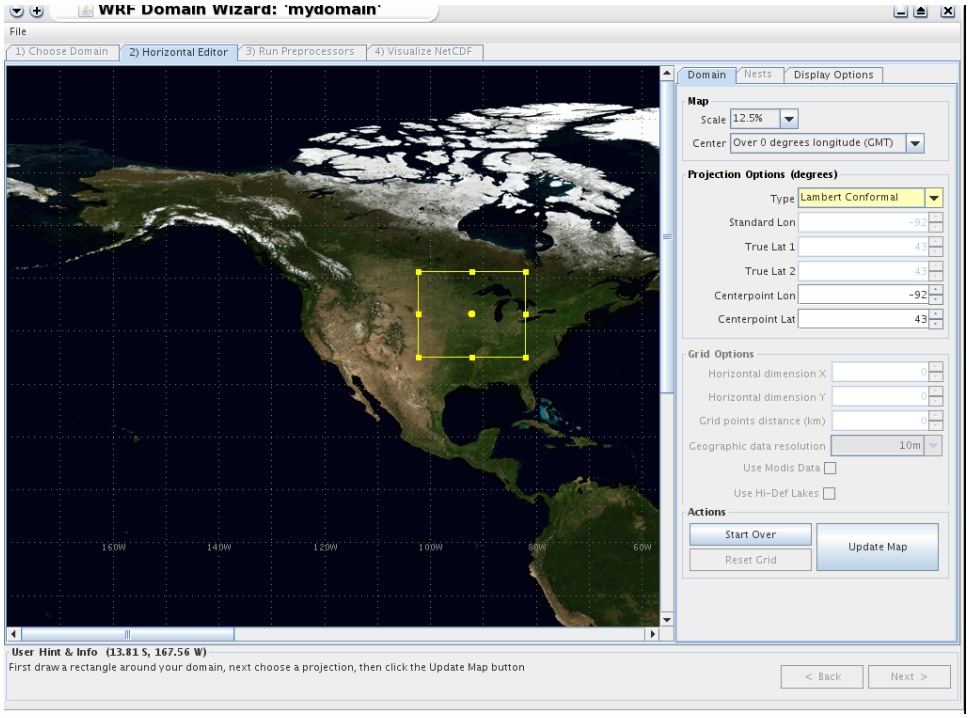
5.2.5.a *The "Draw a Box" challenge*

Initially, only the global map is active in the Horizontal Editor window, with all other interface buttons disabled (grayed out). As you press mouse button 1 (left click) on the global map and drag the mouse to a new location, a yellow domain-bounding box is drawn that specifies a limited area domain. Qualitative status information is displayed in the User Hints & Information panel, specifically the lower left (LL) and upper right (UR) corner latitudes and longitudes in the selected domain. Once you have completed the box drawing challenge your window will look something like the following:



5.2.5.b *Defining your ARW domain*

For this example, we'll choose the Lambert Conformal projection to demonstrate an ARW limited area domain. Once the projection type is chosen, you can modify the center point location of your primary domain (domain 1). These values can be changed either by highlighting and editing the settings in the box or by using the increment/decrement buttons. As the changes are made to the primary domain, the bounding box that defines the domain also changes.

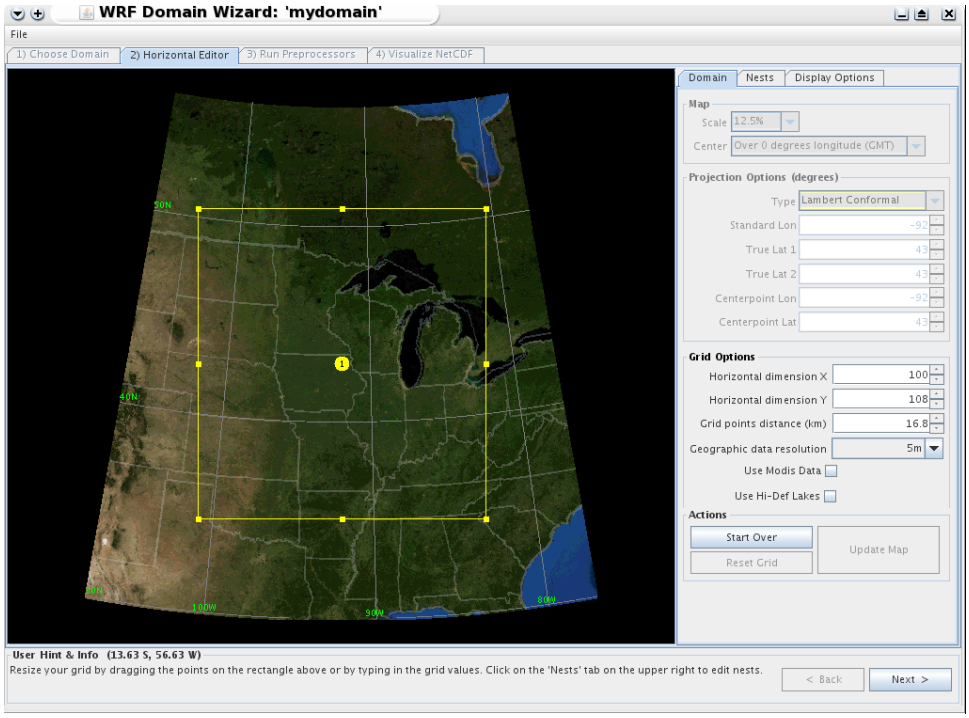


Lambert Conformal Grid projection type for mid-latitudes

Modify center Lat-Lon

Select "Update Map," just because you can.

After the projection is selected and any changes to the center point of the primary domain are made, click "Update Map." The Domain Wizard, (DW, or "Wiz" as we like to call it), will pause for a short time while the map is rendered. After the DW returns from the rendering processes, you will hopefully see the bounding box depicting the areal coverage of your domain now displayed against the chosen projection. If you are unhappy with the center point of your domain, you have no other option than to click on the "Start Over" button and start over. That's the least confusing Wiz button name. Note that the projection type, center point location, and update map buttons will no longer be available after clicking "Update Map," but new buttons will have become "live," thus opening new territory for you to explore:



Grayed out means you cannot modify

Play with these instead

The above image shows a primary ARW domain that is defined with a default value for the number of horizontal grid points in NX (100). The initial grid spacing (16.8km) and NY grid points are obtained by dividing the distance across the center of the domain in the West-East direction to obtain DX and then using that value to calculate the number of grid points needed to fill the South-North expanse. If you are creating a domain with a "Latitude-Longitude" projection, then the grid spacing is presented in degrees.

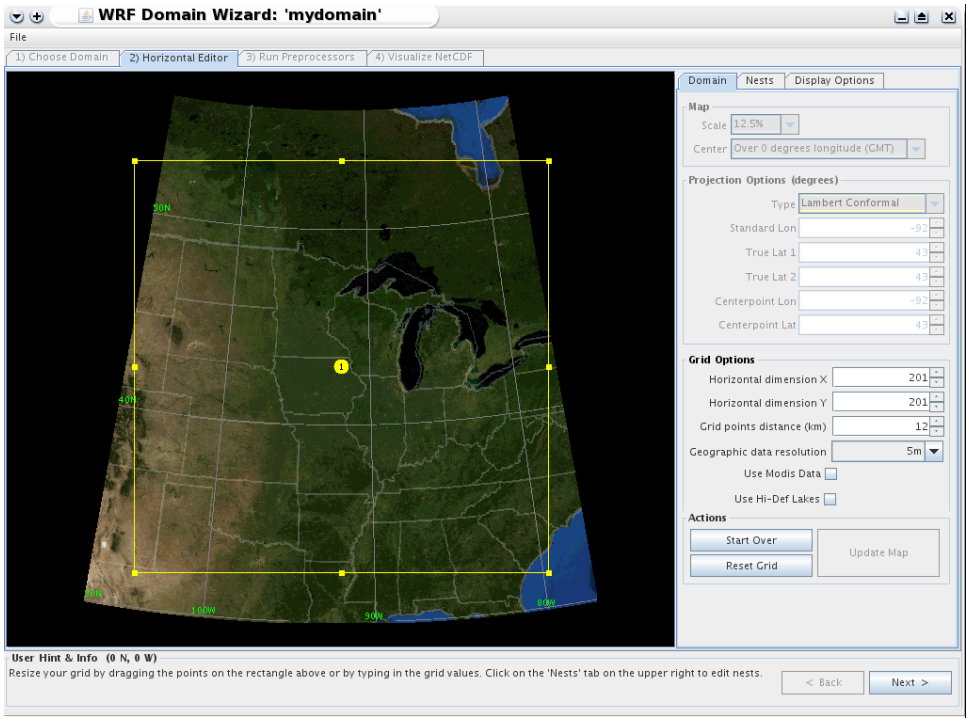
At this point, you will likely want to tune your computational domain by modifying the grid dimensions and spacing. This is achieved through the "Grid Options" section of the domain menu. These values can be changed either by highlighting and editing the settings in the box or by using the increment/decrement buttons. As changes are made to the primary domain, the bounding box that defines the domain also changes. If you decrease the size of your domain, then the box in the window will also decease; however, if you increase the domain size, the bounding box may extend outside of the areal coverage depicted in the window. That is a "feature" in the Wiz, but your domain is fine.

5.2.5.c *Valuable information regarding terrestrial datasets*

Now that you've adjusted your domain to your liking, you have the option to edit the resolution of the terrestrial datasets used to generate the static geographic files, such as topography and land/sea mask. By default, the UEMS DW will select a resolution that is appropriate for the grid spacing used to define your computational domain. Options include 10 minute (~18.5 km), 5 minute (~9.2 km), 2 minute (~3.7 km), and 30-second (~1 km) datasets. If you modify the grid spacing value of an existing domain, the value of your geographic data resolution may also change. If you wish to change the default resolution used, select a different value from the

"**Geographic data resolution**" menu. Don't get too silly though. There is no reason to choose a 30-second resolution dataset when your grid spacing is 24km.

In addition, a feature of the UEMS version of the Domain Wizard is the ability to use the 20-category MODIS land use dataset as an alternative to the 24-category USGS version. To use these data, select the checkbox next to "**Use MODIS Data**" in the window. You also have the option to use the high-definition lake dataset, either alone or with the MODIS dataset, by checking the "**Use Hi-Def Lakes**" box.



Modified 201 x 201 domain at 12 km grid spacing and 5-minute terrestrial dataset resolution

Select "Next" when done, unless you want to make nested domains.

The image above shows a Lambert Conformal computational domain centered over the central US in which the grid dimensions and spacing are modified. The updated domain has 201 points in the NX (~West-East) direction, 201 points in the NY (~South-North) direction and 12km grid spacing. If you don't see it, please contact UEMS support or an eye-care specialist.

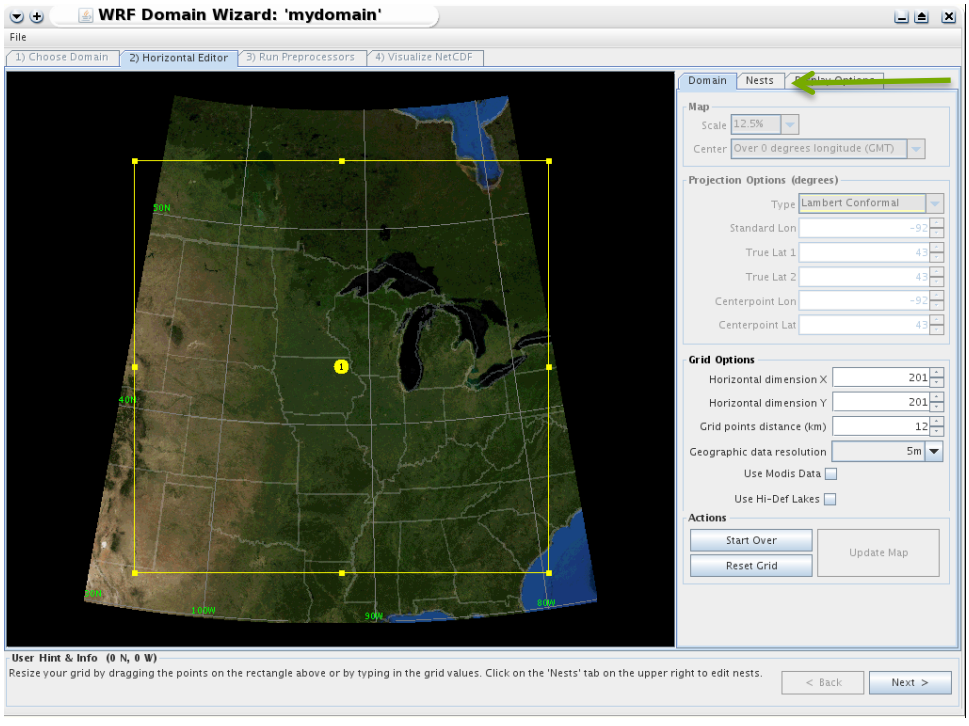
If you would like to create any nested domains, proceed to Section 5.2.7. If you're experiencing feelings of inadequacy with your limited area domain, continue reading to find out about the excitement and fulfillment brought on by global domains. Otherwise, if you don't care about nesting or global domains, skip down to Section 5.2.8, which will take you to the "localization promised land."

5.2.6 Creating nested domains

The WRF ARW core supports concurrent 1- and 2-way nesting, which is neatly integrated into the UEMS. For the uninitiated, 2-way nesting allows for information from a higher resolution child domain to be fed back to its parent domain. The parent domain provides the lateral boundary conditions to the nested domain at each time step and the nested domain is allowed to integrate forward in time with the results fed back onto the parent domain. In the case of 1-way nesting, this feedback mechanism is turned off, so while the parent domain provides the boundary conditions to the nested domain, there is no exchange of information from the nest back to the parent.

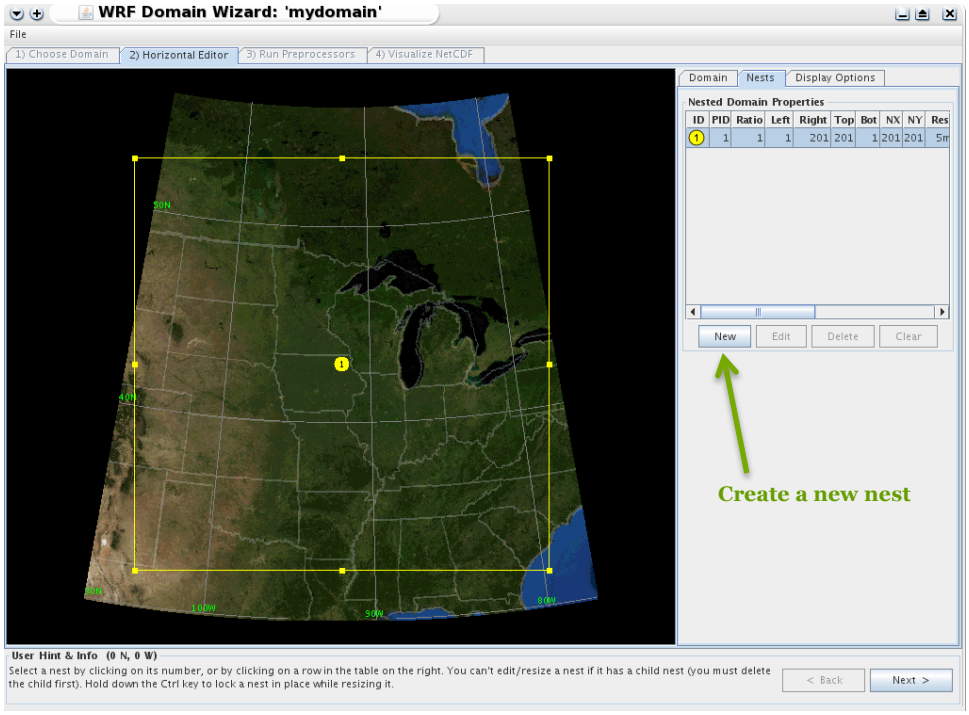
When running the UEMS, all nesting is concurrent with the parent domain, meaning the child domain is executed while the parent domain is running. The benefit of concurrent nesting is that the boundary conditions for the child domain are provided at every parent domain time step. The limitation of this method is that it requires more physical memory on your workstation to run multiple domains simultaneously.

It's lucky for you that the DW may also be used to create nested or sub-domains within your simulation. Once you have settled on your primary domain and clicked the "Update Map" button, the next step is to click on the "Nests" tab to display the nested domain configuration tool. Go ahead, click it. Do it, do it, do it. I double dare you!

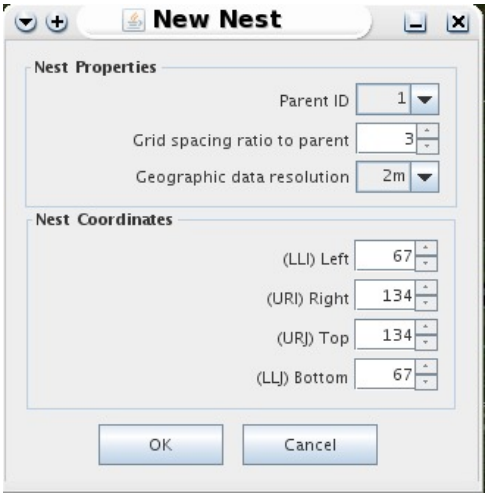


Hey, look here, a "Nests" tab! Go ahead and click it. All the cool kids are doing it.

Since you could not withstand the challenge presented by the "double dare," the "Nests" panel will now be displayed on the right side of the GUI:

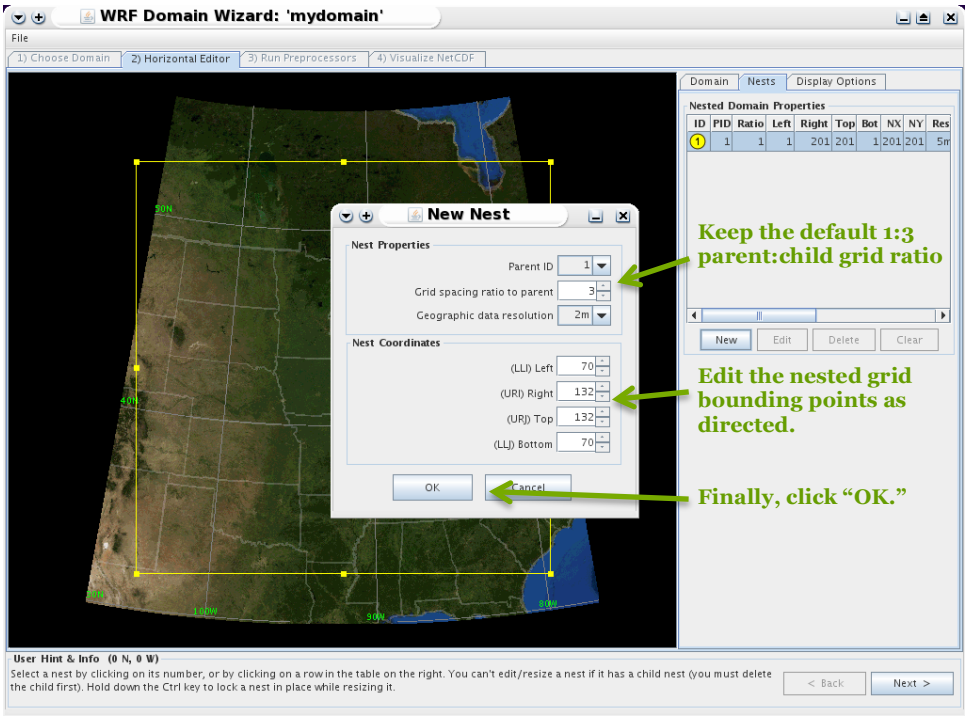


Here you can create new nested domains or edit/delete existing ones. If there are no currently defined nests, then only the "New" button is active. To create a new sub-domain, click on the "New" button to bring up the "New Nest" configuration window.

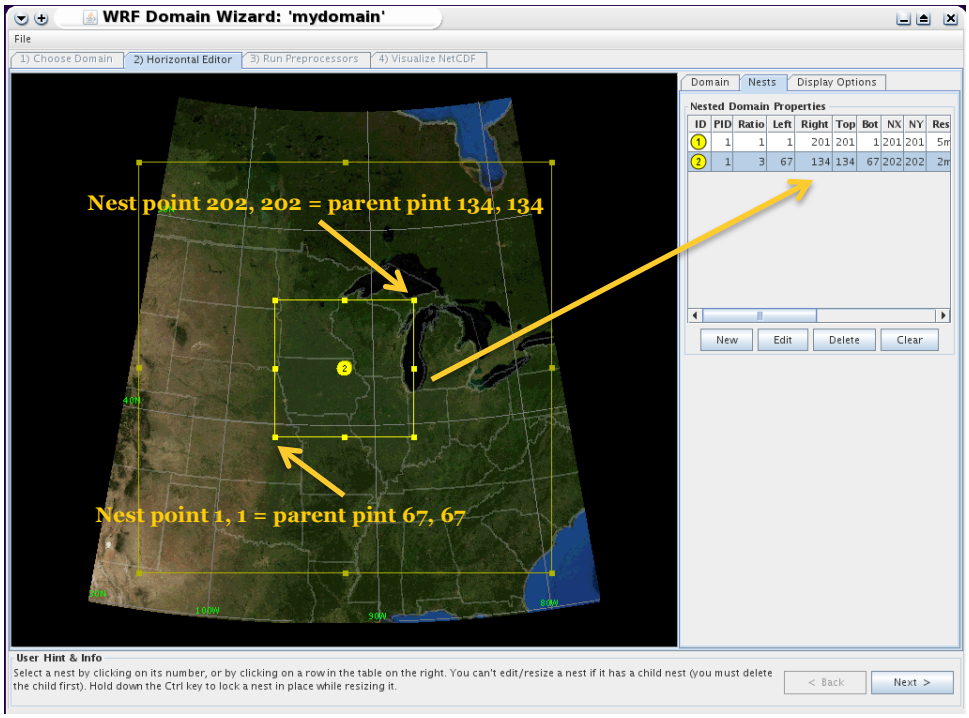


When creating a nested domain, first specify the parent domain with the Parent ID menu. The parent provides the boundary condition updates for the child. Since the domain ID of the primary domain is always 1, the first nested domain you create will have an ID of 2. The second nest you create will have an ID of 3, etc. Note that most values defining a child domain are specified relative to the parent domain (pun intended). In the above example, the new domain (domain 2) is the child of domain 1.

Next, select a "**grid spacing ratio to parent**" value from a list of choices. Note that the default value for the grid space ratio is 3, meaning that the nested domain will have a grid spacing 1/3 that of the parent domain. Since the grid spacing for the primary domain (parent) in this example is 24km, the child domain with have a grid spacing of 8km. **Important**: *The UEMS clairvoyant recommends a parent:child grid space ratio of 1:3, although a ratio of 1:5 may be used with reasonable results for all ARW core nested domains. Using an even ratio, i.e., 1:2, 1:4, 1:6, etc. is (barely) tolerable, but may cause problems in the 2-way feedback within your simulation and with some data processing utilities. Stick with a ratio of 1:3 or 1:5; you will be glad you did.*



Finally, select the "**Geographic Data Resolution**" for your nested domain. An appropriate value is selected by default, but if you don't like it, then change it. Once you have completed this task, click "OK."



Finally, click "Next" when you have finished messing around.

Once these values are chosen, the graphical interface will allow you to manipulate the nested domain using the mouse cursor. You may move the center point or change the lateral boundaries of the domain as necessary. Each nest is constrained to have its corner points coincident with parent domain points, which the GUI will ensure. The DW will insist that a sub-domain is at least five being grid points inside the parent's boundary. The corner point values of the nest are displayed in the entry boxes for the lower left I, J, and upper right I, J. As with any domain, you can adjust the nest as necessary. The nest can be fine-tuned either by manually editing the text values (when creating a new nest or by selecting the nest and clicking the "Edit" button), or interactively using the mouse cursor to change the nest's size and shape.

You can create as many nested domains as desired as long as you don't get carried away, which is always possible when modeling. Be sure to notice the number of grid points used by the nested domains. In the above example, domain 2 is located within domain 1, which has 201x201 points; however, domain 2 has NXxNY dimensions of 202x202 grid points since for every domain 1 grid point there are three for domain 2. **Caution:** *Making nested domains can quickly use up the available memory on your system!*

Valuable information you didn't know regarding the creation of nested domains:

Just because you create a sub-domain doesn't mean you must include it in your simulations. That's correct; the UEMS is designed to withstand even the most egregious incidents of "domain creation debauchery." If you pound down a few too many domains, no worries! You can choose to use (hey another rhyme) a subset of your hoard. It's like a guilty pleasure without any guilt, or pleasure. However, it's up to you to keep track of all the nested domains and parents. The UEMS simulated surrogate nanny can't hold your hand forever.

When you are done creating your computational domain empire and feel giddy from all the excitement, click on "**Next**" to run the localization.

5.2.7 Localizing your stash of domains

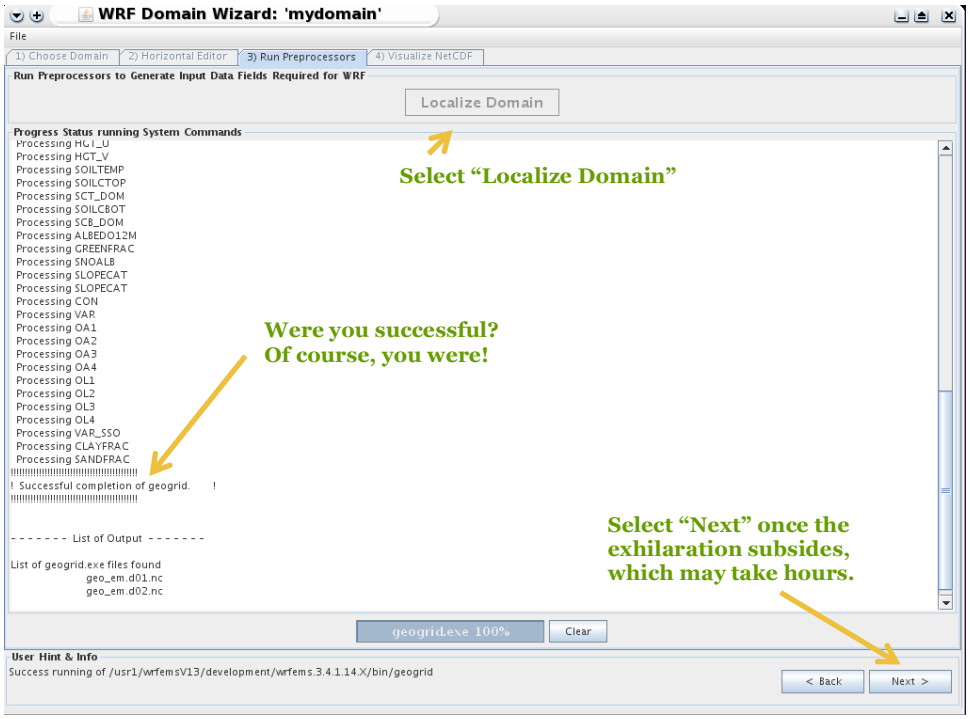
Before your domains can be put to work, they must be localized from the "**Run Preprocessors**" window. It should really be called the "**Localize Your Domain**" window, but it isn't; at least for now. Localization serves to extract the terrestrial data over the area covered by the domain(s) from the large global files at the resolution(s) that you specified during your domain creation "palooza." The files created during the localization processes will reside in the "static" directory underneath your domain directory and are described further in Chapter 6 if you make it that far.

One important thing to know is that the ems_domain.pl script secretly runs in the background when you arrive at the "Run Preprocessors" panel. This script, which is described in nauseating detail in Section 5.3, is responsible for the initial setup of your domain's directories and files before localization. If for some reason this script fails, perhaps by a write-permissions issue, you won't be able to localize your bundle of domain joy since the domain won't exist. You may see an error message in the Domain Wizard, but be sure to check your console window for more information on why the script failed. However, the script should run flawlessly as long as your computing environment is correctly set up.

The time required to localize your domain depends upon the size and number of grid points used to define it, as well as the computer system you are using. If the localization were successful, you would see "Successful completion of geogrid" near the bottom of the window. If the process has failed for any reason, which it almost never does unless someone has done something wrong, check the log files in the "logs" directory. For the sake of progress, it is assumed that you are perfect and your localization was a success. A very fine job!

The steps to localization nirvana are:

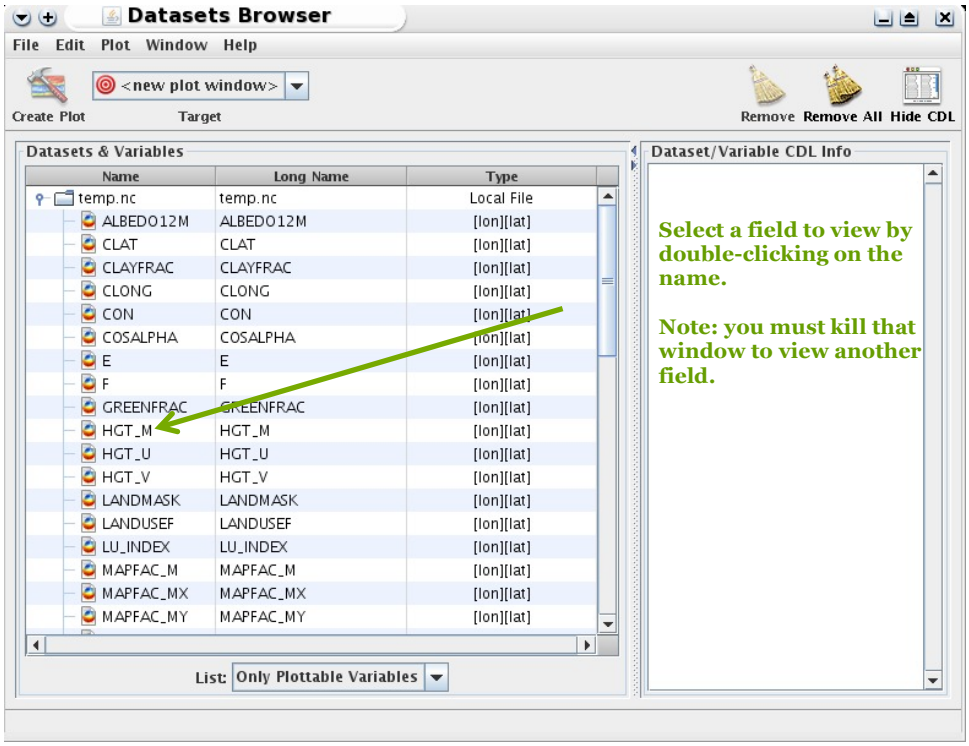
1. **Click on the "Localize Domain" button**
2. **Check if your localization was successful. If "Yes," then**
3. **Select "Next" to view your terrestrial data files**



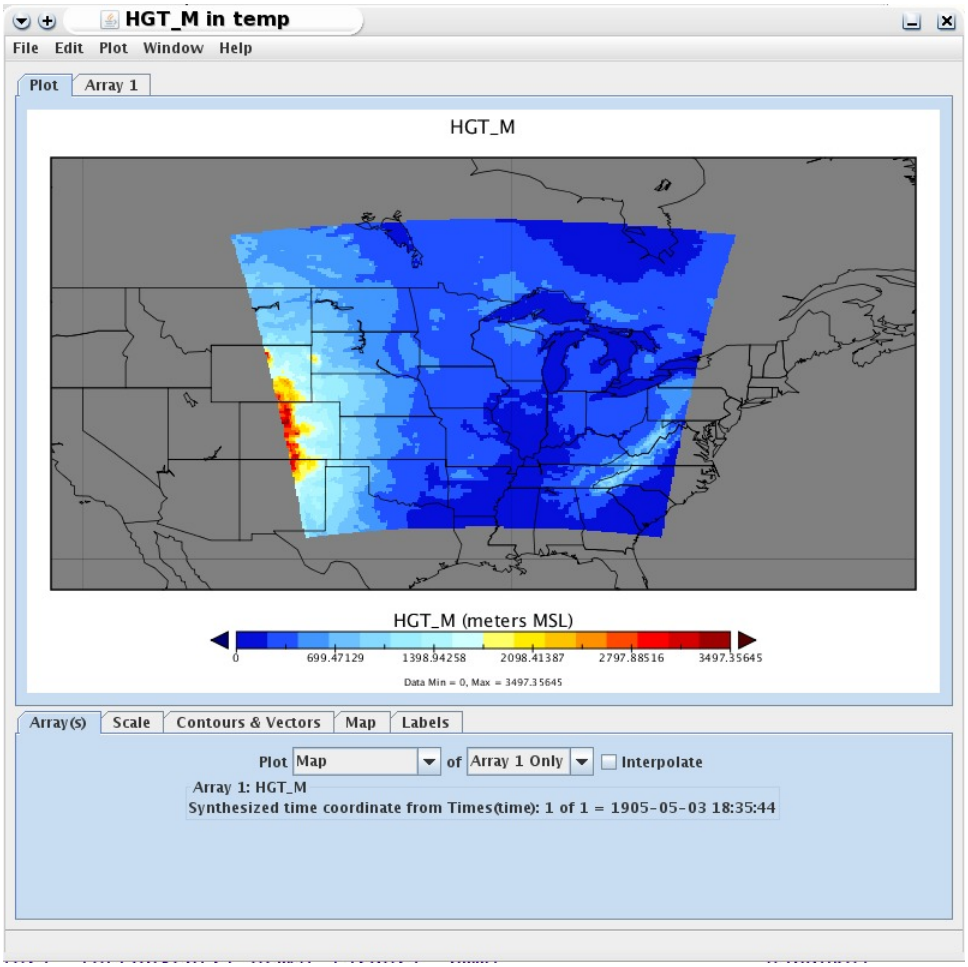
5.2.8 Visualizing the terrestrial data

Provided that your localization was successful, once you select the "Next" button you can view the static terrestrial data used in your simulations. While there are a variety of available tools to accomplish this task, the DW has integrated "Panoply" for visualizing these data. Again, the UEMS has taken care of the messy stuff so you can get back to the important stuff in life, such as being all the modeler that you can be, grammar notwithstanding.

After selecting the domain, click "View in Panoply." You will see a widow that looks something like this one:



To view one of the static fields, double-click on a name in the list. Some of the fields are more interesting than others but feel free to select one that fits your fancy. For starters, it is recommended that you view "**HGT_M**" (surface height at mass points), where you will be entertained by:



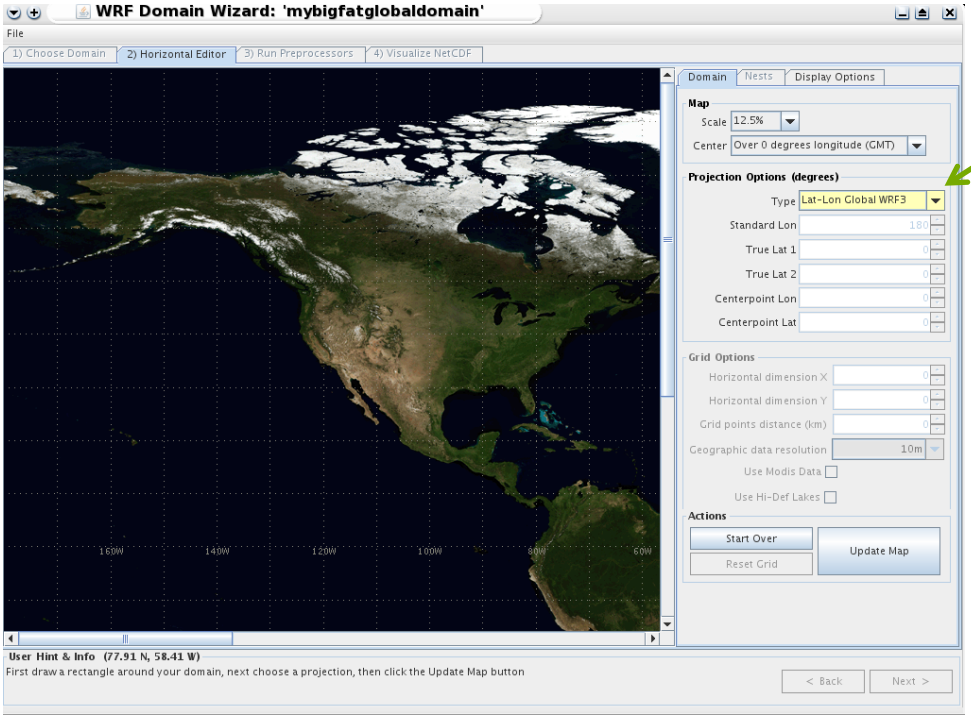
To quit a window, click on the "**X**" in the upper right-hand corner. You will need to kill the Panoply window once you are finished. Once you have completed your tour of the static fields it is time to say "Goodbye" to the DW and "Hello" to running a simulation, only "Goodbye" is spelled "**E-x-i-t**" in the lower right-hand corner of the "**Visualize netCDF**" window.

5.3 Thinking outside the box – Going Global!

So you don't like being governed by lateral boundary conditions? Yeah, me neither! Do you feel as though the NWP "Man" is keeping you down? Well, did you know that with the UEMS, every day is "Primary Domain Emancipation Day"? I bet you didn't know that, did you? Neither did I.

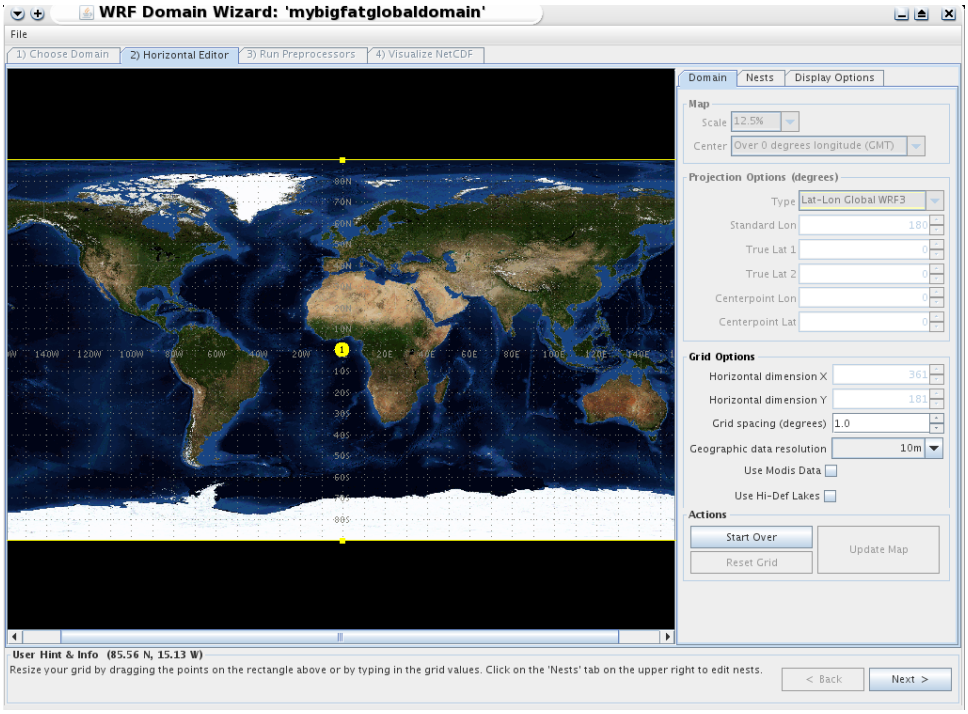
Since you are reading this section, then you must be considering the leap from limited area to global modeling. It's a big move, but I know what you're thinking; "Wouldn't it be cool if I could predict every major freeze event for the Florida citrus crop over the next 250 years? I would make a killing in the futures market!" Well, hold on my commodity cowboy; you are getting a bit ahead of yourself. First, you have to create a global domain before making your fortune. Your dream of being financially set for life will have to wait a few more minutes.

Currently, only the ARW core supports a global domain configuration, which is defined on a cylindrical equidistant latitude-longitude grid. When you started the DW, one of your first tasks was to select a map projection. Fortunately for you, the anxiety in the decision-making process is minimized by the fact that there is only one projection choice. So, to create a global domain, select the "Lat-Lon Global WRF 3" projection from the drop-down menu. You'll be glad you did.



Select "Lat-Lon" Global WRF 3 if you want to play on a sphere, although the earth is not a perfect sphere.

After clicking on "Update Map," your global empire is presented to you as:



"Can't touch this (or these)."

You should immediately notice some things that are different compared to creating limited area domains. The boxes used to define the number of grid points in the NX and NY directions are inactive, (i.e., "Can't touch this"), and the grid spacing is specified in degrees. With the WRF, the only value needed to define a global domain is the grid spacing, which is used for both DX and DY, and is valid at the equator. Once the grid spacing is known, the rest of the information can be calculated easily.

Since you are wearing your "I am super fabulous" thinking cap, you probably realize that the areal coverage of a grid box on a global latitude-longitude domain is maximized at the equator (by the specified grid spacing) and decreases poleward to zero. Specifically, the physical distance between grid points along a latitude circle decreases to zero at the poles, but not along longitude lines. However, zero times something is still usually zero. This simple fact means that the time step used in your simulations will need to accommodate the smallest and goofiest shaped grid boxes to avoid an ugly CFL situation. But what time step is used when the recommended timestep of 6 times the grid spacing is still zero?

Fortunately, the global WRF employs a fast Fourier transform (FFT) filter that will remove high-frequency waves during a simulation from a specified latitude to the poles. This technique will allow you to use a time step larger than zero, which is a good thing if you want to forecast major freeze events. One limitation of this technique is that the allowed number of grid points along a latitude circle is constrained and must adhere to the following equation:

$$NX|NY = 2^P * 3^Q * 5^R + 1$$ (where P, Q, and R are any integers, including zero).

So, you still want to be a global modeler?

Relax - You don't have to figure out whether your value for the number of grid points qualifies. The UEMS version of the DW keeps an internal list of all NX/NY pairs that satisfy the above requirement. So there is no reason to dust off that old HP graphing calculator just yet.

As you increase or decrease the grid space values, the DW will automatically adjust the value so that it satisfies the above requirement. If you enter inappropriate grid spacing, the DW will replace it with the next closest distance that meets the condition.

As mentioned above, the FFT filters are applied from a defined latitude, north and south, to the poles. The default latitude is 45 degrees but can be changed before running a simulation by editing the FFT_FILTER_LAT setting in the **run_dynamics.conf** file; however, it is not recommended that you modify this value.

When defining a nested domain within a global primary domain, it's important that no part of the domain reside poleward of the latitude defined by FFT_FILTER_LAT. Oh, you didn't know that you can create nested domains within your global empire? Go back and read the details in Section 5.2.6 above.

5.4 Domain creation from the command line

So, running a fancy GUI to create your computational domain just ain't for you? You like things done "the old-fashioned NWP way"? Well then, embrace your inner Luddite; the UEMS has you covered.

The UEMS includes the ems_domain.pl utility, which allows you to create and localize computational domains for your simulations. This utility is run as "**ems_domain**" from the command line. The limitation in using ems_domain instead of the DW is that you may* need to manually edit a namelist file to specify the domain configuration and navigation information. Consequently, you must know something about defining WRF domains. Some guidance on this subject is presented below, but you may also need to consult the official ARW guide located in the uems/docs directory.

To be clear, the amount of documentation on the ems_domain utility provided by this "Nearly Complete Guide" is totally pathetic and inadequate. This is what happens when the lone gatekeeper of the UEMS loses enthusiasm for writing documentation. Therefore, you are encouraged to pass the "--help" or "--guide" flags ems_domain to mine the riches that it has to offer.

*Not if you are creating a global domain with ems_domain.

5.4.1 Creating a new domain

If you feel compelled to manually create a new computational domain you can begin by either:

a. Copying an existing domain within the "**uems/runs**" directory,

Or

b. Using the ems_domain.pl utility with the **--newdom** option.

The basic syntax for the command in option "**b**" would be:

% ems_domain.pl --newdom <domain name>

A real-world example might look something like:

% ems_domain.pl --newdom mydomain

Assuming that the above command was successful, a new sub-directory called "mydomain" is created beneath "**uems/runs**." All the necessary configuration and localization files are placed in your newly minted domain directory for you to edit as needed. The new domain is a copy of the benchmark domain, so if the navigation and configuration are OK with you, then your work is done; otherwise, you will need to modify the grid information and re-localize, which means that you might have to continue reading.

5.4.2 Manually configure the navigation for a new domain

Have an existing domain is in need of some navigation tweaks? Then you've come to the right place! Well, almost, as you may also need to consult the official WPS user guide for any information not presented here, which is quite a lot.

When manually defining your domain, you will make changes to the "**static/namelist.wps**" file. The following example of a **namelist.wps** file was "liberated" from the WPS user's guide without consent, and a few of the fields were creatively disguised for UEMS-serving purposes. The fields that can be edited are in boldface below. You don't have to edit all the fields, just those that meet your needs.

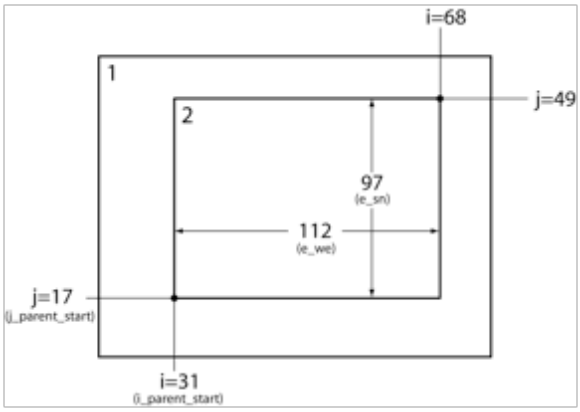
Remember that after you have made any changes to the grid definition or navigation in the namelist file (The "**geogrid**" namelist), you must localize your domain. This part just doesn't happen magically. See the section below on domain localization for additional details.

```
&share
  wrf_core              = 'ARW'
  max_dom               = 2
  start_date            = '2008-03-24_12:00:00','2008-03-24_12:00:00'
  end_date              = '2008-03-24_18:00:00','2008-03-24_12:00:00'
  interval_seconds      = 21600
  io_form_geogrid       = 2
/
```

```
&geogrid
    parent_id           = 1,1
    parent_grid_ratio   = 1,3
    i_parent_start      = 1,31
    j_parent_start      = 1,17,
    e_we                = 74,112
    e_sn                = 61,97
    geog_data_res       = '10m','2m'
    dx                  = 30000
    dy                  = 30000
    map_proj            = 'lambert'
    ref_lat             = 34.83
    ref_lon             = -81.03
    truelat1            = 30.0
    truelat2            = 60.0
    stand_lon           = -98
    geog_data_path      = '/usr1/uems/data/geog/'
/
```

The namelist variables that are affected by nests are shown in the namelist record above. The example shows the namelist variables for a two-domain configuration, a primary domain along with a single nest. The "share" namelist record includes a "**max_dom**" variable, which is set to the total number of domains being created, including the primary domain. The other entries in the "share" section are updated by the ems_prep routine, so don't worry your pretty little head over it. Just move on to the "geogrid" section.

Some of the changes to the "geogrid" namelist record include values for each domain. These fields are **identified in purple**. In the above record, the parent of each nested domain must be specified in the **parent_id** variable. Every sub-domain must have a single parent listed, but a parent may have multiple child domains. The **parent_grid_ratio** field specifies the refinement ratio of a child domain with respect to its parent. This ratio determines the grid spacing of a nested domain in relation to the grid spacing of its parent. The lower-left corner of a child domain is specified as an (*i*, *j*) location in the parent domain with the **i_parent_start** and **j_parent_start** variables. Finally, the dimensions of each domain, in grid points, are given using the **e_we** and **e_sn** variables.



The configuration in the example namelist is illustrated in the figure above showing how each of the variables is determined. The grid point dimensions in the south-north (**e_sn**) and west-east (**e_we**) directions for each child domain must be specified such that the upper-right corner of the grid is coincident with a parent domain grid point. This means that both **e_we** and **e_sn** must be one greater than some integer multiple of the nesting ratio. If you mess up this delicate calculation, the ems_domain wizard behind the curtain will make the necessary adjustments. Finally, for each nest, the resolution of source data from which to interpolate is specified by the **geog_data_res** variable. If you don't know what value to provide for all 27 nested domains you defined, then leave it alone and let ems_domain provide the appropriate values.

If you have completed your task of editing the namelist file, then you may continue with the localization processes.

5.4.3 Need to compensate? Create a global domain!

You most likely read this valuable information previously, but just in case your memory is clouded by personal delusions of grandeur, you can read it again here. Besides, you probably don't remember that you already read it.

Currently, only the ARW core supports a global domain configuration, which is defined on a cylindrical equidistant latitude-longitude grid. Since you are wearing your "I am super fabulous" thinking cap and read the same information in the DW guidance section, you probably know that the areal coverage of a grid box on a global latitude-longitude domain is maximized at the equator (by the specified grid spacing) and decreases poleward to zero. Specifically, the physical distance between grid points along a latitude circle decreases to zero at the poles but not along longitude lines. However, zero multiplied by something is usually zero. This simple fact means that the time step used in your simulations will need to accommodate the smallest and goofiest shaped grid boxes to avoid an ugly CFL condition. But what time step is used when the recommended timestep of 6 times the grid spacing is still zero?

The good news is that the global WRF employs a fast Fourier transform (FFT) filter that will remove high-frequency waves during a simulation from a specified latitude to the poles. This technique will allow you to use a time step larger than zero, which is a good thing if you want to forecast major freeze events. The bad news is the number of grid points in the NX and NY directions is constrained such that they must adhere to the following equation:

$$NX|NY = 2^P * 3^Q * 5^R + 1$$ (where P, Q, and R are any integers, including zero).

Relax - You don't have to figure out whether your value for the number of grid points qualifies. The ems_domain routine does all the work for you.

As mentioned above, the FFT filters are applied from a specified latitude, north and south, to the poles. The default latitude is 45 degrees but can be changed before running a simulation by editing the FFT_FILTER_LAT setting in run_dynamics.conf; however, it is not recommended that you modify this value.

When defining your nested domain, it's important that no part of the domain reside poleward of the latitude defined by FFT_FILTER_LAT. Oh, you didn't know that you could create nested domains within your global empire? Read the details in the section below.

5.4.4 Global domain options and flags

Creating a computational domain that covers the entire globe is fairly easy. Simply run the ems_domain utility with a few options that define the grid spacing and any nested domains you might want include. If you pass the "--help" flag to ems_domain, you will see the following options related to the creation of a global domain:

Option: **--global**

What I Do: Tells ems_domain to create a global domain and ignore much of the other domain information in the namelist.wps file if it exists.

Usage: % ems_domain [other stuff] --global

Description:

Passing the "--global" flag simply tells ems_domain that you wish to create a global domain. Any domain information contained within namelist.wps will be ignored, although a backup of the file will be created. The saved file gets overwritten each time you run ems_domain.pl so if you really want to keep it, you will have to give it another name, such as namelist.wps.nofortune.

Not passing the --global flag does not necessarily mean that you will forgo the creation of a global domain. A global domain may still be created if the geogrid section of the namelist.wps already contains the navigation for a global domain, which might look something like:

```
e_we       = 721
e_sn       = 361
map_proj   = 'lat-lon'
stand_lon  = 180
pole_lat   = 90
pole_lon   = 0
```

Note that parameters **dx, dy, ref_lat**, and **ref_lon**, are not part of a global domain configuration and must not be included in the namelist file. The variables, **ref_x** and **ref_y** will be assigned during the localization process.

Options: **--g_nx** and **--g_ny**

What I Do: The **--g_nx** and **--g_ny** flags define the number of grid points along longitude and latitude circles respectively.

Usage: % ems_domain [other stuff] --global --g_nx 720

Description:

You can pass one or both of these flags to specify the number of grid points over a global domain. The --global flag must accompany them; otherwise, your historical freeze event forecasts will not verify. The options take an integer argument that defines the number of grid points to use in the south-north (--g_ny) and west-east directions (--g_nx).

It is recommended that you only use the --g_nx flag, in which case the ems_domain.pl utility will automatically select a value for --g_ny so that the grid spacing in the NX and NY directions are similar. The value in the NY direction will equal to (NX+1)/2. The rationale for this definition is left to the student, which means you.

Just because you request a specific number of grid points does not mean ems_domain.pl will use your values. There is a WRF global domain requirement that states the number of grid points along a latitude circle must be equal to $NX|NY = 2^P * 3^Q * 5^R + 1$ (where P, Q, and R are any integers, including 0). This limitation is due to the FFT filter implementation. So that you don't have to figure out whether a value for the number of grid points qualifies, the UEMS will select a valid value that is closest to yours, which is usually pretty close.

Option: **--g_dxdy**

What I Do: The **–g_dxdy** flag defines the distance between grid points along the equator.

Usage: % ems_domain [other stuff] --global --g_dxdy 0.5deg

Description:

The --g_dxdy flag can be used an alternative to --g_nx and --g_ny to define the spacing between horizontal grid points along the equator. The grid space value must have a unit identifier appended to it, which can be either "deg" for degrees (e.g. --g_dxdy 0.25deg) or "km" for kilometers (E.g., --g_dxdy 25.5km). The ems_domain routine will calculate the number of grid points for the global domain from this value. The final value may deviate from that requested due to the $NX|NY = 2^P * 3^Q * 5^R + 1$ requirement, but you'll get over it.

Remember that the distance between grid points (say in km) is only constant along a latitude circle and decreases poleward. The DX distance also decreases more rapidly than DY.

Option: **--g_nests**

What I Do: The **--g_nests** flag defines the navigation for sub-domains within a global primary domain.

Usage: % ems_domain --**g_nests** pid:slat:slon:nx:ny:ratio**;**pid:slat:slon:nx:ny:ratio;...

Where:

pid	The ID number of the parent domain
slat	The latitude of point 1,1 (southwest corner)
slon	The longitude of point 1,1 (Use negative for degrees west)
nx	The number of grid points in the NX direction (adjusted to parent points)
ny	The number of grid points in the NY direction (adjusted to parent points)
ratio	The ratio of child to parent grid point distance (either 3, 5, or 7)

Note that:

1. Fields are separated by a colon (:) and nested domains by a semicolon (;).
2. The ID number of a nest (child) begins at 2 and increases sequentially with each requested domain.
3. The Start Lat, Start Lon point are adjusted by ems_domain.pl to be collocated with the closest parent point.
4. NX and NY are the dimensions of a child domain but can be adjusted to parent points. You will be notified if this happens.
5. The user must ensure that the areal coverage of a child domain falls entirely within that of its parent.

Some examples:

--g_nests 1:5:30:151:175:3;2:5:30:151:175:3

Or

--g_nests 1:-5:-100:251:175:3;1:10:-170:151:171:5

Description:

You can use the --g_nests flag to pass the navigation information for as many nested domains that you might want to include in your global simulation. You don't have to include them all in a simulation, but they are available when you need them, just like a stuffy only not as soft. All nested domains are on a limited area latitude-longitude grid, so you will need to specify a parent domain number, a latitude and longitude of the sub-domain start point (point 1,1), the number of grid points in the NX and NY directions (along longitude and latitude circles), and a value for the child:parent grid point ratio-- typically either 3 or 5.

The values for each domain is separated by a colon (:), with multiple domains separated by a semi-colon (;). If necessary, the latitude and longitude of point 1,1 for a nested domain are adjusted so that it is collocated with a parent domain grid point. The final NX and NY values may also be adjusted, so that point NX, NY of your sub-domain is collocated with a parent grid point. Fortunately, these changes are typically minor.

When defining your nested domain, it's important that no part of the domain reside poleward of the latitude defined by FFT_FILTER_LAT. The default value of FFT_FILTER_LAT is 45 degrees N/S and can be changed by editing the run_dynamics.conf file; however, it is not recommended that you change this value.

5.4.5 After the smoke clears, it's time to localize

The next step is trivial, as it only requires that you run ems_domain.pl (again) from the newly created or modified domain directory. This time pass the "--localize" flag:

% ems_domain --localize [domain name]

The purpose of the "--localize" flag is to process the required terrestrial data (land use, topography, landmask, greenness fraction. etc.), for your computational domain. If you make any changes to the navigation or number of points in your domain, you need to run ems_domain with the "--localize" flag. If you want to change a terrestrial dataset, say from modis to nesdis greenness fraction, you need to run ems_domain with the "--localize" flag. If you create a new domain (See --help create) you need to run ems_domain with the "--localize" flag (after your modifications of course).

Embrace the "--localize" flag, and it will embrace you (figuratively, not literally. That would be weird.)

Assuming the localization completed successfully, you are now free to become the master of your domain.

Chapter 6: An Overview of the UEMS Run-Time Domain Directory

Chapter Contents:

6.1 What is a "run-time" directory?

When you successfully create a computational domain, either using the Domain Wizard (DW; Chapter 5) or manually with ems_domain, a "run-time" directory is placed in the location defined by the environment variable $EMS_RUN. By default, $EMS_RUN is set to "uems/runs/," but can reside just about anywhere. The name of the run-time directory corresponds to the name you assigned to it. For example, if you called your domain something clever such as "mydomain," then $EMS_RUN/mydomain (uems/runs/mydomain) directory will exist.

Only run simulations from a run-time directory.

6.2 A run-time directory explained

Each domain directory contains symbolic links and subdirectories that serve a critical function when running a simulation. A more detailed explanation is provided in later chapters (7, 8, 9, & 10), but a brief summary of the more valuable information is presented below.

6.2.1 The run-time routines

As stated in Chapter 3, the run-time routines are the heart and soul of the UEMS. These tools are used to acquire and process initialization data, run simulations, and process the forecast files. They are also used to automate the entire process when running a real-time forecast system. Be kind to these routines, and they will be generous to you.

Each domain directory contains symbolic links, ems_prep, ems_run, ems_post, and ems_autorun that point to uems/strc/Ubin/, where the routines reside. If these links become broken or deleted, you can use "ems_clean --level 0" to repair them.

Here is an all too brief summary of what they do:

ems_prep Used to identify, acquire, and process the datasets used as initial and boundary condition information.

ems_run Ingests the output from **ems_prep**, which is located under wpsprd/, creates the initial and lateral boundary conditions for the simulation, and then executes the model.

ems_post Processes the simulation output located under wrfprd/, and export the files to exotic locations of your dreams.

ems_autorun Automates the process of executing ems_prep, ems_run, and ems_post in succession when creating a real-time simulation experience.

6.2.2 The domain sub-directories

Besides the run-time routine links, there are sub-directories containing files used during a simulation. Here is a summary of those directories and what they contain:

conf/ Contains the configuration files for a simulation. The files for ems_run (Chapter 8), ems_post (Chapter 9), and ems_autorun (Chapter 10) routines are located in similarly named sub-directories beneath conf/.

grib/ Holds the files (GRIB 1 & 2) used to initialize a simulation. The data found in this directory are named according to the convention specified by the gribinfo files located under uems/conf/gribinfo/ (Chapter 7). When ems_prep is run, files used for initialization are placed in this directory before processing.

log/ Contains the log files generated during each step in a simulation.

static/ The static directory contains files used by the UEMS, including:

a. The domain-specific static terrestrial datasets that were created when the computational domain was localized. For the WRF ARW, these files are named "geo_em.d##.nc," where the "##" contains the computational sub-domain for which they were created. These files are in netCDF format and may be viewed with the provided "ncview" and "rdwrfnc" utilities.

b. The WRF and WPS namelist files. The WPS namelist (namelist.wps) file is used during the acquisition and processing of initialization data by ems_prep. The WRF namelist files (namelist.real & namelist.wrfm) are created when running ems_run and contain the model configuration for the WRF ARW. Both namelists include the same information; however, namelist.real is used by the WRF vertical interpolation routine (real_arw.exe), and namelist.wrfm by the WRF model (wrfm_arw.exe). Before running a simulation, ems_run creates symbolic links from "namelist.input" to the appropriate namelist file. *For the most part, users will never edit the namelist files directly*.

c. The **emsupp_cntrl.parm**, **emsupp_auxcntrl.parm**, and **emsupp_hailcast.parm** control files, which manage the fields and levels output to GRIB2 with **ems_post** (Chapter 9).

d. The **emsbufr_stations_d01.txt** file, which contains the BUFR sounding stations to be created with **ems_post**.

wpsprd/ Contains files output from the WPS routines after running **ems_prep**.

wrfprd/ Contains the simulation output after running **ems_run**.

emsprd/ Contains post-processed data files from **ems_post**.

Chapter 7: Using the ems_prep.pl Routine

Chapter Contents:

7.1 "ems_prep.pl? OK, tell me more!

The primary purpose of the ems_prep.pl routine is to identify, acquire, and process the datasets used as initial and boundary conditions in a simulation. It is the first step on the stairway to modeling nirvana. This routine is the most complex of all the run-time scripts as it must sort through a myriad of options to determine what dataset to download, the cycle and forecast times needed, where to obtain and process the files, and then complete a horizontal interpolation to the user's domain. There is a lot of thinking going on behind the ems_prep.pl curtain and it's in your best interest not to peek. It's just like learning how your food is prepared, only different. *So pay no attention to what goes on behind the UEMS curtain.*

As with all the run-time routines, ems_prep.pl is run from the top level of a domain directory. There you will find a link from "ems_prep" to ems_prep.pl, which resides under "uems/strc/Ubin." Beneath "uems/strc/Uprep/ directory are the libraries used by ems_prep.pl. Never run ems_prep.pl directly, and if the ems_prep link gets deleted, it can be recreated by running "**ems_clean –level 0**." For the sake of clarity, "ems_prep" is used throughout this chapter to refer to the link that exists in the domain directory.

7.2 Helping ems_prep help you

In its most rudimentary form, usage of ems_prep looks something like:

```
%    ems_prep --dset DATASET[:METHOD][:SOURCE][:LOCATION][%DATASET[:METHOD][:SOURCE][:LOCATION]] [other flags]
```

The above line might appear both small and confusing at first. A more simplified expression looks something like:

% ems_prep --dset <dataset> [other stuff]

Here, "--dset <dataset>" is the only mandatory flag, as it defines the dataset(s) used to create the initial and boundary condition files. The default information for each dataset is located in a <dataset>*_gribinfo*.conf file, *henceforth known generically as "**gribinfo**,"* where "<dataset>" corresponds to the moniker that you provide as the argument to "--dset." Most of the other flags, i.e., "other stuff," serve to override the default values specified in the *gribinfo* file. If you plan to use ems_prep for real-time forecasting, it is suggested that you modify these default values to minimize the number of flags and options passed. That said, you should run ems_prep directly for real-time forecasting purposes. Instead, it is better to use ems_autorun (Chapter 10), which manages the responsibility for you. *This is just another example of the UEMS "helping you to help yourself."*

Following successful completion of ems_prep, resultant files are located in the "wpsprd/" directory. These files correspond to the dates and times of the data processed for initialization. The files are in netCDF format and used as input to **ems_run (Chapter 8)**, which you will experience shortly.

7.3 The ems_prep configuration files

Unlike the other UEMS run-time routines, there are no local ems_prep configuration files under the conf/ directory. All of the configuration files associated with ems_prep are global and should only be modified by an experienced user, which you will become by the end of this chapter. These are reasonably well documented files, which will help to ease a novice user's "How do I do this?"-fueled anxiety.

Here is a brief description of the ems_prep configuration and ancillary files, in very loose order of importance:

The prep_global.conf file

The prep_global.conf file, located under uems/conf/ems_prep/, contains the global configuration setting available when running ems_prep. There are not many, so you're not missing anything by ignoring this file.

The prep_hostkeys.conf file

The prep_hostkeys.conf file, also located under uems/conf/ems_prep/, contains the data server ID assignments used in the *gribinfo* files. This file contains a list of IDs and associated hostnames or IP addresses used when attempting to acquire files from remote servers, whether it is a computer in another room or a national center. It is only necessary to edit this file when adding another server to the list of available sources.

The *gribinfo* files

The *gribinfo* files, located under the uems/conf/*gribinfo*/ directory, define the default attributes for each dataset available for initialization. Individual files are named as "<dataset>*_gribinfo*.conf",

where <dataset> is replaced with a unique moniker to identify a specific dataset. For example, if "--dset gfs" is passed to ems_prep (more details below), information contained in the "**gfs**_*gribinfo*.conf" file is used for the default values in the initialization. As the "*gribinfo*" name suggests, most of the datasets described by these files are in GRIB 1 or 2 format; however, this is not a requirement. Additional information on modifying these files or adding new datasets is provided in Appendix A.

The Variable Table (Vtable) files

The Vtables specify the fields to extract from the GRIB files for model initialization. Only a subset of the fields contained within the GRIB files are used for initialization. It is usually not necessary to modify the Vtables since adding additional fields does not guarantee that they will be used for initialization. In most cases, you are just wasting your time.

All Vtables are located in the uems/data/tables/vtables/ directory. Each initialization dataset has a Vtable assigned to it by the **VTABLE** or **LVTABLE** parameter in the corresponding *gribinfo* file. Multiple datasets may use the same Vtable, so be careful when making any changes to these files. The simulation you screw up may be your own.

There are two utilities provided with the UEMS that can help when creating or modifying an existing Vtable. The g1print and g2print routines, which are located in the uems/util/bin/ directory, will read a GRIB file and dump information that can be transferred to a Vtable if you ever feel the urge to make changes. Resist the urge!

7.4 What are "Personal Tile" datasets?

To facilitate the acquisition and processing of data for model initialization, UEMS users can employ "personal tile" datasets. Personal tiles are dynamically generated subsets of high-resolution regional and global datasets that are tailored specifically for model initialization. *These datasets retain the full temporal and spatial resolution of the parent data; however, file size and bandwidth are reduced by as much as 98% for a typical mid-latitude domain.* This capability is invaluable when attempting to establish an operational NWP system where bandwidth is limited.

There is no additional configuration necessary before using personal tiles. Everything is dynamic in that all sub-setting occurs on the UEMS servers when a request is made. Changed your domain? No problem. The UEMS does all the work so you can relax.

The available personal tiles are identified by a "**pt**" at the end of a dataset name and are also presented when running "ems_prep --dslist." A request for the GFS personal tile dataset would look something like:

% ems_prep --dset gfspt [flags & stuff]

At which time the magic begins. And you do like magic, don't you?

Currently, there are three active personal tile servers at UEMS world headquarters. In addition to providing access to real-time datasets, each server hosts a multi-week running archive of GFS, RAP,

HRRR, and NAM operational forecasts. Additionally, UEMS ~~victims~~ users have access to an on-line archive of North American Regional Reanalysis (NARR), Global Reanalysis I and II (NNRP), Climate Forecast System Reanalysis (CFSR), and other reanalysis data running historical cases.

7.5 The "must know" ems_prep flags

The most useful flag is probably "--help," which should be obvious in its purpose since it prints a listing of all flags available to the user:

% ems_prep --help

Including the name as an argument to "--help" provides a more detailed description of each flag:

% ems_prep –help dset

To view a list of the supported datasets for model initialization:

% ems_prep --dslist

For a summary of the default settings for a specific dataset:

% ems_prep --dsinfo <dataset>

7.6 All the other ems_prep flags you want to know and love

There are other flags that can be passed to ems_prep. These options serve to override existing settings in the *gribinfo* files and provide you with control over the initialization of your simulation.

A listing of the flags described in this chapter:

Flags: --dset, --sfc, and --lsm

What I Do: Specify and manage the datasets used for model initialization

Usage:

```
% ems_prep --dset dataset[:METHOD:SOURCE:LOCATION]%[DATASET[:METHOD:SOURCE:LOCATION]]
Or
% ems_prep --sfc dataset [:METHOD:SOURCE:LOCATION],[DATASET[:METHOD:SOURCE:LOCATION]],...
Or
% ems_prep --lsm dataset [:METHOD:SOURCE:LOCATION],[DATASET[:METHOD:SOURCE:LOCATION]],...
```

How You Do It:

In their simplest usage, the "--dset," "--sfc," and "--lsm" flags specify the datasets to use for initial and boundary conditions (--dset), surface (--sfc), and land-surface (--lsm) fields respectively.

Note that "--dset <dataset>" is the only mandatory flag while the inclusion of "--sfc" or "--lsm" is optional.

The behavior of these flags is inextricably tied to the parameters and settings found in the associated *gribinfo* files. In particular, the **SERVERS** section (Appendix A), which defines the source(s) for the datasets. Thus, to fully understand the functionality of these flags, you should review the SERVERS section of a *gribinfo* file. Just pick one and read it, they all say the same thing.

For the sake of this discussion, we will assume that the SERVERS section of gfs_gribinfo.conf looks something like:

```
SERVER-FTP = NCEP:/pub/data/nccf/com/gfs/prod/gfs.YYYYMMDDCC/gfs.tCCz.pgrb2fFF
SERVER-FTP = TGFTP:/data/RD.YYYYMMDD/PT.grid_DF.gr2/fh.00FF_tl.press_gr.0p5deg

SERVER-HTTP = STRC:/data/grib/YYYYMMDD/gfs/grib.tCCz/YYMMDDCC.gfs.tCCz.pgrb2fFF
SERVER-HTTP = TOC:/data/RD.YYYYMMDD/PT.grid_DF.gr2/fh.00FF_tl.press_gr.0p5deg

SERVER-NFS = DATA1:/usr1/ems/data/YYYYMMDD/YYMMDDCC.gfs.tCCz.pgrb2fFF
SERVER-NFS = /data/grib/YYYYMMDD/YYMMDDCC.gfs.tCCz.pgrb2fFF
```

The above entries show two FTP sources for initialization files (NCEP and TGFTP), two HTTP sources (STRC and TOC), and two NFS sources (DATA1 and the local system). Each of the server IDs, i.e., NCEP, TGFTP, STRC, TOC, and DATA1 correspond to predefined hostnames or IP addresses located in the uems/conf/ems_prep/prep_hostkeys.conf file. The NFS entry without a server ID specifies the location of the gfs files on the local file system.

So why have entries for remote servers when the data can be obtained locally? Well, this example is for demonstration purposes only; closed course with a professional driver. In other words, you are unlikely to attempt this, but I'm trying to sell a product here.

Let's begin with the simplest usage of "--dset":

```
% ems_prep --dset <dataset>
```

Or how it's done on the street,

```
% ems_prep --dset gfs
```

If you were to run the "ems_prep --dset gfs" command in the example above, ems_prep would use the information located in the SERVERS section of the gfs_*gribinfo*.conf file to acquire the initialization files. The order (HTTP, FTP, and NFS) and sources (server IDs) of the data are semi-randomized so as not to access the sources in the same order every time. If ems_prep is unsuccessful in acquiring the necessary files at the first location, it will move on to the next. This process will continue until all data sources have been exhausted, at which time ems_prep will request that you re-evaluate your life goals. Yes, ems_prep will attempt every possible source identified in the gfs_gribinfo.conf file, because "Working harder so you can slack off!" would be ems_prep's motto if it had one, which it doesn't.

At this point, you should be aware that any files acquired by ems_prep will be written to the <domain>/grib directory and renamed according to the naming convention defined by the LOCFIL parameter in the *gribinfo* file. When you run the routine, ems_prep will first look in <domain>/grib and determine whether the requested files already exist and attempt to obtain any that are missing.

Optional Arguments

There are additional arguments that may be included with the "--dset <dataset>" flag that serve to modify its default behavior:

% ems_prep --dset <dataset>[:[METHOD]:[SOURCE]:[LOCATION]]

Where METHOD, SOURCE, and LOCATION specify the method of acquisition, the source of the files, and the directory location and naming convention used on the remote server respectively. Here is a summary of each:

Placeholder: METHOD

The METHOD placeholder is used to control the method of data acquisition. The default behavior of ems_prep is to attempt each method and source listed in the *gribinfo* file, but you can modify this behavior by including a keyword in the METHOD position shown above. Possible keywords include:

nfs	Only use the **SERVER-NFS** entries in the <dataset>_*gribinfo*.conf file.
ftp	Only use the **SERVER-FTP** entries in the <dataset>_*gribinfo*.conf file.
http	Only use the **SERVER-HTTP** entries in the <dataset>_*gribinfo*.conf file.
nonfs	Don't use the **SERVER-NFS** entries in the <dataset>_*gribinfo*.conf file.
noftp	Don't use the **SERVER-FTP** entries in the <dataset>_*gribinfo*.conf file.
nohttp	Don't use the **SERVER-HTTP** entries in the <dataset>_*gribinfo*.conf file.
none	Don't use any of the methods listed in the SERVERS section. All files are assumed to be correctly named and reside in the <domain>/grib directory.

The above flags should cover just about everything. Here are just a few of examples:

% ems_prep --dset gfs:ftp

Translation: Only use the SERVER-FTP entries in the gfs_*gribinfo*.conf file to acquire data. If ems_prep fails to locate data from NCEP and TGFTP it will not use the other methods listed and you will be devastated.

% ems_prep --dset gfs:noftp

Translation: Do not use the SERVER-FTP entries in the gfs_*gribinfo*.conf file to obtain data. The ems_prep routine will use the other methods listed (NFS and HTTP) to acquire the files.

% ems_prep --dset gfs:none

Translation: Do not attempt to obtain the files as they already are correctly named and reside in the <domain>/grib directory (as explained above). Note that commenting out or deleting all the SERVER entries in the gfs_*gribinfo*.conf file OR by passing may achieve the same behavior:

% ems_prep --dset gfs --nomethod

Placeholder: SOURCE

The SOURCE placeholder is used to specify the source, or server, of the files being requested. It typically takes the form of the server ID as specified in the SERVERS section, (i.e., NCEP, TGFTP, STRC, TOC, and DATA1), and may be associated with multiple methods. For example:

% ems_prep --dset gfs:http:strc

The above command tells ems_prep to only obtain the GFS files from the STRC server via HTTP. The location of the files on the remote server and the file naming convention are obtained from the "**SERVER-HTTP = STRC:**" entry in the gfs_*gribinfo*.conf file.

The use of a METHOD is optional. When excluding METHOD, ems_prep will use all the methods listed that are associated with a given source:

% ems_prep --dset gfs::strc <-- No method specified so use all listed

To obtain files locally that do not have a SOURCE associated with them in the *gribinfo* file, such as in the last SERVER-NFS entry above, use "local":

% ems_prep --dset gfs:nfs:local

Placeholders: METHOD, SOURCE, and LOCATION, just like one big, happy family

The SOURCE may also take the form of a hostname or IP address. This inclusion is best done in combination with METHOD and LOCATION. By using all three arguments, you can request that initialization files be acquired from a location not listed in the *gribinfo* file. The format will look similar to a SERVER entry:

% ems_prep --dset gfs:http:nomad6:/pub/gfs/gfsYYYYMMDD/gfs.tCCz.pgrbfFF

Or

% ems_prep --dset gfs:http:nomads6.ncdc.noaa.gov:/pub/gfs/gfsYYYYMMDD/gfs.tCCz.pgrbfFF

Or

% ems_prep --dset gfs:http:205.167.25.170:/pub/gfs/gfsYYYYMMDD/gfs.tCCz.pgrbfFF

All of the above examples are equivalent, provided that there is a NOMAD6 server ID entry in the prep_hostkeys.conf file. Any placeholders such as YYYYMMDD will be dutifully filled in with the appropriate values. Also, you must specify a METHOD; otherwise something will fail.

Using Multiple Datasets

The "--dset" flag can be used to request different datasets to serve as initial and boundary conditions. For example, if you wish to use 12km NAM files as the initial conditions and 0.5 degree GFS for your boundary conditions, simply separate the two datasets with a "%" in the dataset argument to "--dset", i.e.,

% ems_prep --dset nam%gfs --length 24

Wherein ems_prep will attempt to acquire a single NAM file to use as the initial conditions (00-hour) and GFS files is used for the boundary conditions through 24 hours.

IMPORTANT: The "--length" flag must be used when specifying multiple datasets.

All the optional flags detailed ad nauseam above are available for use with multiple datasets as well. For example, knock yourself out with such classics as:

% ems_prep --dset nam:http%gfs::strc --length 36

Translation: Only use the SERVER-HTTP entries in the nam218_*gribinfo*.conf file to obtain data for use as the initial conditions, and use all the methods listed in the gfs_*gribinfo*.conf file to obtain the boundary conditions files through 36 hours.

And then there's this classic:

% ems_prep --dset nam:http:strc:/pub/nam.tCCz.pgrbfFF%gfs:nfs:local:/data/YYYYMMDD/gfs.tCCz.pgrb2fFF --length 180

Translation: Override any nam_*gribinfo*.conf SERVER entries and obtain the NAM file via HTTP from the source identified by STRC in prep_hostkeys.conf. The data files are located in the /pub directory on STRC with the naming convention of nam.tCCz.pgrbfFF. Also, override the gfs_*gribinfo*.conf SERVER entries and copy (cp) the GFS data files through 180 hours from the /data/gfs/YYYYMMDD directory. Any placeholders such as YYYYMMDD will be dutifully filled in with the appropriate values. Phew!

Using Time-invariant Surface (--sfc) and (Sometimes) Time-Invariant* Land Surface datasets (--lsm)

There are two additional flags used for acquiring and processing of surface and land-surface dataset named "--sfc" and "--lsm" respectively. These flags behave very similar to "--dset", i.e.,

--sfc DATASET[:METHOD:SOURCE:LOCATION],[DATASET[:METHOD:SOURCE:LOCATION]],...
And
--lsm DATASET[:METHOD:SOURCE:LOCATION],[DATASET[:METHOD:SOURCE:LOCATION]],...

The primary difference between "--sfc", "--lsm", and the "--dset" flag is that multiple datasets are separated by a comma (",") and not a "%" as with "—dset."

When multiple datasets are designated, such as:

--sfc sstpt,rtgsst

Or

--lsm lis,rappt

The first dataset listed in the string will take priority in the initialization processes. By using multiple datasets, users can specify a fail-over dataset should a more desired one not be available.

The "--sfc" flag allows users to acquire and process surface fields such as sst or snow cover, which is used to replace or augment a field or fields in the primary initialization dataset specified by "—dset." An example of using the "--sfc" flag would be:

% ems_prep --dset gfs --sfc rtgsst:ftp:nomads

The above use of the "--sfc" flag would replace the sst fields in the GFS dataset with the 1/12th degree sst field from the polar ftp server.

* The determination whether an LSM field is time-variant or invariant is provided by the TIMEVAR parameter in some *gribinfo* files

Fail-over options with Land Surface Datasets (--lsm)

For including surrogate LSM-based fields such as skin temperature, soil moisture and temperature, the "--lsm" flag can be used. A major difference between the "--sfc" and "--lsm" flags is that while the "--sfc" datasets are not required for model initialization, ems_prep will terminate if any of the "--lsm" datasets are missing. To reduce the likelihood of your forecast coming to an untimely demise, fail-over datasets can be specified if your first choice is not available. In this application, the pipe (|) character is used to separate a succession of datasets in decreasing order of desirability (from left to right). If a dataset in this list is not available, ems_prep will proceed to the next. Once a dataset from the list is acquired, the search ends and ems_prep continues with processing of the files; however, if none of the listed fail-over options are found, the entire process terminates and your forecast dreams are dashed once again.

The use of the "|" separator should not be confused with the comma-separated LSM datasets. The comma is used to specify multiple datasets that are mandatory for model initialization, while a "|" is used to specify alternatives to each mandatory data type. Additionally, when using "|" separator with the "--lsm" flag, the entire string must be in quotations. This is due to how the Linux command line interpreter handles the pipe symbol.

Here is a slightly less confusing example:

--lsm "Alsm1|Alsm2|Alsm3,Blsm1|Blsm2" ***Note the use of quotations (")***

Assuming that the datasets listed above mean anything, this example specifies that two LSM datasets are required for model initialization; one from group "A", Alsm1|Alsm2|Alsm3, and one from group "B", Blsm1|Blsm2. The ems_prep routine will first attempt to get Alsm1 from group A. Should that dataset not be available, it will try to get the Alsm2 dataset, and so on. If successful,

the routine will next attempt to obtain Blsm1 from the second group with Blsm2 serving as a fail-over. Should ems_prep fail to locate a dataset from any group, model initialization is over for you.

Flag: **--domains**

What I Do: Provide control over domains included in the simulation

Usage: % ems_prep --domains domain1[:START HOUR],...,domainN[:START HOUR]
Where N <= Max Domains

How You Do It:

Passing the "--domains" flag specifies a list of (nested) domain(s) to initialize for inclusion in a simulation. Any domain(s) must be defined and localized previously when running the Domain Wizard or ems_domain utility. If you created any sub-domains (multiple nests), then passing "--domains" will activate them. You will not be able to run a nested simulation unless you activate the sub-domains!

Important: Domain 1 (primary domain) is the Mother, or Parent, of all domains and is always included by default.

If you plan to start integration on all domains at the same time as the primary domain (Domain 1), then you only need to include the last domain as an argument to "--domains." For example, if you create three nested domains (four total domains) and wish to activate all three, then both:

% ems_prep --dset dataset --domains 2,3,4 (NO spaces between domains)

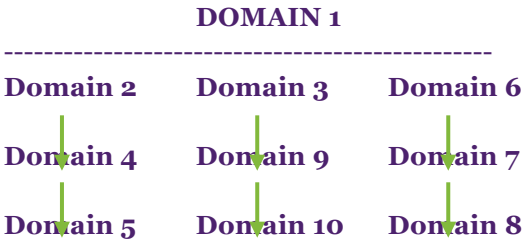
Or

% ems_prep --dset dataset --domains 4

results in domains 1 through 4 being activated.

"But what if I want to start integration of my sub-domains at different times?"

You can control the start time of individual domains by including a ":START HOUR" in the list, where START HOUR is the number of hours after the start of the simulation (Domain 1). To demonstrate the power of "--domains", let's assume that you have created 10 domains (1 Primary and 9 sub-domains) with the following configuration:



In the above example, Domain 1 is the parent of Domains 2, 3, and 6. Domain 2 is the parent of Domain 4, Domain 6 is the parent of 7, Domain 9 the parent of 10, and 7 is the parent of 8, etc.

Some domains will be included in the initialization whether you want them or not, regardless of any relationship within the family tree of domains. This is because, as described earlier, explicitly requesting the initialization of any sub-domain will automatically cause ems_prep to include all the domains from 1 to N, where N is the domain specified. So, if you were to pass "--domains 9" to ems_prep, then Domains (1), 2,3,4,5,6,7, and 8 will also be included in the initialization. The start time for the implicitly included domains will be the same as that of their parent. That's just the way things work here. I don't just make the rules; I also enforce them.

Including a START HOUR

Let's say that the length of the simulation (Domain 1) is 24 hours. However, you want to start Domains 2 and 6, 9 and 12 hours respectively AFTER the start of Domain 1. If this is the way you like to rock then you will need to include the START HOUR in your argument list:

% ems_prep --dset gfs --domains 2:9,6:12

The above example will initialize Domain 2 to start nine hours after Domain 1 (2:9). Domain 6 will begin 12 hours after Domain 1. It should be noted that the start hour for a sub-domain must coincide with a boundary condition update time. If you are using 6-hourly GFS files for your boundary conditions, then you can not start Domain 2, nine hours after the simulation start time, unless you include the "--hiresbc" flag. The "--hiresbc" flag will linearly interpolate the temporally lower resolution BC dataset to hourly update times, thus allowing you to do lots of crazy stuff your mother warned you about like playing with models. I believe this is what she meant.

Here is a more complicated example:

% ems_prep --dset gfs --domains 2:3,3:12,5:15,7:6,8:24,9:6

There is a lot of stuff going on here. First, note that Domain 10 is not implicitly or explicitly included in the list, so it will not be included in the initialization. Additionally,

- Domain 2 is (explicitly) initialized to start three hours after Domain 1
- Domain 4 is (implicitly) initialized to start three hours after Domain 1
- Domain 5 is (explicitly) initialized to start 15 hours after Domain 1
- Domain 3 is (explicitly) initialized to start 12 hours after Domain 1
- Domain 9 is (explicitly) initialized to start six hours after Domain 1
  However, since Domain 9 is the child of Domain 3, this start hour is BEFORE the parent domain, so it will be overridden and Domain 9 will start 12 hours after Domain 1.

- Domain 10 is (explicitly) turned OFF
- Domain 6 is (implicitly) initialized to start at the same time as Domain 1
- Domain 7 is (explicitly) initialized to start 6 hours after Domain 1
- Domain 8 is (explicitly) initialized to start 24 hours after Domain 1

  However, since the total length of the domain 1 simulation is 24 hours, *Domain 8 will automatically be turned off!*

Flag: **--[no]scour**

What I Do: Controls the amount of cleaning done before the start of ems_prep

Usage: % ems_prep --[no]scour [more important stuff]

How You Do It:

Passing the "--scour" flag overrides the default level of cleaning done before the start of ems_prep. If this flag is not passed, the default level is the same as running "ems_clean --level 3", which deletes all non-essential files in the domain directory while keeping any initialization data under grib/.

Passing "--scour" is the same as running "ems_clean --level 4", which also includes the removal of all files from grib/. You might include this flag if you wanted to download a fresh set of GRIB files for initialization while deleting any existing files.

Passing "--noscour" is the same as running "ems_clean --level 0", which deletes old log files and recreates the symbolic links to the run-time scripts.

Flag: **--date**

What I Do: Specify the date of the dataset to use for initialization. NOTE: not for use with real-time forecast applications! No, just don't do it.

Usage: % ems_prep --date [YY]YYMMDD [other important stuff]

How You Do It:

Passing the "--date" flag specifies the date of the dataset used for model initialization. Usually, this will also be the initialization date of the simulation, but this isn't always the case (see below).

The Default Behavior When NOT Passing "--date":

Not passing the "--date" flag will cause ems_prep to use the date (YYYYMMDD) of the most recent model run for which data are available, which is normally the current system date. So if today is 30 February 2021, ems_prep will use a value of 20210230 unless the "--date" flag is passed with a better date. There is an exception to this rule in that when the most recent date for which data are available is from the previous day, then ems_prep will adjust accordingly. Just remember: "The UEMS thinks, so you don't have to," which is poor grammar but gets the point across.

Behavior When "--date" is Passed:

Passing the "--date" flag overrides the default behavior described above and tells ems_prep that

you want to use a dataset from YYYYMMDD to initialize your simulation. Again, while this typically will be the same date as your model initialization, it doesn't have to be the case. For example, if you passed:

```
%   ems_prep --dset gfs  --date 19751109  --cycle 12:24
```

then you are instructing ems_prep to use the 24-hour forecast from the 9 November 1975 12 UTC cycle (12:24; see --cycle) of the GFS (if it existed) to initialize your simulation. Consequently, your initialization 00-hour date will be 10 November 1975 for all you maritime disaster aficionados.

Important: *This flag is not to be used for real-time modeling purposes!*

Did You Know? : The ”--date” flag works in harmony with "—cycle,” and so should you.

Flag: **--cycle** HOUR[:INITFH[:FINLFH[:FREQFH]]]

What I Do: Defines the cycle times (UTC) of the forecast or analysis system from which the data files are available. For an operational forecast model, this is often the 00-hour time of that run. For an analysis system, the cycle time typically identifies the hour at which the dataset is valid. NOTE: not for use with real-time forecast applications! No, just don't do it.

Usage: % ems_prep --cycle HOUR[:INITFH[:FINLFH[:FREQFH]]] [other stuff]

How You Do It:

Important: *This flag is not to be used for real-time modeling purposes!*

First - The Default Behavior When NOT Passing "--cycle":

Not passing the "--cycle" flag will cause ems_prep to use the UTC cycle time of the most recent model run from which data are available. The list of available cycles is already defined by the CYCLES parameter in each of the *gribinfo* files. To determine the most current available cycle time, ems_prep accounts for the amount of time required for an operational model to run and process any data files for distribution. This delay between the official UTC cycle time and when the data files first become available is defined in the DELAY parameter in the *gribinfo* file.

For example, if the 12 UTC GFS takes three hours to run and process forecast GRIB files for distribution (DELAY=3), ems_prep will not attempt to obtain the data until after 15Z. If ems_prep is run, say at 14:55 UTC (again, without "--cycle"), data from the 06 UTC cycle will be obtained because that is the most current run from which data are available.

Now the Behavior When "--cycle" is Passed:

The --cycle flag specifies the cycle time of the model dataset to use for initialization of your simulation. The general usage is:

```
%  ems_prep --dset gfs --cycle CYCLE
```

When you specify the cycle time of the dataset to be acquired, ems_prep will attempt to access the most recent dataset available corresponding to that cycle time (assuming "--date" was not passed). So using the GFS example above, passing "--cycle 12" at 14:55 UTC will cause ems_prep to obtain data from the previous 12 UTC run as opposed to the 06 UTC cycle if "--cycle 12" had not been passed (default).

But wait, there's more whether you want it or not.

The --cycle flag accepts additional arguments that override the initial forecast hour, final forecast hour, and frequency of the boundary condition files, the default values of which are defined in each *gribinfo* file as INITFH, FINLFH, and FREQFH respectively. The format for the argument list is:

```
% ems_prep --dset gfs --cycle HOUR[:INITFH[:FINLFH[:FREQFH]]]
```

Where the brackets indicate that everything optional. These optional arguments are passed as a string, with each separated by a colon (:). Trailing colons need not be included.

Here are a few examples. Feel free to make some of your own.

```
% ems_prep --dset gfs --cycle 00:00:24:03
```

Translation: Use the 00 UTC cycle time, the 00-hour forecast for the initialization time, the 24-hour forecast for the final BC time (thus a 24 hour forecast), and use 3-hourly files for boundary conditions. In this example, ems_prep will attempt to download the 00, 03, 06, 09, 12, 15, 18, 21, and 24-hour forecast files from the most recent 00 UTC cycle of the GFS. All default values for these parameters are overridden, or in UEMS World Headquarter's parlance, "Crushed."

More examples - Just because we're having so much fun together:

```
% ems_prep --dset gfs  --cycle 06:06:30
```

Translation: Use the 06 UTC cycle time, the 06-hour forecast for the initialization hour and the 30-hour forecast for the final boundary condition time (a 24 hour forecast). Because the BC update frequency (FREQFH) was not passed, the default value in gfs_*gribinfo*.conf file is used.

Tell me if you've heard this one before. What is the difference between the following:

```
% ems_prep --dset gfs --cycle CYCLE:INITFH:36:12
```
And
```
% ems_prep --dset gfs --cycle ::36:12
```

Answer: NOTHING! In both cases, the data files from most current available cycle hour is used (default) starting from the default initialization forecast hour found in gfs_*gribinfo*.conf (INITFH) through the 36-hour forecast, with a 12-hourly BC update frequency. So if INITFH = 0 in gfs_*gribinfo*.conf, then the 00, 12, 24, and 36-hour forecast files from the most current run of the GFS is used to initialize your simulation. The inclusion of 'CYCLE:INITFH', or 'CYCLE:INITFH:FINLFH:FREQFH' are not necessary but still valid.

If you choose not to use the placeholders, then remember that leading colons are required, but trailing ones are not:

```
% ems_prep --dset gfs --cycle :::12
```

Use all default values except 12 hourly BC updates.

```
% ems_prep --dset gfs --cycle :24:84
```

Use the defaults for cycle & BC update use 24- through 84-hour forecast files.

You have the power; use it wisely and often.

Note: The period, or length of the forecast, may also be overridden with the "--length" flag. This option usurps the length of the forecast defined by any other method.

Also Note: The DELAY setting in *gribinfo* file can be overridden with the "--nodelay" flag.

Important: *For the final time, this flag is not to be used for real-time modeling purposes.*

Flag: --syncsfc

What I Do: Synchronize surface datasets to simulation initialization (00-hour) time

Usage: % ems_prep [other stuff] --syncsfc [<sfc dataset>,<sfc dataset>,...]

How You Do It:

Passing "--syncsfc", tells the ems_prep to synchronize the valid hour of the specified surface dataset(s) with the initialization time of the model simulation, i.e., only use those data that are valid near the simulation start hour. The argument to "--syncsfc" is a list of surface datasets, separated by commas, to which this requirement is applied. Should a dataset not be available near the simulation start date and time, ems_prep will look for data 24 hours earlier rather than use the next closest verification time. The number of previous 24-hour periods (days) searched is defined by the AGED parameter in the *gribinfo* associated with each surface dataset listed.

The "--syncsfc" flag can be passed with or without an argument. Any surface dataset listed also must be included with the "--sfc" flag; however, not all "--sfc" datasets must be included with "—syncsfc." If no dataset is specified, i.e., just "--syncsfc," then all "--sfc" datasets are used whether you like it or not.

Why do this?

Some datasets, such as MODIS SSTs, have a diurnal variation that needs to be taken into account when used to initialize a simulation. It may not be appropriate to use a dataset from 00 UTC for a

12 UTC run start, even if that file time is closest to the simulation start date. If "--syncsfc" is passed, ems_prep will look for data valid near 12 UTC from the previous day rather than use data from 00 UTC the current day. Note that the times do not have to exactly match model initialization since the verification hour closest to the model initialization will be determined.

For use with **"--sfc"** datasets only!

Flag: **--local**

What I Do: Instructs ems_prep to only look in the <domain>/grib directory for initialization files

Usage: % ems_prep [other stuff] --local

How You Do It:

Pass the "--local" flag if you want the ems_prep routine to only look in the local grib/ directory for the initialization files. This flag does the same thing as specifying "--dset <dataset>:local," but too much of a good thing always makes for an even better thing, so "--local" was added. Enjoy them together!

Flag: **--length HOURS**

What I Do: Specify the simulation length (hours) for the primary domain

Usage: % ems_prep --length HOURS [other stuff]

How You Do It:

Passing the "--length" flag overrides the value of for the final hour of a simulation (FINLFH; See: "--cycle" flag) in defining the maximum length of a simulation (excluding global).

Passing:

% ems_prep --dset <dataset> --cycle 00:06 --length 36

is the same as passing:

% ems_prep --dset <dataset> --cycle 00:06:42:03 (Note: 42-6 = 36)

The "--length" flag overrides everything when defining the length of your Simulation. The "--length" flag is king. All hail "—length!"

Important: *The "--length" flag must be used when specifying separate datasets for initial and boundary conditions (See: "--dset" flag)*

Flag: **--analysis**

What I Do: Tell ems_prep to use a series of analyses rather than forecasts for initialization

Usage: % ems_prep [other stuff] --analysis [FCST HOUR]

How You Do It:

Passing the "--analysis" flag results in your simulation being initialized from a series of analyses rather than a single forecast. The default behavior of ems_prep is to look for a sequence of forecast files from a single model run to serve as the boundary condition update times. For example, when passing:

% ems_prep --dset gfs --date 20170230 --cycle 12 --length 24

The ems_prep routine will look for 24 hours of forecast files from the 12 UTC cycle run of the operational GFS from 30 February 2017 to initialize your simulation.

However, by passing the --analysis flag, you can initialize your simulation from a succession of cycle runs, in which case the 00-hour (default) forecast files from each of the GFS cycles runs beginning with 30 February 2017 through 31 February 2017 will be used.

This flag is primarily intended for use with historical or reanalysis datasets where a series of forecasts is not available. This limitation would apply to datasets such as the North American Regional Reanalysis (NARR), Climate Forecast System Reanalysis (CFSR) or the ECMWF ERA Interim analysis dataset. While this flag can be used with operational model guidance such as the GFS, it is not recommended since the BC update frequency is likely to be much lower than that you would get from the available forecast files.

But wait, there's more! What if you're feeling all wild & crazy and want to initialize your simulation from a series of nonzero-hour forecasts? Well then, you can just pass an argument to the "—analysis" flag like:

% ems_prep --analysis 24 [Other stuff]

Which will initialize your simulation from a series of 24-hour forecasts. Mind blowing, eh?

However, if you are seriously considering such actions, you may want to get a good nights rest, as chances are that you are suffering from some serious sleep deprivation. This flag will still be there in the morning.

Sweet dreams.

Flag: **--nudge**

What I Do: Process the domains for 3D Analysis/Spectral nudging

Usage: % % ems_prep --nudge [crazy other stuff, but not too crazy]

How You Do It:

Passing the "--nudge" flag instructs ems_prep to process the initialization dataset files for use with 3D analysis or spectral nudging during the simulation. While this step is required if you intend to use analysis or spectral nudging, you still have the flag to turn it off later on (See Chapter 8 of the "UEMS Guide to Simulation Excitement").

Flag: --nodelay

What I Do: Override the DELAY setting in the *gribinfo* file

Usage: % ems_prep [other stuff] --nodelay

How You Do It:

Passing the "--nodelay" flag will turn off (set to 0 hours) the DELAY value defined in the *gribinfo* file for each dataset being used for initialization. Using the "--nodelay" flag is kind of like saying "I want it now!" even though this effort might be futile.

Flag: --hiresbc

What I Do: Interpolate between available file times to create hourly BC update files

Usage: % ems_prep [wishful thinking] --hiresbc

How You Do It:

By passing the "--hiresbc" flag, you are requesting that 1-hourly boundary condition update files be created instead of the default frequency, which would be the same as the temporal frequency of the boundary condition files downloaded for model initialization. This flag serves to provide no benefit to the quality of the model simulation, but instead will allow:

1. The user to start the integration of a child domain at a time that does not coincide with a default boundary condition update time.

2. A simulation to end at a time that does not coincide with a default boundary condition update time.

For example, if your boundary condition files were 3-hourly, but you wanted to run a 2 hour simulation, or if you wanted to start the integration of a child domain 2 hours after the start of the primary domain. Passing "--hiresbc" would allow you to accomplish this task.

Flag: **--nointdel**

What I Do: Do not delete the intermediate files after processing (Default: Delete)

Usage: % ems_prep -- nointdel [cool stuff]

How You Do It:

Pass the --nointrdel flag if you do not want the processed WRF intermediate files scoured from the wpsprd directory following successful completion of ems_prep. The default behavior is to delete these files since they are no longer needed.

The WRF intermediate files contain information extracted from the GRIB files for the fields identified by the designated variable table (Vtable) and written out in a binary format. Typically, you do not need to keep the intermediate files since they are processed into WRF netCDF for use in creating the initial and boundary condition files. However, there are times when you need to do some troubleshooting and thus require these files for the investigation.

Flag: **--noproc**

What I Do: Do not process the GRIB files for model initialization after an acquisition

Usage: % ems_prep --noproc [make animal sounds]

How You Do It:

Passing the "--noproc" flag will instruct ems_prep to acquire the requested GRIB files for model initialization but do not process them for use in a simulation. The GRIB files will be placed in the <domain>/grib directory and named according to the convention defined in the *gribinfo* file, but that's all the action you're going to get.

Flag: **--benchmark**

What I Do: Tell ems_prep to processes initialization data for the benchmark case

Usage: % ems_prep --benchmark [--domain 2]

How You Do It:

Passing the "--benchmark" flag tells ems_prep to process the initialization data for the 27 April 2011 benchmark simulation provided with the UEMS. This flag is only valid when running the benchmark case, located under uems/util/benchmark/27april2011; otherwise, it will likely have no effect although cases of infatuation with the UEMS developer have been reported.

Flag: **--dsquery|dsinfo DSET**

What I Do: Query the default configuration settings for the specified dataset

Usage: % ems_prep –dsinfo DSET

How You Do It:

The "--dsquery" flag allows you to review the contents of a <dset>*_gribinfo*.conf file, which provides the default configuration values including a description of the dataset, file sources, naming conventions, and other valuable information.

The argument to "--dsquery" is the moniker used to identify a specific dataset, a list of which is provided by the "--dslist" flag. Loads of good information presented.

Here is an example for the gfsp25 dataset:

% ems_prep --dsquery gfsp25

Default settings from the gfsp25 GRIB information file: gfsp25*_gribinfo*.conf

Description: GFS 0.25 degree dataset on pressure surfaces

Dataset Category	:	Forecast
Vertical Coordinate	:	Pressure
Default Initial Forecast Hour	:	00
Default Final Forecast Hour	:	24
Default BC Update Frequency	:	03 Hours
Available Cycles (UTC)	:	00, 06, 12, 18
Remote Server Delay	:	03 Hours
Local Filename	:	YYMMDDCC.gfs.tCCz.0p25.pgrb2fFFF
Vtable Grid Information File	:	/uems/data/tables/vtables/Vtable.GFS
LVtable Grid Information File	:	None
Maximum BC Update Frequency	:	03 Hours

METHOD	HOST	LOCATION
HTTP	nomads.ncep.noaa.gov	/data/YYYYMMDDCC/gfs.tCCz.pgrb2.0p25.fFFF
HTTP	www.nomads.ncep.noaa.gov	/data/YYYYMMDDCC/gfs.tCCz.pgrb2.0p25.fFFF

Flag: **--dslist**

What I Do: Provide a summary of available initialization datasets

Usage: % ems_prep --dslist

How You Do It:

Passing the "--dslist" flag provides a brief summary of the datasets supported by the UEMS for initializing a simulation. Further information about a specific dataset can be obtained by passing the "--dsquery" flag along with the corresponding moniker.

Flag: **--timeout**

What I Do: Override the default MPICH timeout value when doing the horizontal interpolation

Usage: % ems_prep [more other better stuff with benefits] --timeout [seconds]

How You Do It:

Passing --timeout serves to override the value of TIMEOUT located in the default prep_global.conf file. The purpose of TIMEOUT is to avoid problems with the WRF horizontal interpolation routine (metgrid) hanging after processing of the intermediate files into netCDF. On most systems, this is not an issue and processing ends normally, but sometimes the routine will fail to exit even though all files have been successfully processed. In that event, the TIMEOUT setting will define the length of time (seconds) from the beginning of processing until the horizontal interpolation is forcefully terminated.

Note that the TIMEOUT period starts at the beginning of the horizontal interpolation, so for some very large datasets or simulations with many boundary condition files it is possible to exceed the timeout period while processing is ongoing. If this situation occures, the simulation will fail during initialization. In most cases though, the default setting of 1199 seconds should be sufficient.

Finally, passing "--timeout 0" turns the timeout option OFF (no time limit).

Flag: **--bndyrows #ROWS**

What I Do: Specify the number of outer rows to use as lateral boundary conditions

Usage: % ems_prep --bndyrows 8 [fill in the blank]

How You Do It:

If you are looking for a flag lacking in sex appeal, then you've come to the right place. Passing the "--bndyrows" flag overrides the default number of rows to process for use as lateral boundary

conditions. The default value is 5, which should be good enough for you, but if you insist upon changing it then knock yourself out and pass "--bndyrows."

Did You Know?: The maximum value for #ROWS is 10. Don't make me play the enforcer!

Flag: **--noaerosols**

What I Do: Do not include the monthly aerosol climatology in the initialization dataset

Usage: % ems_prep --noaerosols [way more important stuff]

How You Do It:

Pass the "--noaerosols" flag if you do not want to include the Thompson Water/Ice Friendly Aerosols in the initialization dataset. These "WIF" data henceforth known generically as with the Thompson "Aerosol Aware" microphysics scheme (MP_PHYSICS = 28) during the simulation. By default, the UEMS includes these data regardless of your future choice of microphysics, so that when you are ready, you can go "Aerosol Aware" without going back to ems_prep.

The downside of this approach is that the initialization files in wpsprd/ are much larger with the inclusion of these data. So if this fact bothers you to the point of giving you a frowny face, then the "--noaerosols" flag is here for you.

Just because the UEMS HATES frowny faces!

Did You Know?:

Since you asked, here is a summary of the aerosol climatology data from the WRF website:

> "Aerosol number concentrations were derived from multi-year (2001-2007) global model simulations (Colarco, 2010) in which particles and their precursors are emitted by natural and anthropogenic sources and are explicitly modeled with multiple size bins for multiple species of aerosols by the Goddard Chemistry Aerosol Radiation and Transport (GOCART) model (Ginoux et al. 2001). The aerosol input data we used included mass mixing ratios of sulfates, sea salts, organic carbon, dust, and black carbon from the 7-year simulation with 0.5-degree longitude by 1.25-degree latitude spacing. We transformed these data into our simplified aerosol treatment by accumulating dust mass larger than 0.5 microns into the ice nucleating, non-hygroscopic mineral; dust mode, NIFA, and combining all other species besides black carbon as an internally-mixed cloud droplet nucleating, hygroscopic CCN mode, NWFA. Input mass mixing ratio data; were converted to final number concentrations by assuming log-normal distributions with characteristic diameters and geometric standard deviations taken from Chin et al.; (2002; Table 2)."

Flag: **--noaltsst**

What I Do: Do not use the derived daily-average surface air temperatures in the absence of actual SST data

Usage: % ems_prep --noaltsst [I forgot]

How You Do It:

Passing the --noaltsst flag instructs ems_prep not to use the WRF derived daily-average surface air temperatures for water temperatures in the absence of data when initializing lake SSTs. This flag only applies to simulations 24 hours or longer run on domains localized with the modis_lakes or usgs_lakes terrestrial dataset. If this does not describe your simulation, then please entertain yourself with another "--help" topic.

The Background Information (Selectively liberated from the WRF User's Guide):

The treatment of water temperatures, both for oceans and lakes, normally involves simply interpolating the SST field to all water points in the WRF domain. However, if the lakes that are not well resolved in either the WRF domain or GRIB data, and especially if those lakes are geographically distant from resolved water bodies, the SST field over lakes will most likely be extrapolated from the nearest resolved body of water in the GRIB data. This situation can lead to lake SST values that are either unrealistically warm or unrealistically cold.

An alternative to extrapolating SST values for lakes is to manufacture a “best guess” at the SST for lakes. This step is accomplished by using a combination of a special land use dataset that distinguishes between lakes and oceans, and a field to be used as a proxy for SST over lakes. The special land use dataset is necessary, because WRF needs to know where the manufactured SST field should be used instead of the interpolated SST field from the GRIB data.

Now, How It Happens in the UEMS:

By default, the UEMS uses the alternative initialization method for lake SSTs described above when a domain is localized with the modis_lakes or usgs_lakes dataset, and the simulation is greater than 24 hours in length. However, there exists a potential problem because the alternate SST field is based on air temperature, it sometimes does not accurately represent the water temperatures during certain parts of the year. For example, sub-freezing air temperatures may be assigned to water points when climatologically; the water temperatures are much warmer. This mis-assignment can result in open water being treated as ice covered during the simulation. As a possible workaround to this issue, the "--noaltsst" flag may be passed that effectively turns off the alternative initialization method, for better or worse.

One last comment before I go. The ”--noaltsst” flag is automatically turned ON for global simulations or if the computational domain is localized without inland lakes (USGS & MODIS land use datasets).

Flag: **--ncpus #CPUS**

What I Do: Specify the number of processors to use when running WRF metgrid (horizontal interpolation)

Usage: % ems_prep --ncpus #CPUS [bla, bla, bla]

How You Do It:

Passing the "--ncpus" flag overrides the default value for the number of processors to use when running the WRF metgrid routine. If "--ncpus" is not passed, then the UEMS will determine the number of physical cores on the system and use that value.

Note that just because you want X processors when running metgrid, doesn't mean the UEMS will use that many. The system will only use available physical (non-hyper threaded) cores and will test for domain over-decomposition and reduce the number of CPUs as necessary.

7.6 A few ems_prep examples

As stated above, likely multiple times, there are many command line flags and options available that can be used to override the default behavior of ems_prep. If the minimalist approach is not for you, then feel free to read the flag summaries provided in Section 7.6 and experiment (with the options).

The best way of demonstrating some of the ems_prep functionality is through examples, lots and lots of examples. A few examples are presented below; however, an alphabet's worth of possibilities is waiting for you in Appendix B. For those who can't wait, here are a few tempting morsels of ems_prep wizardry:

7.7.1 The lone requirement - "--dset"

There is only one required flag for ems_prep, **"—dset,"** which specifies the dataset(s) you wish to use for initial and boundary conditions. So, if you are a minimalist type of person, you might run:

% ems_prep --dset gfsp25

Which tells ems_prep to use the GFS 0.25 degree dataset for model initialization and boundary conditions. In the absence of any other flags, ems_prep will use the gfsp25_*gribinfo*.conf file to obtain the default values for the parameters needed to complete its mission. The routine will first check the local "grib/" directory for any existing GFS 0.25 degree files, as identified by the **LOCLFIL** parameter, before searching any external sources for the data. Because no method of acquisition is specified (nfs, ftp, or HTTP), ems_prep attempt each location specified in the **SERVERS** section of the *gribinfo* file when acquiring the data files.

Note: If you place GRIB files in local grib/ directory and ems_prep fails to find them, the most likely reason is that you are not using the proper filenames. The routine expects initialization files named according to the rules defined in the *gribinfo* file. Take a look at the error messages and the **LOCLFIL** parameter in *gribinfo* for more guidance.

Here is another example:

% ems_prep --dset gfsp25:ftp

Translation: The user is again requesting the most recent GFS 0.25 degree forecast GRIB files for the initial and boundary conditions, but this time ems_prep will only use the FTP servers listed in the gfsp25_*gribinfo*.conf (**SERVER-FTP**) file and ignore other sources. Each FTP server is checked for the requested data until all are downloaded to the local machine and renamed to the file naming convention defined by **LOCLFIL**. Note that when not using an anonymous FTP server you must have the user and password information in your ~/.netrc file. If you are reading this and wondering, "What is FTP," then never mind, "junior."

Kicking it up a notch with:

% ems_prep --dset gfsp25:nfs::/data/gfs.tCCz.pgrb2.0p25.fFFF

Translation: The above example is similar to the first example except it accesses the dataset via the copy (cp) command on a local drive, and defines the location (/data) and naming convention for the files (gfs.tCCz.pgrb2.0p25.fFFF). Each *gribinfo* file contains an explanation of the filename placeholders, i.e. [YY]YY, MM, DD, HH, CC, FF, TT. You do not need to include "::/data/gfs.tCCz.pgrb2.0p25.fFFF" on the command line if this information is defined in the gfsp25_*gribinfo*.conf file with a **SERVER-NFS** key. This example is for demonstration purposes only.

7.7.2 Using the "--cycle" flag like you mean business

In the absence of a specified cycle time, ems_prep will use the current UTC on your system and the entries from the **CYCLES** parameter in the *gribinfo* file to determine the most recent run from which data are available. Yes, this routine works like magic. However, if you want to use a specific cycle time, you can use the **"--cycle"** flag:

% ems_prep --dset gfsp25 --cycle 12

In the above example, **"--cycle"** is used to request initialization data from the most recent 12 UTC run of the gfsp25 dataset. It doesn't matter if the 18 UTC run is the most current available, ems_prep doesn't care because you have quested the 12 UTC data and that's exactly what you are going to get no matter how much it hurts.

Here is some more **"--cycle"** fun:

% ems_prep --dset gfsp25 --cycle 12:06:30:06

Translation: The above request is similar to the first example except that the cycle time (12 UTC), initialization forecast hour (06), final forecast hour (30), and boundary condition update frequency (06 hourly) are included as arguments to **"--cycle."** As a consequence, the length of the simulation will be 24 hours (final forecast hour (30) – start forecast hour (6)). If the current dataset is from the 00 UTC cycle, that run (00 UTC) will be ignored in favor of the 12 UTC data

from the previous day. Remember, you have the power - use it wisely.

This time, something really interesting (at least to me):

% ems_prep --dset gfsp25pt:http --cycle CYCLE:06

Translation: Use the UEMS gfsp25 personal tiles beginning with the 06 hour forecast from the most recent run of the GFS. The "**CYCLE**" is a placeholder for the current cycle run, and the "06" is used to override the default initial forecast hour (INITFH) value in gfsp25pt_gribinfo.conf with the 6-hour forecast.

Alternatively, you can skip the "CYCLE" in the example above and use:

% ems_prep --dset nampt:http --cycle :06

Which should have the same result (must be "**:**06" though).

7.7.3 Witness the power of "--length"

The **"--length"** flag allows you to specify the maximum length of your simulation. It defines the length of the period over which initialization files will be obtained and processed for the simulation. It's the *maximum* length because the actual length of the simulation can be reduced with the **"--length"** flag in ems_run, but that discussion will be left for the next chapter.

Here is a simple example of the **"--length"** flag:

% ems_prep --dset gfsp25 --length 48

Translation: Obtain and process 48 hours of forecast files on the gfsp25 dataset from the most recent operational GFS. Override the default simulation length defined in the gfsp25_*gribinfo*.conf file. That example was easy enough.

Here is a more complicated and confusing use of the **"--length"** flag:

% ems_prep --dset gfsp25 –cycle CYCLE:12:36 --length 72

Translation: Without the **"--length 72"** flag, ems_prep will acquire and process 24 hours of forecast files from the most recent GFS run (**CYCLE**) beginning with the 12 hour and going through the 36 hour forecast, in which case the maximum length of the simulation would be 24 hours (36 - 12). However, passing **"--length 72"** overrides the values in the argument from the **"--cycle"** flag and thus ems_prep will obtain and process 72 hours of forecast files, beginning with the 12-hour and going through the 84-hour forecast. Pretty sneaky!

BTW – The above command could be written as:

% ems_prep --dset gfsp25 –cycle :12:36 --length 72

With the same result.

7.7.4 Examples using the "—date" flag

The **"--date"** flag allows you specify the date of the initialization files used in the simulation. This flag overrides the default behavior of using the current system date.

Here is a simple use of the **"--date"** flag:

% ems_prep --dset gfsp25 --cycle 12 --date 20140230

Wasn't that easy? It's similar to previous examples except that ems_prep will acquire and process the GFS dataset files from the 30 February 2014 12 UTC cycle run. If the files are not already located in the local "grib/" directory, then ems_prep will scour the world looking for your data, or at least those sources specified in the gfsp25_*gribinfo*.conf file. You might use this example for running simulations of historical cases, or maybe not.

Note: If you do not include the "--cycle" flag along with "--date," ems_prep will default to the last available cycle time for that dataset on that date, which in this case would be from the 18 UTC run.

7.7.5 Working with multiple initialization datasets

The ems_prep routine can handle the use of different datasets for initial and boundary conditions. This functionality is often used when extending a simulation beyond the file times available from the initialization dataset. For example:

% ems_prep --dset hrrr%gfsp25 --length 24

Translation: Use the most recent HRRR 00-hour forecast as the 00-hour (initialization) time for the simulation and the 0.25 degree GFS as the lateral boundary conditions out to 24-hours. The GFS dataset is being used for the boundary conditions because, at the time of this writing, the HRRR runs through 18-hours, so a 24-hour simulation using only the HRRR is not possible. The routine will attempt to acquire the 00-hour forecast from the HRRR dataset and then the 01 to 24-hour forecast files from the GFS run.

Important: *The* ***"--length"*** *flag must be included when using a different dataset for initial and boundary conditions.*

So, what happens if the current cycle time for each dataset is not the same? For example, what if currently available HRRR dataset is from 11 UTC, but the GFS is from 06 UTC?

"Well, what's the great and almighty UEMS going to do for me then?"

Never fear! The ems_prep routine is all-wise! If the cycle times of the initialization and boundary condition datasets do not coincide, the routine will adjust either the cycle time or the initial forecast hour of the boundary condition dataset to meet the requested simulation length. In the above situation, the HRRR 00-hour forecast from 11 UTC is used for 00-hour in the simulation, and the 06 through 23-hour forecasts from the 06 UTC GFS is used for the boundary conditions.

Don't bother to do the math; it's much too complicated.

How about this mind blower, again let's assume its 11 UTC:

% ems_prep --dset hrrr%gfsp25 --length 24 --cycle CYCLE:12

Translation: Use the 11 UTC HRRR 12-hour forecast for your 00-hour time (23 UTC) and the 18 (00 UTC) through 41-hour forecast from the 06 UTC GFS run.

Note that the above command is the same as this one:

% ems_prep --dset gfsp25%gfs --length 24 --cycle :12

In which the "**CYCLE**" is not included. That's not magic, that's Voodoo. Besides, the ems_prep doesn't care because it doesn't have feelings, but I'm working on that limitation.

Finally, you may also request the initial and boundary condition datasets be obtained from different sources:

% ems_prep --dset nampt:http:ems3%gfspt:http:ems1 --cycle CYCLE:84 --length 48 --sfc rtgsst

Translation: Use the 12km NAM personal tiles (nampt) available from EMS3 HTTP server beginning with an 84-hour forecast from the most recent run **(--cycle CYCLE:84**) for the initial conditions. Use the GFS 0.5 degree personal tiles (gfspt) from the EMS1 data server. Total run length will be 48 hours (**--length 48**). Also, get the 8.3km high-resolution SST dataset (**--sfc rtgsst**). Note that ems_prep will download the appropriate GFS files from the most recent GFS run. BTW – Don't attempt the above command. It probably won't work for you.

7.7.6 Fail-over LSM datasets

A major difference between the "--sfc" and "--lsm" flags is that while the "--sfc" datasets are not required for model initialization, ems_prep will terminate if any of the "--lsm" datasets are missing. If your first choice is not available, fail-over datasets can be specified to reduce the likelihood of your forecast coming to an untimely demise. In this application, the pipe (|) character is used to separate a succession of datasets in decreasing order of desirability (from left to right). If a dataset in this list is not available, ems_prep will proceed to the next. Once a dataset from the list is acquired, the search ends and ems_prep continues with processing of the files; however, if none of the listed fail-over options are found, the entire process terminates, and your forecast dreams are dashed once again.

The use of the "|" separator should not be confused with the comma-separated LSM datasets. The comma is used to specify multiple datasets that are mandatory for model initialization, while a "|" is used to specify alternatives to each mandatory data type. Additionally, when using "|" separator with the "--lsm" flag, the ***entire string must be in quotations***. This requirement is due to how the Linux command line interpreter handles the pipe symbol.

Here is a generalized example:

% ems_prep –dset gfsp25 --lsm "Alsm1|Alsm2|Alsm3,Blsm1|Blsm2"

Assuming that the datasets listed above mean anything, this example specifies that two LSM datasets are required for model initialization; one from group "A," Alsm1|Alsm2|Alsm3, and one from group "B," Blsm1|Blsm2. The ems_prep routine will first attempt to get Alsm1 from group A. Should that dataset not be available, it will try to get the Alsm2 dataset, and so on. If successful, the routine will next attempt to obtain Blsm1 from the second group with Blsm2 serving as a fail-over. Should ems_prep fail to locate a dataset from any group, model initialization is over for you.

7.7.7 "What do we want? CESM/CMIPS5, ERAx, CFSR, NARR, and NNRP data!"

In an effort to further simplify your life; the SOO STRC and UEMS have combined their powerful resources to provide a near effortless way of running historical case studies using the Climate Forecast System Reanalysis (CFSR), North American Regional Reanalysis (NARR), Global Reanalysis I and II (NNRP), and ECMWF climate reanalysis datasets. Why all the hullaballoo? Try using these datasets without the "UEMS Advantage©" and then you'll be thanking your Uncle UEMS for the effort - big time!

The UEMS servers now provide most of these datasets on-line (ECMWF excluded) for use with the system. The CFSR global dataset includes the period from January 1979 through current, the NARR dataset covers the period from 1979 to 2014, and the global NNRP dataset runs from 1948 to 2004 (Reanalysis I from 1948-1979 and Reanalysis II from 1980-2004). More dates will be available as disk capacity is increased, so don't take the date ranges to be accurate.

The CFSR, NARR, and NNRP datasets:

To use these datasets, simply request "cfsr", "cfsrpt", "narrpt" or "nnrp" with the **"--dset"** flag:

% ems_prep --dset cfsr --date 19840327 --cycle 12 --length 36

Or

% ems_prep --dset cfsrpt --date 19871214 --cycle 12 --length 36

Or

% ems_prep --dset narrpt --date 19931214 --cycle 12 --length 24

Or

% ems_prep --dset nnrp --date 19630311 --cycle 12 --length 48

Caveat: If you are running a case that occurred more than 50 years from the current date, you may see the following error:

% ems_prep --dset nnrp --date 19481124 --cycle 12 --length 24

Day too big - 28817 > 24853
Cannot handle date (00, 00, 00, 24, 10, 2048) at /bla/bla/Others.pm line ...

A “design flaw” in Perl, not in the UEMS, causes the error. To fix this problem, you must edit the “Time/Local.pm” module in your local Perl libraries and decrease the “50” value to a suitable value such as 30.

The ERAx (E.g., ERA20, ERA5, ERA-i) datasets:

The UEMS also supports the ECMWF "ERAx" datasets; however, using these data requires that you have an account on the NCAR research data archive (RDA) system to download the files for use with the UEMS. The datasets are in GRIB 1 format with the fields needed to initialize a simulation broken into three different files, which is a bummer, especially if you are not familiar with ECMWF GRIB1 field IDs.

Fortunately, because the UEMS Overlord likes you, individual scripts are provided to greatly simplify the task of downloading and processing these data for simulation initialization. So when you are ready to take the “ERAx leap,” read the “How-To (let the UEMS do everything for you)” guidance provided in uems/util/extras**/ERA20|ERA5|ERA-i**/ for more details. You’ll be glad you did.

Using the CESM/CMIPS5, and similar WRF intermediate datasets:

Additionally, the UEMS supports the NCAR CESM Global Bias-Corrected CMIP5 (CESM/CMIP5; available from NCAR/RDA) dataset. The problem is that it’s only available in WRF intermediate format, whereas all previously discussed datasets are assumed to be in GRIB. The ems_prep routine produces files in this format as an intermediate step (hence the name) between GRIB and the netCDF files used to initialize a simulation, but it was not originally designed to ingest them directly.

All welcome the “--wrfint" flag!

```
% ems_prep --wrfint [matching string] [the other flags]
```

The “--wrfint" flag allows users to skip the GRIB file processing and use WRF intermediate files, because you want action now and don't have time for "GRIB play." If this statement describes how you roll, then the "--wrfint" flag delivers the modeling lifestyle you deserve.

By passing the "--wrfint" flag, the UEMS will use the accompanying string to match files in the <domain>/wpsprd/ directory. For example, if you want to use the NCAR CESM Global Bias-Corrected CMIP5 data set in WRF intermediate format, download the files from the NCAR research data archive:

https://rda.ucar.edu/datasets/ds316.1/index.html (you need an account)

Then place them (CCSM4_CMIP5_MOAR_BC_RCP85:YYYY-MM-DD_HH) in the wpsprd/ directory and unleash the power of "--wrfint."

```
% ems_prep --wrfint CCSM4_CMIP5_MOAR_BC_RCP85 [other flags & stuff]
```

Wherein the ems_prep routine will select all files matching "CCSM4_CMIP5_MOAR_BC_RCP85" for use in the simulation. These files will be used to determine the simulation start and stop times, as well as the lateral boundary condition update frequency.

A few additional flags are implicit (i.e, thay are automatically included) when using "--wrfint," specifically,

--analysis	Analysis dataset flag
--noscour	Do no delete existing files prior to run
--nointdel	Do not delete intermediate files after horizontal interpolation
--noaerosol	Do not include the aerosol climatology dataset

Finally, only a single file is needed for global simulations, but you must also pass the "--length" flag unless you want simulate how the world ends.

Chapter 8: Mastering ems_run.pl

Chapter Contents:

8.1 Beginning the ems_run.pl experience

The ems_run.pl routine handles many minor details that typically encumber users prior to running a simulation. The routine is responsible for making sure that the configuration is valid, creating the initial and boundary conditions, and then running the simulation. It also manages and controls the pre-run MPI configuration if you are distributing the job across multiple computers and processors. These are just a few of the tasks, and the UEMS understands that you have better things to do with your time. Making your life easier so you can slack off, that's the primary responsibility of ems_run.pl.

When running a simulation, the ems_run.pl routine runs the WRF "real_arw.exe" program, and then the ARW core ("wrfm_arw.exe"). The "real_arw.exe" program, when not longing for a better name, handles the vertical interpolation of the initialization data output from ems_prep to the specified model levels, followed by the creation of the initial and lateral boundary conditions. After successfully creating the initial and boundary condition files, the routine will set up and run the simulation. During execution, output files are placed in the top level of the domain directory. Once the simulation has completed, data files are moved into the "wrfprd/" subdirectory, log files are placed under "logs/," and ancillary files are deleted.

As with all the run-time scripts, ems_run is run from the top level of a simulation domain directory. There you will find a link from ems_run to the actual routine, ems_run.pl, which resides under "uems/strc/Ubin/." The "uems/strc/Urun/" directory contains all the libraries used by ems_run.pl. You never run ems_run.pl directly. If the ems_run link gets deleted, it can be recreated by running "ems_clean --level 0." For the sake of clarity, "ems_run" is used throughout this chapter to refer to the link that exists in each domain directory.

***Important**: You must run ems_prep prior to running ems_run, but I probably didn't have to tell you this information.*

8.2 Meet the ems_run configuration files

There are many configurable parameters and options used with ems_run to create the initial conditions and run the model, all of which are described in quasi-organized configuration files. When you create a new domain using the Domain Wizard or ems_domain (Chapter 5), default configuration files are copied to the <domain>/conf/ems_run directory. *These are the files that you should edit*

when setting up a simulation.

Novice users should not feel intimidated by all the parameter options presented in these files. The UEMS comes pre-configured to run most simulations, so there are only a few changes that a user may consider. In addition, the parameters and options presented are relatively well documented, which should help ease a user's "What/How do I do this?"-fueled anxiety. Any non-WRF spawned anxieties must be handled professionally, but feel free to include the UEMS as part of your therapy.

Here is a brief description of the ems_run configuration files, in a very loose order of importance:

run_wrfout.conf

The run_wrfout.conf file handles the data output frequency from the primary and any nested domains. It also specifies the file types to output and a few additional items of interest. Most users typically modify the output file frequency of each domain as needed. The output format is netCDF, which is designed for use with the ems_post routine (Chapter 9).

run_physics_<scheme>.conf

The run_physics_<scheme>.conf configuration files handle the physics configuration for the primary and any nested domains used in your simulations. There are individual files for the various scheme types, each containing a self-serving description. Even the most WRF-savvy users may learn something new by reviewing these files. Note except for the cumulus (CU_PHYSICS) and planetary boundary layer (BL_PBL_PHYSICS) parameterization schemes, all domains included in a simulation are required to use the same settings. This "executive decision" by the UEMS Emperor was made to reduce potential problems across domain boundaries. Even if you attempt to set the value for a nested domain to something other than that of its parent, the UEMS will automatically reset the nested. So don't try any funny stuff, OK?

run_ncpus.conf

The run_ncpus.conf configuration file manages the decomposition of the model domain and specifies the nodes, machines, and number of CPUs to use when running a simulation. When the UEMS is installed, an attempt is made to determine the number of physical CPUs (SOCKETS) and cores per CPU (CORES) on the local system. Those values are used to configure the environment variable files that specify the default values for the number of processors included in a simulation. If you want to override these values, or if you are running on a multi-node system, then you should take a look at this file.

*It is recommended that you check the configuration of **REAL_NODECPUS** and **WRFM_NODECPUS** in run_ncpus.conf prior to running ems_run.*

run_timestep.conf

The run_timestep.conf configuration file specifies how the large timestep for a simulation is determined. You can select from a number of methods for calculating a timestep or define a timestep value yourself. The default method will compute a time step, based on WRF recommendations, from the smallest grid spacing within a primary computational domain. This approach should be sufficient for most applications.

run_levels.conf

The run_levels.conf configuration file defines the number and/or distribution of native model levels within the computational domain. The default number of levels is 45, which should be sufficient for many applications. The UEMS dictator is not without the capacity for love and guidance. Consider that the number of levels should be proportional to the amount of baroclinicity in the model-simulated atmosphere. Thus, the number of levels may need to be increased for cool season simulations. Conversely, the number of levels may be decreased for warm season and tropical simulations.

As a rule, the UEMS gatekeepers do not appreciate a reduction in resolution of any kind.

run_nests.conf

The run_nests.conf configuration file specifies the feedback (one- or two-way) and smoothing methods used when running a nested simulation. The default value is for 1-way nesting but nothing is stopping you from some 2-way excitement; however, if 2-way nesting is failing for some unknown reason, consider switching back to 1-way nesting before complaining to the lone, under-appreciated UEMS "Support Minion."

run_dynamics.conf

The run_dynamics.conf configuration file defines the configuration for the dynamics used in the model. For the most part, the default settings are sufficient for nearly all applications. The exception would be if you were running a simulation with less than 1km grid spacing. If super high-resolution simulations are in your modeling plans, then make sure you know what you are doing before pushing buttons.

run_dfi.conf

The run_dfi.conf configuration file is available for those users wishing to run the WRF digital filter initialization (DFI). The DFI option allows for the reduction in model spin-up time during the early stages of integration due to a mass/momentum imbalance in the initial conditions.

Note that the use of DFI can increase the computational time of your model run, so use this option wisely. Also, testing has been limited so there are few promises as to whether this option will work as advertised.

run_nudging.conf

The run_nudging.conf configuration file is for users wishing to include 3D analysis or spectral nudging as part of a simulation. Nudging can be useful for some types of retrospective simulations and regional climatology studies by reducing the amount of forecast "drift," or error, during model integration.

For studies that involve the dissection of model forcing, it is recommended that nudging NOT be applied to any domains directly involved in the analysis. For example, if you are running a simulation with two nested sub-domains and plan on using only the innermost nest for the research, then turn OFF nudging for that domain. This warning is because the inclusion of nudging introduces additional (non-physical) forcing into the model atmosphere that you would have to explain, and you don't want to go there.

More information on 3D Analysis Nudging is available in the Appendix.

run_namelist.conf

The purpose of the run_namelist.conf file is to allow users to include an alternate version of the namelist file (namelist.input) to that created from the configuration files. Users also have the option to override individual WRF namelist file parameters that are not accessible through the UEMS configuration.

run_wrfreal.conf

In this file you will find configuration options for the vertical interpolation of the (WPS) output data by the WRF REAL program. The REAL routine manages the vertical interpolation to native model surfaces of 3D variables to be used in model integration and then creates the initial and boundary condition files for the primary and any nested domains.

Most of the configurable parameters contained in this file define the methods and rules for the interpolation. Most users don't need to make any changes to this file unless they are really bored.

run_suite.conf

Tired of jumping from configuration file to configuration file each time you want to change a bunch of physics settings for your simulation? Sure, we all are! Then the run_suite.conf file was created just for you.

The run_suite.conf file contains the UEMS variant of the WRF "PHYSICS SUITE," which allows you to specify a set of physics configuration settings with a single parameter. This option differs slightly from the official WRF release version in that besides the suite options offered by WRF, the UEMS allows you to define your personal physics suite. Pretty "sweet," hey.

run_auxhist1.conf

The run_auxhist1.conf configuration file turns ON the output of additional surface fields from WRF simulations in netCDF format. When the AUXHIST1_INTERVAL parameter is turned on, an additional set of near surface fields (**auxhist1**_*) will be output during a simulation. The advantage of this option is that the files may be output at a higher frequency than the primary **wrfout** files. By default, output to these files is turned off because most fields are also contained in the primary **wrfout** files. Also, the contents of these files are likely to change with each UEMS release and the developer does not have to let you know in advance. Just warning you.

The **static/emsupp_auxcntrl.parm** file manages the output to the "auxhist1" files. If you plan on including this dataset with your simulation output, it's recommended that you peruse the valuable information contained within. Fields found in the "**auxhist1**" files include, but not limited to:

Shelter Level (2 and 10 meter) Fields:

a.	2 Meter Temperature	K
b.	2 Meter Specific Humidity (ARW only)	kg kg^{-1}
c.	2 Meter Relative Humidity (NMM only)	%
d.	10 Meter U-Wind	m s^{-1}
e.	10 Meter V-Wind	m s^{-1}

Accumulated and Total Precipitation Fields:

f.	Accumulated Total Precipitation	mm
g.	Accumulated Freezing Rain	mm
h.	Accumulated Snow & Ice (Liquid Equiv.)	mm
i.	Accumulated Snowfall (Liquid Equiv.)	mm
j.	Accumulated Graupel (Liquid Equiv.)	mm
k.	Accumulated Hail (Liquid Equiv.)	mm
l.	Snow Depth on Ground	meters
m.	Snow (Liquid Equiv.) on Ground	mm

Fabulous Fun-Filled Fields* (and alliteration too):

n.	Maximum 10 Meter Wind Speed	m s^{-1}
o.	Maximum 10 Meter Wind Gust Potential	m s^{-1}
p.	Maximum Updraft	m s^{-1}
q.	Maximum Downdraft	m s^{-1}
r.	Maximum 1000m Reflectivity	dbZ
s.	Maximum 3000m Reflectivity	dbZ
t.	Instantaneous Updraft Helicity (ARW only)	m^2 s^{-2}
u.	Maximum Updraft Helicity	m^2 s^{-2}
v.	Maximum Column Integrated Graupel	kg m^{-2}

* "Maximum" fields are computed between output times

Afterthought: A review of the auxhist1 file contents reveals a large number of fields not included in the above list, but the author is too lazy to include them at this time. The UEMS Overlord suggests that you harden-up and inspect the files yourself.

See Chapter 9 for details on processing the data files.

run_hailcast.conf

The run_hailcast.conf configuration file turns ON the output of available HAILCAST fields from WRF simulations in netCDF format. HAILCAST is a one-dimensional, physically based hail forecasting model. It is a revised version of the coupled cloud and hail model developed by Poolman (1992) and improved upon by Brimelow et al. (2002) and Jewell and Brimelow (2009).

HAILCAST is designed to be embedded within a convective-allowing forecast model. It will produce forecasts of hail size, and if desired, hail density and terminal velocity. Key features include:

a. Hailstone growth through liquid water accretion, ice particle collection, condensation, and sublimation
b. Explicit calculation of hailstone temperature
c. Regimes of wet and dry growth
d. Use of multiple initial embryo sizes and insertion temperatures
e. Parameterized horizontal motion of the hailstone through the convective updraft
f. Variable hail density including rime soaking
g. Temperature-dependent ice collection efficiency
h. Liquid water shedding
i. Enhanced melting during collision with >0°C water

The **static/emsupp_hailcntrl.parm** file manages the output to the "hailcast" files. If you plan on including this dataset with your simulation output, it's recommended that you peruse the valuable information contained within. Fields found in the "hailcast" files include, and limited to:

Hailcast Fields:

a.	Hail Diameter, 1st rank order	mm
b.	Hail Diameter, 2nd rank order	mm
c.	Hail Diameter, 3rd rank order	mm
d.	Hail Diameter, 4th rank order	mm
e.	Hail Diameter, 5th rank order	mm
f.	Period Maximum Hail Diameter	mm
g.	Period Mean Hail Diameter	mm
h.	Period Standard Deviation of Hail Diameter	mm
i.	Updraft Mask	-
j.	Updraft Duration	s

Reflectivity Fields:

a.	Simulated Composite Reflectivity	dBZ
b.	Reflectivity on Hybrid (Native Model) Surfaces	dBZ

Lightning Field:

a. Lightning Potential Index non-dim

8.3 Managing output from the model

All data files are placed in the "wrfprd/" directory following completion of a simulation. These netCDF files contain a set of variables on surface, shelter, and native model levels. The UEMS package provides a number of utilities that may be used to interrogate the raw netCDF files, including rdwrfnc, ncview, and ncdump. Users wishing to view their data on isobaric or some other surface must process the output with additional software, which is why the **ems_post** routine is provided.

For more information on ems_post, see Chapter 9.

8.4 Hyper-threading = bad, You = awesome!

This diversion was initially presented in Chapter 3, but this is as good a place as any to cover this topic again, because with the UEMS, excess is always a good thing. Regarding the use of hyper-threading, *The UEMS Council of Youngers does not recommend using the additional threads available when hyper-threading is turned on in the BIOS.* The reason for this statement is that you are much more likely to see degradation in performance rather than improvement. Yes, there are a limited number of conditions under which you might see some improvement depending upon your processors, the amount of cache, your domain configuration and the decomposition; however, you are far more likely to see a degradation in performance.

Keep in mind that with hyper-threading, only the number of threads available on a processor is increased. *You do not increase the computational resources*, such as the total number of processors available. If the number of threads created exceeds the number of available processors, then multiple threads will be assigned to a single processor. Consequently, each thread will have to share the resources allocated to that processor, such as memory and cache. This configuration is only beneficial if the total amount of resources needed by the multiple threads is less than that available on the processor and there is no sharing of resources between threads. It is the sharing of resources, specifically cache, that increases the overhead, and degrades performance.

As an example, if you have a dual CPU, quad-core system that has a total of 8 processors available and you specify that all 8 processors be included in a simulation, the WRF model will decompose the domain into 8 patches, each of which is assigned by the kernel to a processor as a single thread. Since there is a single thread assigned to each processor, there is no sharing of available core resources. The model/system handles all the communication between the patches.

When hyper-threading is turned on and you specify 16 processors be used, the model will decompose your domain into 16 patches and the kernel will create 16 threads. The problem is that since you only have 8 cores, there will be 2 threads assigned to each core. Remember that the threads will likely have to share resources, and the overhead involved in sharing between threads leads to a degradation in performance.

8.5 Running ems_run from the command line

The ems_run routine can be run directly from the command line to test real-time forecasting systems and execute case study simulations:

% ems_run [--domains 2,...,N] [additional optional flags]

Where [**additional optional flags**] may include the host of available options described in Section 8.6 below.

Important: If you are running a simulation on a single domain, then you do not need to include any flags when starting ems_run. If you want to include nested domains in your simulation, then you must pass the **"--domains"** flag; otherwise, the run will consist of only the primary domain (Domain 1).

8.6 The ems_run command line options

Various flags and arguments can be passed to ems_run. These options serve to override existing settings in the configuration files. If you are using ems_run for real-time modeling, then it is recommended that you modify the values in the appropriate configuration files. You have been warned.

One of the most useful options is "--help," which should be evident in its purpose as it provides a simple listing of the options available to the user. Nonetheless, a somewhat more informative description of the available options is provided below.

Flag: --[no]scour

What I Do: Controls the amount of directory cleaning done prior to the start of ems_run

Usage: % ems_run --[no]scour [more important stuff]

How You Do It:

Passing the "--[no]scour" flag overrides the default ems_run penchant for cleaning up the run-time directory before starting a new simulation. Without this flag, the default level is the same as running "ems_clean --level 2", which deletes all non-essential files in the run-time directory.

Passing "--noscour" is the same as running "ems_clean --level 0", which completely deletes old log files and recreates the symbolic links to the run-time scripts.

Flag: --[no]dfi

What I Do: Turn ON [OFF] Digital Filter Initialization (DFI) for the simulation

Usage: % ems_run --dfi [dfi option[:dfi filter]] [again with the other stuff]

How You Do It:

Passing of the "--dfi" flag is used to override the DFI_OPT and DFI_NFILTER parameter settings in the run_dfi.conf file. Passing "--dfi 0" turns OFF all DFI processing unless DFI is already turned off (DFI_OPT = 0), in which case it does nothing.

An argument may be included with the "--dfi" flag to specify the DFI option number (1,2, or 3), and filter used (0, 1, 2, 3, 4, 5, 6, 7, or 8), separated by a colon. For example:

```
% ems_run  --dfi  3:7
```

Which instructs ems_run to use Twice DFI (3) with the Dolph (7) filter.

Current DFI options include:

0 - No DFI will be used
1 - Digital filter launch (DFL)
2 - Diabatic DFI (DDFI)
3 - Twice DFI (Default)

The current DFI filters include:

0	Uniform
1	Lanczos
2	Hamming
3	Blackman
4	Kaiser
5	Potter
6	Dolph window
7	Dolph
8	Recursive high-order

If "--dfi" is passed without an argument, or with a partial argument, the default values are used.

Option 3 (twice DFI) is recommended until a sextuple DFI is available.

Flag: **--domains**

What I Do: Provide control over which domains to be included in the simulation

Usage: % ems_run --domains domain1[:FCST LENGTH],...,domainN[:FCST LENGTH]

How You Do It:

Passing the "--domains" flag specifies the domain(s) to include in your simulation. All the domain(s) must be included when running ems_prep prior to ems_run; otherwise, an error is generated and you will be left with a sinking feeling and wet shoes.

By default, ems_run will only execute the primary domain (Domain 1) unless the "--domains" flag is passed with an argument. Those are the rules.

The "--domains" flag can also be used to refine the length of nested simulations, which is done by including a length of time followed by the units; d (days), h (hours), m (minutes), or s (seconds). A colon (:) is used to separate the domain number from the length. For example:

% ems_run --domains domain#:length[units],...

Where,

Domain# The domain number you wish to include
Length The length of the nested domain simulation in specified units

Note that:

1. **Domain#** is mandatory, but **Length** is optional
2. The **Length** value is preceded by a colon (:) and must include the units (**d|h|m|s**)
3. Multiple domains are separated by a comma (,)
4. In the absence of a length value, the simulation will default to the stop time of the parent domain simulation.
5. If the **Length** value provided extends the stop date/time of the child simulation beyond that of its parent, ems_run will reduce the length of the nested simulation so that it coincides with that of the parent.

Here is an example:

% ems_run **--domains 2:3h,4:6h**

Translation: Include domains 2 and 4 in the simulation and run for 2 and 6 hours respectively. If domain 4 is the child of 2, then the 6 (hours) will be ignored and domain 4 will be run for 3 hours unless the parent of domain 2 (domain 1) is terminated prior to the end of the 3 hour forecast, in which case all three domains will end at the same time. If domain 4 is the child of domain 3, which is not listed, then domain 3 *will be automatically included* with a start time and length of its parent domain.

Trust your Uncle UEMS; there are a lot of checks to make sure the start and stop times of the nested simulations are correct!

Flag: **--interp**

What I Do: Interpolate nested static fields from the parent domain (ARW nesting only)

Usage: % ems_run **--interp**

How You Do It:

Interpolate nested domain static surface fields (terrain, land-sea mask, etc.) from the parent domain. When you crafted your computational domain, separate terrestrial datasets were created for the primary and each sub-domain, the resolution of which was commensurate with its grid spacing. These tailored fields are used for each sub-grid when running a nested simulation.

Passing the "--interp" flag tells the UEMS not to use these higher (generally) resolution fields during a simulation, but rather, interpolate the data from the lower resolution (again, generally) primary domain terrestrial dataset.

Typically, you do not want to pass the "--interp" flag unless you are investigating the influence of the higher resolution terrestrial fields on a simulation. You know, like research stuff.

Flag: **--length**

What I Do: Specify the simulation length for the primary domain

Usage: % ems_run **--length** TIME<d|h|m|s>

How You Do It:

Passing the **"--length"** option overrides the default length of the model simulation for the primary domain (Domain 1). The default length of the simulation was established when ems_prep was run to process the initialization files. In the absence of the **"--length"** option flag, ems_run will use the period covered by the initialization data set to determine the run length.

The **"--length"** option can only be used to shorten the length of the simulation for the primary domain. Values that exceed the default run length are ignored, as will you.

Important: Reducing the length of the primary simulation may have an undesired effect on nested domains included in your simulation. For example, if an original (default) length of a primary domain simulation was to be 24 hours with a 06 hour nested simulation scheduled to start 12 hours into the primary simulation, and the **"--length 12h"** was passed to ems_run, the nested simulation would automatically be turned off since its start time is the same as the end time of the parent domain. Got that? Yes, I know, run-on sentences are tough.

Flag: **--levels**

What I Do: Specify the number of vertical levels to include in the simulation

Usage: % ems_run **--levels** #levels

How You Do It:

The **"--levels"** flag serves to override the LEVELS parameter in run_levels.conf. The difference is that the **"--levels"** flag only accepts an integer number of levels and not a defined vertical distribution. All the domains included in the simulation will use the same number of vertical levels and distribution. In a UEMS world, every domain is almost equal.

If you choose to specify the number of vertical levels with a single integer value, e.g. **"--levels 61",** the WRF real program will use this value to define a set of well-spaced model levels. As possibly stated above, the default number of levels is 45, which should be sufficient for most applications. However, as a rough guide, the number of levels should be proportional to the amount of baroclinicity in the model-simulated atmosphere. Thus, the number of levels may need to be increased during the cool season and may be decreased during the warm season. Additionally, it is recommended that the number of levels increases with smaller horizontal grid spacing, but this is just a suggestion.

Finally, remember that increasing the number of levels will proportionally increase the amount of time required to run your simulation.

Flag: **--noreal**

What I Do: Do not run the REAL program prior to starting forecast model

Usage: % ems_run **--noreal**

How You Do It:

Pass the **"--noreal"** option if you do not want to run the WRF real program prior to executing a simulation. The real program is used to create the initial and boundary condition files used for the run, so you would only pass **"--noreal"** if you already have **wrfbdy**_ and **wrfinput**_ files lying around in the run directory.

This option is of limited use unless you are debugging something.

Flag: **--nowrf**

What I Do: Terminate ems_run prior to running the model. Just run WRF REAL program

Usage: % ems_run **--nowrf**

How You Do It:

Pass the "--nowrf" option if you do not want to run the simulation after creation of the initial and boundary condition files. This flag is used for testing and debugging purposes only.

Flag: **--nudge**

What I Do: Turn ON 3D analysis or spectral nudging during a simulation

Usage:

% ems_run **--nudge [1|2]** bla, bla, bla and more bla

How You Do It:

Passing the "--nudge" flag turns ON 3D analysis or spectral nudging during a simulation. When "--nudge" is passed, configuration information is read from the run_nudging.conf file, so please, do not play with the "--nudge" flag until you have read and understood (read twice) the available configuration options.

The lone argument to "--nudge" [1|2] serves to override the GRID_FDDA parameter in the configuration file, which specifies whether to do analysis (1) or spectral (2) nudging. Not including 1 or 2 will cause the UEMS to default to the value of GRID_FDDA in run_nudging.conf.

To summarize:

% ems_run --nudge -> Nudging ON & use value of GRID_FDDA
% ems_run --nudge 1 -> Analysis nudging ON & ignore value of GRID_FDDA
% ems_run --nudge 0 -> Spectral nudging ON & ignore value of GRID_FDDA

Super Important: If you want to do any nudging during a simulation, you must also pass the --nudge flag when running ems_prep. This flag requests that the required nudging fields created from the initialization datasets.

Note: The "period" of nudging begins with model integration of domain 1 (T0)

More information on 3D analysis nudging in the WRF is provided in Appendix G.

Flag: **--rundir**

What I Do: Set the simulation run-time directory if not current (.) directory

Usage: % ems_run **--rundir** <simulation directory>

How You Do It:

Pass the "--rundir" flag to specify the run-time domain use for a simulation. The domain directory must exist or ems_run will terminate, and you don't want that to happen now do you?

Note: This flag should not be passed by the user, but rather, is used by **ems_autorun** internally. Nobody here at UEMS world headquarters uses it, so neither should you.

Just say no to "—rundir."

Flags: **--sdate**

What I Do: Set the start date for the model simulation (Primary domain)

Usage: % ems_run **--sdate** [YY]YYMMDD

How You Do It:

Passing the "--sdate" flag allows users to begin a simulation after the date/time assigned when running ems_prep. The date and time specified by YYYYMMDDHH must correspond to an initialization data file residing in the "wpsprd" directory; otherwise, ems_run will give you the "Stink Eye" and quit.

For example, if ems_prep is used to initialize a 24-hour simulation beginning at 00 UTC 30 February 2019, including a 3-hourly boundary condition update frequency, passing "--sdate 2019023006" will result in ems_run to use the initialization file from 06 UTC as the initial conditions for the simulation. When "--sdate" is passed, the length of the simulation will be adjusted accordingly.

Warning: passing "--sdate" along with "--domains" may cause problems unless the start times for any sub-domains are specified to be later than that of the primary domain when running ems_prep. You probably don't want to go there.

Important: passing "--sdate" along with "--domains" may cause problems unless the start times for any sub-domains are specified to be later than that of the primary domain when running ems_prep. You probably don't want to go there.

Chapter 9: Getting the most out of ems_post.pl

Chapter Contents:

9.1 Beginning the ems_post.pl experience

The ems_post.pl routine manages the processing of data files output from simulations by utilizing a highly modified version of the Unified Post Processor (UPP). Following completion of a simulation, data files are moved to the "wrfprd/" directory. These files are in netCDF format and contain fields on native model levels, which may not meet the demanding needs of UEMS users. Running ems_post allows users to process these data into a variety of secondary formats and then export the information to other locations. The processing options include, but are not limited to, converting the netCDF to GRIB 2, writing GEMPAK and GrADS files, and creating BUFR, GEMPAK, and BUFKIT sounding files. Processed files are then placed in the "emsprd/" directory, with log files in the "logs/" directory. Additionally, users have the option of exporting files to other locations and remote systems via secure copy (SCP), file transfer protocol (FTP) secure file transfer protocol (SFTP), or a simple copy command (CP).

If no flags or options are passed to ems_post, the default behavior is to process all available primary output files for domain #1 in accordance with the parameter settings in the post_uems.conf file. If you are looking for something different, read the description for each flag and option below or pass "--help <topic>" to ems_post, and then let your imagination run wild.

As with all the run-time scripts, ems_post is run from the top level of a domain directory. There you will find a link from ems_post to uems/strc/Ubin/ems_post.pl. Additionally, the uems/strc/Upost/ directory contains the libraries used by ems_post.pl. You never run ems_post.pl directly. If the ems_post link gets deleted, it can be recreated by running "ems_clean --level 0." For the sake of clarity, "ems_post" is used throughout this chapter to refer to the link that exists in each domain directory.

9.2 How do I post thee? Let me count the ways!

What can ems_post do for you? Well, the ems_post routine can:

1. Processes both primary (wrfout) and auxiliary (auxhist1) output files from global and nested limited area simulations into a variety of secondary data formats,
2. Run concurrent with a simulation (See autopost),
3. Process fields on as many as 80 pressure and potential temperature levels, 10 potential vorticity surfaces, and 80 Above Ground Levels (AGLs), (Chapter 9.5),
4. Manage data files with up to a 1-minute temporal frequency,
5. Process multiple domains from a nested simulation independently.
6. Produce information in multiple formats, including GRIB 2, BUFR, GEMPAK grid files, GEMPAK sounding files, BUFKIT, GrADS, and netCDF,
7. Create images for display on the web,
8. Create BUFR and BUFKIT files with up to 1-minute temporal frequency,
9. Export files to remote systems via FTP, COPY, SCP, RSYNC, and RCP,
10. Allow users to fine-tune the frequency of the files being processed and exported for each format type.

So, that's what your ems_post.pl can do for you!

9.3 The ems_post configuration files

There are many configurable parameters and options used by ems_post to process output from a simulation, all of which are described in quasi-organized configuration files. When you create a new domain using the Domain Wizard or ems_domain (Chapter 5), default configuration files are copied to the <domain>/conf/ems_post directory. *These are the files that you should edit when specifying your post-processing requirements.*

The unenlightened within the UEMS community should not feel intimidated by all the post-processing possibilities presented in these files. These parameters and options are relatively well-documented, which should help ease a user's "Who wrote this POS?!"-fueled anxiety. Any non-WRF spawned fears must be handled professionally, but feel free to include the UEMS as part of your therapy, which will hopefully involves washing your mouth and mind with soap.

If you are processing output as part of ems_autorun (See Chapter 10), it is imperative that you edit the configuration files before running the autorun routine. If you are executing ems_post from the command line, then you have options and flags available that override the configuration file values. Nonetheless, it is recommended that you review your local configuration to make sure it smells and tastes just the way you like it before moving ahead with post-processing.

Below is a brief description of the various ems_post configuration files, in no particular order of importance, except for the first one.

A) post_uems.conf

The post_uems.conf configuration file governs the post-processing to be performed. This file is of primary importance when executing real-time forecasts using the ems_autorun routine since you do not have access to the command-line flags. Here, the user specifies the processed data formats, which in turn, determines whether any of the other configuration files are read. For example, if GEMPAK = Yes (or pass the "--gempak" flag), the post_gempak.conf file is read. Therefore, before using ems_autorun, make sure to make any necessary changes to this file.

B) post_export.conf

Along with post_uems.conf, the post_export.conf file is always read by ems_post. The parameters in this file control the exporting of model output and any processed files to other locations. The only command-line flag associated with to file is "--noexport," which turns off the all exporting of data. See the "--noexport" option below for some additional information.

C) post_grib.conf

The post_grib.conf file controls the processing of output fields on native model surfaces to isobaric, height, theta, and potential vorticity surfaces, and then writing these fields to GRIB 2 files. This processing would be the heart and soul of the ems_post routine if it had a heart and soul, which it does. *I know, because it used to be mine*. The post_grib.conf file also controls the file processing frequency, which emsupp_cntrl.parm file to use, the precipitation accumulation (bucket dump) period, the GRIB file naming convention, and the number of cores used in processing.

D) post_gempak.conf

The post_gempak.conf file manages the processing of newly minted GRIB 2 data into GEMPAK grid files. Additional options allow for the creation of gif images with the POSTSCR parameter.

If you decide to use the GEMPAK gif generation scripts, it is recommended that you run some tests first, as additional configuration/editing/hacking may be necessary to get them to work on your local system. All the GEMPAK scripts are located under uems/util/nawips/uems/scripts/ with the individual gif generation routines identified by the leading "**gf**" in the file name. These scripts were written assuming that you are running X virtual frame buffer (**Xvfb**) so you may need to install **Xvfb**, which is pretty straightforward, provided it is still available for your system.

To start **Xvfb** create a **/etc/init.d/xvfb** file with the following contents:

```
#! /bin/sh
#
/usr/bin/Xvfb :1 -screen 0 1280x1024x24 &
```

Finally, make a link from **/etc/rc3.d/S87xvfb** to **/etc/init.d/xvfb**.

E) post_grads.conf

The post_grads.conf configuration file manages the processing of GRIB 2 files for use with Grid Analysis and Display System (GrADS). Additional options allow for the generation of gif images from the model output with the POSTSCR parameter.

Should you decide to use the semi-preconfigured GrADS gif generation scripts, it is strongly recommended that you make a few tests, as additional configuration/editing/hacking may be necessary to get them to work on your system. All the GrADS scripts are located in the **uems/util/grads/products** directory. There you will find the kumar/ directory that contains contributed gif generation routines, named for their creator Kumar Vadlamani (NOAA/NWS/CPC). Be sure to admire and appreciate his hard work and effort. I know that I do!

F) post_bufr.conf

The post_bufr.conf file contains parameters that control the post-processing of simulation output into fabulous BUFR and secondary formats. Additional options allow for the creation of GEMPAK surface, sounding, and BUFKIT files. I think there are additional parameters in there as well.

The list of BUFR sounding stations is specified by an ASCII file located in the **<domain>/static** directory. The name of this file is defined in post_bufr.conf (Default: emsbufr_stations_dWD.txt), although you may also specify a different file for each nested domain or use the same file for all domains. It's up to you. Additional information is provided in post_bufr.conf.

9.4 Running ems_post from the command line

As stated above, if no flags or options are passed to ems_post, the default behavior is to process all available primary output files for domain #1 in accordance to the parameter settings in the post_uems.conf file. Any flags and arguments passed serve to override the parameter values in the configuration files. If you are using ems_post for real-time modeling, then it is recommended that you make any necessary changes within the configuration files since you will not be able to pass them directly. You have been warned.

When running ems_post interactively, the basic usage looks like:

```
% ems_post [--domain 1,...,N] [additional optional flags]
```

You do not need to include the "--domain" flag if you are only processing output from the primary domain (domain 1). If you want to process data from a nested (child) domain, then you must include the "--domain #" flag, where "#" is the domain number to process. Multiple domains may be specified with a comma-separated list. In the absence of any command line flags, ems_post will always consult the information in the configuration files before doing whatever it wants to do.

Simulation output files must exist in the "wrfprd/" directory prior to running ems_post or else all heck will break loose, and you will get an intimidating error message and a note to take home to your mother.

One of the most useful options is "--help," which should be evident in its purpose as it provides a simple listing of the options available to the user. More specific guidance is also provided by including the flag name, i.e.,

% ems_post --help <flag>

Or

% ems_post --help domain

Nonetheless, a somewhat more informative description of the available options is provided below.

9.4.1 General ems_post options and flags

Flag: **-- emspost**

What I Do: Specifies the domain number to be post processed

Usage: % ems_post --emspost 1[:dsA[:dsB]],...,Infinity+11 [superior flags]

How You Do It:

The "--emspost" flag is typically used internally by ems_autorun to specify the domains and datasets to process following the completion of a simulation; however, that doesn't mean that you can't harness its power when running ems_post from the command line.

The argument to "--emspost" specifies the domains and datasets that you want to process. There are currently two output datasets available for processing, "primary" and "auxiliary". One or both of these datasets, along with a domain ID, may be included as part of a "rules group", the format for which is:

% ems_post --emspost ID#:dsA:dsB

Where ID# is the domain number to apply the rules, and dsA & dsB are placeholders for "primary" and|or "auxiliary", i.e, the "rules". For example:

% ems_post --emspost 2:primary:auxiliary

Specifying rules for multiple domains is accomplished by separating individual rule groups with a comma. A default rule group may also be included by excluding the domain ID. This default will be applied to any domain for which post-processing is turned ON but does not have its own rule group:

% ems_post --emspost primary:auxiliary,2:auxiliary,3,4:primary

In the above ill-advised example, domains 2, 3, and 4 are included for post-processing. The "primary:auxiliary" serves as the default rule group, which will be applied to the processing of the domain 3 output files only since both domain 2 (auxiliary) and 4 (primary) have their own rules. In the absence of any rules, only the primary dataset will be processed.

A few additional comments:

1. The order is not important, either within an individual rule group or among groups in the argument list.
2. Passing "--emspost" without an argument is the same as passing "--emspost 0" or not passing it at all.
3. The "primary" dataset refers to the "wrfout*" simulation output files.
4. The output frequency for the "primary" dataset is specified in the run_wrfout.conf configuration file.
5. The "auxiliary" dataset refers to the "auxhist1*" simulation output files.
6. The output frequency for the "auxiliary" dataset is specified in the run_auxhist.conf configuration file.
7. The "hailcast" dataset refers to the "hailcast*" simulation output files.
8. The output frequency for the "hailcast" dataset is specified in the run_hailcast.conf configuration file.
9. When using the "--emspost" flag, the "--auxhist" and --wrfout" flags are ignored.
10. If the "--domain" flag is passed, the domains will be merged with "--emspost" and the default rules applied if necessary.

The author is aware that some of the above information may be confusing or poorly worded, but that never stopped him from writing even more blather.

Flag: **--domain**

What I Do: Specifies the domain number to be post-processed

Usage: % ems_post **--domain #,** Where # is the domain number 1(or not) … N

How You Do It:

Passing the "--domain" flag specifies which domain output files to process with ems_post. You do not need to include the "--domain" flag if you are only processing output from the primary domain (domain 1). If you want to process data from a nested (child) domain, then you must include the "--domain #" flag, where "#" is the domain number to process. Multiple domains may be specified with a comma-separated list.

Flag: **--[no]scour**

What I Do: [Do Not] Whack the emsprd/ directory prior to running ems_post

Usage: % ems_post **--[no]scour** [more important options]

How You Do It:

By default, ems_post will do some selective pruning of files and directories beneath emsprd/ prior to processing model output. This includes deleting any log, error, and input files while retaining the final post-processed products from previous ems_post runs. However, any products scheduled for processing for the specified domain WILL be scoured prior to (re)creation.

Should you not care for this behavior, then passing "--noscour" will serve to keep everything intact under emsprd/. You might think "--noscour" is the perfect flag for you, but it serves a limited purpose. For example, if you process a subset of available simulation output into GRIB2 and then want to continue with the remaining files, the first batch of GRIB files will be deleted prior to the creation of any new files. This is true even when the files are not included in the second group. Note that this only applies to files of the same domain and file type (primary or auxiliary). All other files are spared the wrath of the UEMS sanitation machine.

Passing "--scour" tells ems_post to perform a heavy handed "cleansing" of the emsprd/ directory, meaning that all the subdirectories are deleted prior to performing any new post-processing activities. You are more likely to use "--scour" than "—noscour," which is why it was created just for you.

Finally, the exception to the above behavior is when the UEMS autopost feature is ON or when also passing the "--noupp" flag, in which case NO scouring is completed, i.e., same as "--noscour".

You probably will not need this option, but it's there for taking.

Flags: **--info & --summary**

What I Do: Print a summary of the post-processing tasks

Usage: % ems_post --info | --summary [other options, like you mean business]

How You Do It:

Passing "--info" causes ems_post to print a summary of the post-processing tasks to be completed and then exits without any processing being done. Use this flag to check the final settings based upon input from the configuration files and any command-line flags passed. The flag gives you the opportunity to "try it before you buy it" with ems_post.

The "--summary" flag is similar to "—info," except that it continues with the data processing as described in the printed summary. You would use the "--summary" if you think you know what you are doing but want some reassurance.

"Try it before you buy it" is the best approach in life.

Flag: **--wrfout**

What I Do: Requests the processing of primary history (simulation output) files

Usage: % ems_post **–wrfout** [whatever suits your fancy, again]

How You Do It:

Passing "—wrfout" tells ems_post that you want to processes the primary netCDF files (wrfout*) output during the simulation. These files are activated when the user specifies an output frequency greater than 0 (default; 60) in run_wrfout.conf prior to running a simulation.

Since ems_post defaults to processing primary output files if nothing else is passed, this flag only serves a purpose when combined with "—auxhist," in which case both file types will be processed.

Flag: **--auxhist**

What I Do: Requests the processing of hailcast output files

Usage: % ems_post **--hailcast** [whatever suits your fancy, should you have a fancy in need of a suit]

How You Do It:

Passing "--hailcast" tells ems_post that you want to processes the hailcast files output during the simulation. These files are activated when the user specifies an output frequency greater than 0 (default; OFF) in run_hailcast.conf prior to making a run.

Flag: **--hailcast**

What I Do: Requests the processing of primary history (simulation output) files

Usage: % ems_post **–wrfout** [whatever suits your fancy, again]

How You Do It:

Passing "—wrfout" tells ems_post that you want to processes the primary netCDF files (wrfout*) output during the simulation. These files are activated when the user specifies an output frequency greater than 0 (default; 60) in run_wrfout.conf prior to running a simulation.

Since ems_post defaults to processing primary output files if nothing else is passed, this flag only serves a purpose when combined with "—auxhist," in which case both file types will be processed.

Flag: **--noexport**

What I Do: Turn off the export of processed files to exotic destinations and workstations

Usage: % ems_post **--noexport** [type1,type2,...,typeN] [other options]

How You Do It:

Passing "--noexport <string>" tells ems_post that you want to turn off the exporting of files that are otherwise scheduled for departure in post_export.conf. Including the "type1,type2,..,typeN" argument serves to selectively turn OFF exporting of specific file types identified by the EXPORT parameter in the configuration file. Each "type" must match a "FILE TYPE" defined as part of EXPORT string in the file.

For example,

% ems_post **--noexport GRIB,BUFR** [other options]

Will turn off the exporting of GRIB and BUFR formatted files, whether they were scheduled in the configuration file or not. The string should be carefully specified so as not to turn off the exporting of files unintentionally.

In the absence of an argument, passing "--noexport" will turn off the export of all data types listed in the configuration file, regardless of your intent.

9.4.2 Options when creating GRIB files, or not

Flag: **--[no]grib**

What I Do: Turn ON [OFF] processing of netCDF into GRIB 2 and all derived products

Usage: % ems_post **--[no]grib** [**FREQ:START:STOP**] [less important stuff]

How You Do It:

Passing "--[no]grib" gives you the type of superpower you always wanted while in college, including the ability to focus every cell phone at a party on you, just after you've screamed "Hey, Watch This!". *Use the power wisely my friend.*

The "--[no]grib" flag is used to override the GRIB parameter value in post_uems.conf. If GRIB = Yes and you want to turn OFF processing, just pass "--nogrib" and ems_post will magically not make it happen for you. Note that turning OFF the creation of GRIB files also turns OFF processing of all secondary data formats such as GrADS and GEMPAK sounding files.

Passing "--grib" turns on the creation of GRIB 2 files with the EMSUPP routine. This flag might be used if you set GRIB = No in post_uems.conf and are too lazy to change it. However, the

primary purpose for this option is to allow for finer control over the processing of simulation netCDF files into GRIB 2.

The "--grib" flag also accepts an argument that specifies the start and stop times along with the frequency of netCDF files to process, "FREQ:START:STOP". All values are all defined in minutes from the start of integration for the domain being processed.

Here are a few examples:

% ems_post --grib 60

Translation: Have the EMSUPP process 60-minute (hourly) output netCDF files from the primary simulation (wrfout) into GRIB2. If the frequency of simulation output is hourly, then the "60" was not necessary and you don't know what you are doing. If the frequency of wrfout files is 30-minute, then ems_post will only process every other netCDF file. If the output frequency from the model was less than 60-minute, say 3-hourly, then ems_post will adjust the processing frequency to that of the model output and then give you a snarky response before continuing.

% ems_post --grib 60 --auxhist

Translation: The same as for the previous example except the auxiliary history files will be processed rather than the primary output from the simulation.

% ems_post --grib 60 --auxhist --domain 2

Translation: The same as for the previous example for the auxiliary history files but this time output for domain 2 will be processed.

% ems_post --grib 60:180

Translation: Have the EMSUPP process 60-minute (hourly) output netCDF files from the primary simulation (wrfout) into GRIB2, beginning with the 180-minute (3-hour) output file and continuing through the final model output time. In this example, all simulation output before the 3-hour mark will be skipped.

% ems_post --grib 60:180:720

Translation: Have the EMSUPP process 60-minute (hourly) output netCDF files from the primary simulation (wrfout) into GRIB2, beginning with the 180-minute (3-hour) output file and continuing through the 12-hour output file.

% ems_post --grib ::720

Translation: Have the EMSUPP process every domain 1 simulation output netCDF file (wrfout) into GRIB2, beginning with the zero hour (T0) output file and continuing through the 12-hour output file.

So you can see where this is going and can figure the rest out for yourself. Note that the

management of GRIB file generation does impact secondary file formats such as GrADS and GEMPAK, since these data are created from GRIB.

Flag: **--noupp**

What I Do: Turn off processing of netCDF to GRIB format via UEMS UPP

Usage: % ems_post --noupp [other options, but nothing too stupid]

How You Do It:

Passing "--noupp" turns off the processing of netCDF files into GRIB 2 format. Use this flag if GRIB files have already been created, presumably during a previous running of ems_post, but you now want to create a dataset that is derived from GRIB such as GrADS or GEMPAK. If this is your intent, then be sure that the necessary GRIB files exist; otherwise, you should not be playing with this flag.

Flag: **--ncpus**

What I Do: Specify the number of CPUs to use for running EMSUPP

Usage: % ems_post --ncpus #CORES

How You Do It:

Passing "--ncpus" overrides nearly all other parameters in specifying the number cores to use when running EMSUPP on a local system. The argument is an integer number of processors (cores) you wish to use. If the value passed is anything other than an integer greater than zero, the flag will be ignored. If the value exceeds the number of cores (not threads you hyper-threaders!) on the system, the value will be set to the total number of available cores, and you will like it.

The "--ncpus" flag overrides both the EMSUPP_NODECPUS parameter (post_grib.conf) and the "--emspost" flag by specifying that the EMSUPP runs on the local machine only using the requested number of cores, which is the same as setting "EMSUPP_NODECPUS = local:#cores" in the configuration file.

9.4.3 Options when creating GEMPAK grid files

Flag: **--[no]gempak**

Note: The UEMS includes a full installation of NAWIPS/GEMPAK display software for visualizing simulation output; however, the package is turned OFF by default in the etc/EMS.cshrc|profile files (EMS_NAWIPS) since many users already have GEMPAK installed on their system or they are not familiar with the software. If you wish to install the UEMS GEMPAK package, then break out the trusty “uems_install.pl” routine and run:

```
% uems_install.pl --addpack nawips
```

Afterward, uncomment the EMS_NAWIPS environment variable in the etc/EMS.cshrc|profile file.

Now, back to the flag:

What I Do: Override the generation of GEMPAK files as defined in **post_uems.conf**.

Usage: % ems_post **–[no]gempak** [other options]

How You Do It:

Passing "--[no]gempak" overrides the GEMPAK parameter setting in post_uems.conf. It should come as no surprise that passing "--nogempak" turns OFF the creation of GEMPAK files, while "--gempak" turns them ON with all processed files located in the emsprd/gempak/ directory.

Note that the --[no]gempak flag only serves to turn ON|OFF GEMPAK file processing. The frequency of the GRIB files to be processed into GEMPAK is controlled by the FREQ_WRF|AUX_GEMPAK parameter in post_gempak.conf and the specified GRIB file processing.

Finally, the NAWIPS/GEMPAK package provided by the UEMS includes executables that are compiled for use with shared system libraries. Consequently, you may need to install libraries that were excluded when the Linux OS was installed. To determine whether you are missing a shared library, you can use the "uems/util/nawips/bin/libcheck" utility.

9.4.4 Options when creating of GrADS grid files

Flag: **--[no]grads**

What I Do: Override the processing of GRIB2 into GrADS files as defined in post_uems.conf

Usage: % ems_post **--[no]grads**

How You Do It:

The UEMS includes an installation of the GrADS display software for visualizing simulation output, which is located under uems/util/. If you already have the GrADS installed and just want to create GrADS files, the ems_post routine will still use the UEMS installation for file processing, but you may use your installation for viewing the data.

Passing "--[no]grads" overrides the GRADS parameter setting in post_uems.conf. It should come as no surprise that passing "--nograds" turns OFF the creation of GrADS files, while "--grads" turns them ON with all processed files located in the emsprd/grads/ directory provided everything went well, which it always does for the developer.

Note that the --[no]grads flag only serves to turn ON|OFF grads file processing. The frequency of the GRIB files to be processed into GrADS is controlled by the FREQ_WRF|AUX_GRADS parameter in post_grads.conf and the specified GRIB file processing.

9.4.5 Options when creating BUFR, GEMPAK, and BUFKIT sounding files

Flag: **--[no]bfinfo**

What I Do: [Do Not] write lots of interesting blather to the BUFR log file

Usage: % ems_post **–[no]bfinfo** [other options]

How You Do It:

Passing the "--[no]bfinfo" flag overrides the BUFR_INFO setting in the post_bufr.conf file. Passing "--bfinfo" or setting BUFR_INFO = YES in post_bufr.conf will include additional and hopefully interesting information in the post_emsbufr.log file. The downside is that all the extra I|O will slightly increase processing time but may also enhance your social status as you will have plenty of fodder for scintillating conversation with complete strangers while waiting for your bus to arrive.

Flag: **--[no]bufr**

What I Do: Turn ON [OFF] processing of netCDF into BUFR and all derived products

Usage: % ems_post **--[no]bufr [FREQ:START:STOP]** [some less important stuff]

How You Do It:

Passing "--[no]bufr" gives you the type of superpower you always wanted while in elementary school, except for X-ray vision, which will be included in a future UEMS release.

The "--[no]bufr" flag is used to override the BUFR parameter value in post_uems.conf. If BUFR = Yes and you want to turn OFF processing, pass "--nobufr" and ems_post will magically [not] make it happen for you. Note that turning OFF the creation of BUFR files also turns OFF processing of all secondary data formats such as BUFKIT and GEMPAK sounding files. Alternatively, passing "--bufr" turns ON BUFR file generation; however, the processing BUFR into secondary formats is dependent upon the parameter settings in post_bufr.conf and command-line flags.

Besides the guidance provided above, you can fine-tune your BUFR files processing by including the "FREQ:START:STOP" option (because it's optional). This string overrides the value of FREQUENCY_WRFOUT in post_bufr.conf by defining the frequency (minutes) of raw netCDF files to process into BUFR, the data file with which to start, and the final file time to process. The START & STOP values are specified in minutes from To (integration start time for the domain being processed) and FREQ is the number of minutes between data files. For

example, if the output frequency of a simulation is every 15 minutes but you want to only process 30-minute data between minute 60 and 180, pass:

% ems_post --bufr 30:60:180

The UEMS checks to ensure that your time range falls within that of your simulation, so you can't mess things up too badly. Note that any secondary BUFR file processing will be impacted if you use this option since the resultant files will be carried into the additional steps.

Flag: **--[no]gemsnd**

What I Do: Override the generation of GEMPAK sounding files as specified in post_bufr.conf

Usage: % ems_post **--[no]gemsnd** [other options]

How You Do It:

Passing "--[no]gemsnd" overrides the generation of GEMPAK sounding files as specified in the post_bufr.conf file, just as you read above, but I want to make sure you are paying attention. Passing "--gemsnd" also turns on BUFR file processing whether you like it or not since the GEMPAK station files are created from BUFR. Conversely, passing "--nogemsnd" does not affect the processing of BUFR files.

After the smoke clears, all GEMPAK sounding files will be located in the "emsprd/gemsnd/" directory. If you want to export the files to other exotic locations, then check out the possibilities presented in the post_export.conf file. You'll be glad you did.

Flag: **--[no]bufkit**

What I Do: Override the generation of BUFKIT files as specified in post_bufr.conf

Usage: % ems_post **--[no]bufkit** [anything else you might have on your mind]

How You Do It:

Passing "--[no]bufkit" overrides the generation of BUFKIT files as specified in the post_bufr.conf file, just as you read above, but I want to make sure you are paying attention. Passing "--bufkit" also turns on BUFR file processing whether you like it or not since the BUFKIT station files are created from BUFR. Conversely, passing "--nobufkit" will not affect the processing of BUFR files.

After the smoke clears, all BUFKIT files will be located in the "emsprd/bufkit/" directory. If you want to export the files to other exotic locations, then check out the possibilities presented in the post_export.conf file. You'll be glad you did.

Flag: **--[no]gemsnd**

What I Do: Override the generation of GEMPAK sounding files as specified in post_bufr.conf

Usage: % ems_post **--[no]gemsnd** [Advertise your goods & services here]

How You Do It:

Passing "--[no]gemsnd" overrides the generation of GEMPAK sounding files as specified in the post_bufr.conf file, just as you read above, but I want to make sure you are paying attention. Passing "--gemsnd" also turns on BUFR file processing whether you like it or not since the GEMPAK station files are created from BUFR. Conversely, passing "--nogemsnd" does not affect the processing of BUFR files.

After the smoke clears, all GEMPAK sounding files will be located in the "emsprd/gemsnd/" directory. If you want to export the files to exotic locations, then check out the possibilities presented in the post_export.conf file. You'll be glad you did, just like the bufkit files.

9.5 Managing the fields output to your GRIB files

UEMS users can control the fields and levels contained in the GRIB2 files created when running ems_post. Individual fields may be turned ON or OFF on a variety of vertical levels and surfaces. This data management is accomplished through the use of a control file that is read by the UEMS version of the Unified Post Processor (EMSUPP). Various versions of control files can be found in the uems/data/tables/post/grib2 directory. The default files used with the primary and auxiliary datasets have "*.MASTER" at the end of the filenames, e.g., emsupp_auxcntrl.MASTER & emsupp_cntrl.MASTER and are copied into <domain>/static/ as emsupp_auxcntrl.parm & emsupp_cntrl.parm when a new domain is created. The other available control files include:

emsupp_cntrl.HYBRID_ALL - Contains all fields available on hybrid (native model) surfaces. If you want to include these fields in your GRIB files, replace the list of hybrid level fields in the existing static/emsupp_cntrl.parm with the contents of this file (cut & paste). Make sure to read the information provided regarding the specification of levels.

emsupp_cntrl.PRESS_ALL - Similar to emsupp_cntrl.HYBRID_ALL, this file contains all fields available on pressure (isobaric) surfaces. If you want to include these fields in your GRIB files, simply replace the list of pressure level fields in the existing static/emsupp_cntrl.parm with the contents of this file (cut & paste).

emsupp_cntrl.THETA_ALL - Just like emsupp_cntrl.HYBRID_ALL & emsupp_cntrl.PRESS_ALL, only with theta surfaces.

emsupp_cntrl.parm.ARW-native - This file is used to create GRIB 2 files from an existing UEMS simulation that serve as the initialization data set for a different WRF ARW run. This file replaces emsupp_cntrl.parm, which is residing in your <domain>/static directory. This file contains only those fields necessary for model initialization. Be sure to read the additional guidance provided at the top of the file before proceeding.

emsupp_cntrl.parm.ARW-pressure – Similar to emsupp_cntrl.parm.ARW-native except that the fields used for initialization are on isobaric surfaces. If your primary (donor) domain contains fewer than 80 hybrid levels and you like pressure, use this file. If you have more than 80 hybrid levels, say 500, use the emsupp_cntrl.parm.ARW-native file.

emsupp_cntrl.MASTER+CRTM – This file contains all the fields in emsupp_cntrl.MASTER plus all available synthetic satellite imagery fields. The reason for this file not being the default control file in the UEMS is that turning ON the creation of synthetic satellite imagery significantly increases the time requires to generate the GRIB 2 file. Buyer beware!

9.5.1 GRIB file contents for different domains

It is possible to specify different control files for each domain included in a simulation, thus allowing the resulting GRIB files to have a different set of fields and levels. This configuration is accomplished through the GRIB_CNTRL_WRF|AUX parameters in the post_grib.conf file. These parameters contain a comma-separated list of control files, with each entry corresponding to the file used when processing a specific domain. For example:

```
GRIB_CNTRL_WRF = emsupp_cntrl_d01.parm, emsupp_cntrl_d02.parm, ..., emsupp_cntrl_doN.parm
```

Wherein "emsupp_cntrl_d01.parm" will be used for the primary domain (domain 1), "emsupp_cntrl_d02.parm" will be used for domain 2 (1st nested domain), etc. The name of the fine is not important, as long as that file can be found under the "<domain>/static" directory. Also, it is the position in the list and not the filename that determines the file to be used with a specific domain. For example, if a list contains:

```
GRIB_CNTRL_WRF = emsupp_cntrl_d02.parm, emsupp_cntrl_d01.parm, ..., emsupp_cntrl_doN.parm
```

The "static/emsupp_cntrl_d02.parm" file will be used with domain 1 and "emsupp_cntrl_d01.parm" with domain 2. If ems_post fails to locate a control file, it will immediately terminate and give you a tongue-lashing, the likes of which you have never seen from any Microsoft product. Finally, you do not have to specify a unique control file for each domain. The last file in the list becomes the default for any additional domains to process.

9.5.2 Deciphering the emsupp control file

The initial look at an emsupp control files can be very confusing. You will find many semi-cryptic field names followed by an incomprehensible string of 1s and 0s. It is this string of 1|0s that you will edit to turn **ON|OFF** a level for each field represented in the file. The emsupp routine is expecting to find a string of 80 1|0s organized into 16 groups of 5 values. It doesn't matter that "surface pressure" is only available on 1 level, having fewer than the 80 levels represented will cause the UEMS post processor to crash, so don't attempt anything funny, because the UEMS doesn't do "funny."

Multi-level fields

There are many fields available on multiple surfaces and coordinate levels. For example, temperature is available on pressure, theta, height (both AGL & ASL), potential vorticity, pressure AGL, at the

surface, and on a few other vertical coordinates. Each vertical coordinate has its own set of individual levels available. These individual levels are assigned a value of 0 or 1 depending on whether you want to turn a particular level OFF (0) or ON (1). Here is an example of a line showing the distribution of levels to be output for a particular field to the GRIB file:

L=(11111 11111 11111 11100 10010 01001 00100 10010 01001 00100 10010 01001 00100 10010 01001 00101)

For the sake of this discussion, we will assume these represent pressure levels although they could also be hybrid, height, potential vorticity, or isentropic levels.

The DEFAULT pattern of 1s and 0s, as depicted above, outputs 3D fields every 25 hPa between the surface and 25hPa. Additional pressure levels have been added that includes every 10mb from the surface to 500mb for those users needing higher resolution within the lower troposphere. *Remember that the vertical resolution of the 3D fields in the GRIB files is no better than the vertical resolution of the model!*

The complete list of available pressure surfaces (hPa) in groups of 5, as represented by the 1s and 0s from left to right is:

Pressure Levels (hPa)					ON/OFF
10	25	50	100	150	**11111**
200	225	250	275	300	**11111**
325	350	375	400	425	**11111**
450	475	500	510	520	**11100**
525	530	540	550	560	**10010**
570	575	580	590	600	**01001**
610	620	625	630	640	**00100**
650	660	670	675	680	**10010**
690	700	710	720	725	**01001**
730	740	750	760	770	**00100**
775	780	790	800	810	**10010**
820	825	830	840	850	**01001**
860	870	875	880	890	**00100**
900	910	920	925	930	**10010**
940	950	960	970	975	**01001**
980	990	1000	1010	1013	**00101**

Each of the five 1s and 0s in the last column of each row corresponds to the pressure level listed in the leading five columns. For example, the first line in the list, **10,25,50,100,150 (hPa)**, corresponds to the block of five 1s and 0s on the right (**11111**).

Again, a **"1"** indicates that the level is **ON** while a value of **"0"** means it is turned **OFF**.

The last row of pressure levels, **980, 990, 1000, 1010, 1013 hPa**, corresponds to the block of **1s & 0s (00101)** and indicates that the **1000 and 1013 pressure levels will be turned ON** in the GRIB output while the **980, 990, and 1010 hPa levels are OFF**.

Single level fields

Many fields are only valid on a single level, such as the surface. **In this case, the output is controlled by the first 1/0 listed in the string**. So, if you want total precipitation turned ON, the entry in the **emsupp_cntrl.parm** would look like:

```
(ACM_APCP_ON_SURFACE)        SCAL=(4.0) !  Simulation Accumulated Total Precipitation (Liquid Equivalent; kg/mp2)
L=(10000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000)
```

Finally, fields are available over a variety of depths, such as sub-surface and boundary layer based fields. An example of this would be storm-relative helicity and convective available potential energy. The storm-relative helicity field is available from the surface to 1000 and 3000 meters above the ground. For these fields, the levels are controlled by a string of 1s and 0s beginning on the left:

```
(HLCY_ON_SPEC_HGT_LVL_ABOVE_GRND)        SCAL=(4.0) !  Storm Relative Helicity over Height Layer (m2 s-2)
L=(11010 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000)
```

Chapter 10: Harness the power of ems_autorun.pl

Chapter Contents:

10.1 Living La Vida ems_autorun.pl

The purpose of the ems_autorun.pl routine is to automate the entire process of running a simulation, from data acquisition and model initialization to managing the output files. This routine reads a user-controlled configuration file and then executes ems_prep, ems_run, and ems_post in succession. It is ideally suited for real-time forecast applications; however, it may also be used to conduct case and sensitivity studies. For real-time forecasting, there are features and options to improve the reliability of your forecasts. Additionally, there is even an option for processing output data concurrent with a run.

The ems_autorun routine is typically executed from within a shell script or wrapper file initiated via cron. If you prefer a more hands-on approach, the routine can be started from the top level of a domain directory. There you will find a link from ems_autorun to uems/strc/Ubin/ems_autorun.pl. Additionally, the uems/strc/Uauto/ directory contains libraries used by ems_autorun.pl. Never run ems_autorun.pl directly. If the ems_autorun link gets deleted, it can be recreated by running "**ems_clean --level 0**." For the sake of clarity, "ems_autorun" is used throughout this chapter to refer to the link that exists in each domain directory.

10.2 Doing it right the first time - configuring ems_autorun

When creating a new domain, either with the Domain Wizard or ems_domain, a copy of the default configuration is written to conf/ems_auto/ems_autorun.conf. Inside this exceptionally well-documented file are the parameters used when running ems_autorun. It's in your best interest to review the information. You will be mighty glad you did.

Important: Although there is only one configuration file used by ems_autorun, you will likely need to also edit those for ems_run, ems_post, and if desired, ems_autopost (Section 10.6).

10.3 Getting the most out of your real-time forecasting experience

As stated above, the ems_autorun routine is ideally suited for real-time forecasting, allowing users to run the UEMS on a regular schedule without intervention. At least that's the plan. To facilitate this process, an entry is placed in the user's crontab file during installation.

Using the "crontab -l" command, you should something like:

```
#8 2 * * * /usr1/uems/strc/Ubin/uems_autorun-wrapper.csh --rundir /usr1/uems/runs/<domain> >& /usr1/uems/logs/ems_autorun.log 2>&1
```

Important: The tiny example above should serve as the template for all real-time forecasting applications.

The wrapper files for both (T)csh and Bash shells are located under uems/strc/Ubin. Because it is important that the environment variables are set prior to running ems_autorun*, uems_autorun-wrapper.<shell>, and not ems_autorun.pl*, should be run from the cron. The wrapper scripts set the environment prior to starting ems_autorun.pl, and any arguments to *uems_autorun-wrapper.<shell>* are passed through.

To initiate a regular real-time forecast, a user should:

1. Create the computational domain using ems_domain or Domain Wizard.
2. Edit the **ems_run** configuration files under conf/ems_run/
3. Edit the **ems_post** configuration files under conf/ems_post/
4. Edit **ems_autopost.conf** (if using concurrent post-processing)
5. Edit **ems_autorun.conf** under conf/ems_auto/
6. Edit your crontab file ("crontab -e"):
   i. Remove the comment ("#") from the crontab entry
   ii. Specify the time(s) to initiate the forecast process ("8 2 * * *")
   iii. Specify the computational domain to use (<domain>)
7. Get the start time for your forecast correct:

   Once you have selected the initialization data set(s) for your real-time forecasts, be sure to note when these data become available on the remote server(s). The number of hours between the timestamp in the filename and when those data are made available on the host server must be taken into account. This time lag is specified by the **DELAY** parameter in the <data set>_gribinfo.conf files located under uems/conf/gribinfo/. The **DELAY** setting defines the number of hours following an official "cycle time" before the files are made available.

   The information contained in the gribinfo/ files is used to determine what data is available at any time of day. For example, if you wish to use data from the most current operational GFS,

which has a **DELAY** value of 3 (hours), then files from the 00 UTC will not be available until sometime after 03 UTC. If you start your real-time run before 03 UTC, even at 2:59 UTC, the UEMS will attempt to download files from the previous GFS cycle (18 UTC previous day), which is probably not what you want to happen.

All the gribinfo/ files come pre-configured with a reasonable **DELAY** value for each supported data set; however, availability times do change, so it's in your interest to monitor the situation. Also, if you are using an operational data set to initialize your forecasts, then understand that there may be additional delays that can't be accounted for by the UEMS. I know it's surprising, but even the UEMS has limitations, even if it won't admit them.

Important: You may also want to ensure the system time on your computer is correct as this can be a source of problems, especially with incorrect time zone information.

8. Before running from cron, test, test, test!

Prior to initiating a simulation from cron, each step in the process should be tested by running the individual run-time routines in succession. This task will hopefully expose any obvious mistakes or problems in the configuration, and thus, will keep you from the personal devastation experienced when your run fails.

Begin by running the ems_prep routine. Include the "--dset", "--length" and "--domains" flags just for practice. Follow ems_prep with ems_run and then ems_post. Make sure each step completes successfully. Pay special attention to the time required for ems_run to complete and whether it fits into your operational window. You should also consider inspecting the simulation output in case a domain needs to be relocated or you don't like the configuration choices. Now is the time to sweat the small stuff.

Once you are happy with the results, try running ems_autorun from the command line (Section 10.4). Again, pay attention to the total amount of time required for the entire simulation to complete. If you plan to use the autopost option, this is a good opportunity to ensure that the routine is working.

Only after everything runs satisfactorily should you initiate your forecasts from the cron.

10.4 Driving ems_autorun from the command line

For testing a real-time forecasting system or executing a case study simulation, ems_autorun may be run directly from the command line:

```
% ems_autorun [ Other flags if you are in the mood]
```

Where **[flags]** includes the available flags (Section 10.5).

When using ems_autorun for case studies, it is critical to include the "--date" and "--cycle" flags since the system will default to the current date and cycle time for initialization data. Also recommended is that you pass the "--noscour" flag as the default configuration (**SCOUR = Yes** in ems_autorun.conf)

will delete all files from grib/ before running ems_prep, which is probably not what you want for a case study. Alternatively, you can set **SCOUR = No** and be done with it. You have other stuff about which to worry.

10.5 You're the boss! The available command line options

The flags and arguments available to ems_autorun primarily serve to override existing settings in the configuration files. Some of the options are used by ems_autorun directly, while others are passed along to the various run-time routines. *Just like with magic, it's more fun if you don't ask too many questions.*

As always, the most useful flag is "--help," which should be evident in its purpose as it spits out a listing of all available flags.

```
% ems_autorun --help
```

Including a flag name as an argument to "--help" provides a more detailed description:

```
% ems_autorun --help dset
```

Many of the ems_autpost flags are the same as those used by the other run-time routines, particularly ems_prep.

10.6 Getting your groove on with concurrent post-processing

The UEMS can process output from a simulation while it is running. This functionality is quite useful for real-time forecasting purposes when timeliness of the information is critical. This option is activated by the **AUTOPOST** parameter in the ems_autorun.conf file; however, using the auto post-processing option is not without its caveats:

Caveat: *Running autopost on the same system as the simulation will severely degrade the performance of the run!*

If you are thinking about using this auto post-processing feature, and I know you are, it is strongly recommend that you configure a separate machine for this task. It is easy to do, and the autopost will handle most of the details. All you need is to complete the following steps:

1. Find a Linux system with a minimum 8Gb of memory. The system should be running an x64 Linux distribution similar to that on the primary (simulation) machine to avoid headaches.

2. The UEMS must be available on the autopost machine and in the same directory path as that on the primary system. The simplest way of doing this is to export the UEMS partition to the autopost machine and create any necessary links.

3. Create a user on the autopost machine with the same name, ID, and group, as that on the primary system. You can also export the user's home directory to the autopost machine is you don't want to manage individual home directories. The accounts must be the same for permission purposes and to ensure all environment variables are correctly set.

4. Configure passwordless SSH back and forth (primary <--> autopost) between the two systems for the UEMS user. One-way (primary --> autopost) is not sufficient as the autopost machine must monitor the simulation system to determine when the run has completed.

   The "cnet" command can be used as a basic test to determine whether there are any communication issues between machines. The primary purpose of cnet is to determine whether another system is reachable from the machine on which it is being run. The program takes either a host name or IP address as an argument and then skillfully checks whether that address can be resolved on the local network and if the system is reachable via passwordless ssh.

   CNET USAGE:

   % cnet [--verbose] [--nossh] hostname1 … hostnameN

   If the answer to those magic questions is "Yes", then cnet spits out the hostname, IP, and network interface used to reach the remote system.

   A problem is indicated by a "0" value returned on the command line. For Example:

   % roz@parlee -> cnet kielbasa --ssh
   kielbasa 128.117.127.55 eth0 1 - *All good - no obvious problems*

   But

   % roz@parlee -> cnet kielbasa --ssh
   kielbasa 128.117.127.55 eth0 0 - *Your bad - The ssh test failed (0)*

   Returned values of 0 (host, ip, iface, [ssh]) indicate that there was a problem, which may be better illuminated by passing the '--verbose' flag.

   % roz@parlee -> cnet kielbasa –ssh --v
   ☺ **Collecting information about the machine known as "kielbasa"**

   ! Error connecting to kielbasa: Passwordless SSH Connection error to kielbasa - Connection refused

   The 2-way SSH configuration is necessary is because the machines must communicate when their tasks have been completed. When autopost is ON, the ems_autopost.pl routine checks whether the simulation has ended before making a final processing loop. The ens_run routine then waits for autopost to complete before doing a final clean up and exiting.

5. The most common cause of problems when running autopost is the firewall configuration. When troubleshooting, it is recommended that you turn OFF your firewall and SELinux, even if your local system administrator or network security people don't like it. Besides, you have important work to accomplish. Once you determine that the problem is related to local security, you can incrementally enable the security configuration until you have isolated the source of the issue.

6. Finally, edit the ems_autopost.conf file under <domain>/conf/ems_auto/ so that it contains the hostname of the post-processing machine and the number of available CPUs.

Preface – For users still looking for "reading materials"

Caveat Lector!

The Science and Training Resource Center (STRC) Unified Environmental Modeling System (UEMS) is a complete and relatively easy to use state-of-the-science numerical weather prediction (NWP) modeling system. Nonetheless, users (that would be you!) are forewarned that running any numerical model requires some understanding of both atmospheric science and computers. Even the most NWP-savvy individuals experience complete and demoralizing failure, some of which is their own fault whether they want to admit it or not. Occasionally, the problems encountered are the fault of others, but they're not going to admit it either. Other times, stuff just happens for unexplained reasons that are outside the control of anybody or anything. This is real science, and while it isn't perfect, it's getting very close. Consequently, the National Oceanic and Atmospheric Administration (NOAA), National Weather Service (NWS), Forecast Decision Training Division (FDTD), and the Science Operations Officer (SOO) STRC will take no responsibility for the accuracy or reliability of your forecasts, even if you ask nicely.

Besides, the world of NWP can be a dangerous place regardless of your forecast. Understand that while attempting to mine the potentially vast riches the UEMS has to offer, stuff can happen and things may break that can't be fixed by the lone forsaken support person. That's just the way life is, sometimes challenging and unfair. But this cold hard fact shall not be a deterrent, as you are a ray of simulated sunshine, blinding the eyes of evil and illuminating an atmospheric river to freedom for abandoned puppies. With the unbridled power of the UEMS, your glass of cucumber-infused optimism is always half full. And please don't look behind the curtain, because the UEMS Overlord never wears clothing.

When asking for help from UEMS Support Central, be kind and patient. Adulation and offers of baked goods sometimes work, but not always (although it's worth the risk). If you are confused or concerned, then stop reading now, close this guide, wash your hands, and run far, far away; otherwise, if you want the thrill and adventure of running your personal NWP system, then carry on citizen modeler!

This document provides nearly unlimited guidance for using the UEMS. However, it should be noted that while a half-hearted attempt has been made to ensure that this guide is complete, the prose was written under the influence of sleep deprivation, a highly precocious 16-year old, and an exceptionally intelligent and beautiful wife; not necessarily in that order. Consequently, there is a good chance that errors, both intentional and unintentional, may exist.

Any questions or suggestions for improvement should be sent directly to the author, just because he cares.

As the UEMS chief marketing strategist, campaign manager, and washroom attendant requires me to say:

"Think Globally, Model Locally"

Seventh Printing: Release 19, March 2018
Author: Robert Rozumalski – NOAA/NWS/OCLO/FDTD
SOO/STRC Website: http://strc.comet.ucar.edu

Contact Information:

Robert A. Rozumalski, PhD
NOAA/NWS National SOO Science and Training Resource Coordinator
Forecast Decision Training Division
COMET/UCAR PO Box 3000
Boulder, CO 80307-3000
Phone: 303.497.8356

Robert.Rozumalski@noaa.gov

Appendix A: Adding and modifying initialization datasets

Appendix Contents:

A.1 Welcome to the gribinfo.conf files!

When ems_prep or ems_autorun is run from the command line, the user specifies the data set(s) to use for the model initial and boundary conditions by passing the "--dset" flag. This flag is optional when running ems_autorun but mandatory for ems_prep:

% ems_prep --dset nam212 [other options]

The above example shows the NAM 212 grid being used for both model initialization and boundary conditions. There are, however, many data sets from which to choose. You can get a summary of the available data sets by passing the "--dslist" flag to ems_prep:

% ems_prep --dslist

Following which you should see a brief description of the available initialization data sets along with their moniker that can be passed as an argument to --dslist. Should you want more information about a specific data set then simply use the "--dsinfo" option in ems_prep:

% ems_prep --dsinfo <moniker>

Or

% ems_prep -- dsinfo nam212

The information displayed on the screen comes from the **<moniker>_gribinfo.conf** files located in the **uems/conf/gribinfo** directory. Each file defines the default attributes for each data set available for model initialization. When accessing these files, <moniker> is replaced with a key word that identifies a specific data set. So when you run ems_prep and include "--dset gfs", ems_prep will look in **gfs_gribinfo.conf** to get all the default settings for that data set. If you need to modify the information for an existing data set or create a new one, these are the files you would change.

A.2 Anatomy of a gribinfo.conf file

There will be times when you need to modify the contents of a _gribinfo.conf file or create a new file for a data set you wish to add. To add a new data set, simply copy an existing template and edit the information as necessary. There is documentation in each file to assist you, although a more compete summary of the parameters included in a gribinfo.conf file is provided below:

A.2.1 CATEGORY

What I Do: Specifies the general data set type

Description:

The CATEGORY parameter is used to categorize this data set for the purpose of better organizing the available initialization data options when passing the "--dslist" flag to ems_prep. The category may be anything you choose but it's recommended that you stick to a few established conventions unless you have a good reason to create your own. The current category list includes:

Category		Description
Land Surface Model	(LSM)	Data sets containing LSM-related fields (--lsm)
Surface	(SFC)	Data sets containing static surface fields (--sfc)
Forecast	(FCST)	Operational Forecast data sets
Analysis	(ANAL)	Operational Analysis data sets (--analysis)
Model Forecast	(MODL)	Data sets from Non-operational model runs
Historical	(REAN)	Historical or Reanalysis data sets

The following may be appended to the category to indicate a personal tile data set

Personal Tiles	(PTIL)	STRC Personal tile data sets

If you want something different just make up a category name and it will be handled appropriately by ems_prep.

Leaving CATEGORY blank or undefined will result in the data set being placed in the "Land of misfit data sets" category.

A.2.2 INFO

What I Do: Provides a brief summary of the data set

Description:

The INFO parameter provides some general information about the data set such as forecast frequency, vertical and horizontal resolution and coordinate system. Any information that is able to be fit on a single line.

Example: **INFO** = .5 degree Global - Isobaric coordinate - 3hourly grib format

A.2.3 VCOORD

What I Do: Identifies the vertical coordinate used for the 3D fields

Description:

VCOORD identifies the vertical coordinate of the primary 3D fields contained in the data set. This is necessary because mismatched vertical coordinated in the initial and boundary condition data sets may cause problems, although this issue may be corrected in the future.

Typical values include:

press	Isobaric Coordinate
hybrid	Hybrid Coordinate (such as RUC)
theta	Isentropic Coordinate
height	Height Coordinate
sigma	Sigma Coordinate (Native ARW or NMM)
none	No vertical coordinate (Surface-based) data

A.2.4 INITFH

What I Do: Defines the default initial forecast hour for the data set

Description:

The INITFH parameter is the initial (forecast) hour of the data set you wish to download. It is the default value but may be overridden from the command line with the **CYCLES** option.

For example, to use the 00 hour forecast from the data set as your 00 hour forecast:

INITFH = 00

And to use the 06 hour forecast from the data set as your 00 hour forecast.

INITFH = 06

A.2.5 FINLFH

What I Do: Defines the default final forecast hour for the data set

Description:

The FINLFH parameter is the final (forecast) hour of the data set you wish to download. It is the default value but may be overridden from the command line with the CYCLES option.

Example: FINLFH = 48

To use the 48 hour forecast from the data set as your last time for your boundary conditions.

Important: FINLFH - INITFH defines the length of your run unless otherwise overridden. See the CYCLES option or the "--length" ems_prep or ems_autorun option for more details.

A.2.6 FREQFH

What I Do: Defines the default boundary condition file frequency for the data set

Description:

The FREQFH parameter is the frequency, in hours, of the (forecast) files you wish to use between INITFH and FINLFH. This server as your boundary condition frequency and it is suggested that you use the highest frequency available (lowest value), which is usually 3-hourly (FREQFH = 03). Do not set this value lower than the frequency of the available data because bad stuff will happen.

Example: FREQFH = 03

A.2.7 CYCLES

What I Do: Define the cycle times on the data set

Description:

The CYCLES parameter defines the cycle hours (UTC) for which forecast files are generated from the operational model runs. The standard format for this parameter is:

CYCLES = CYCLE 1, CYCLE 2, …, CYCLE N,

Where each cycle time (UTC) is separated by a comma (,).

Example: **CYCLES = 00,06,12,18**

Important: The times listed in CYCLES are critical to ems_prep and ems_autorun working correctly as they identify the most recent data set available when executing real-time simulations. For example, if you want to download a 12 UTC run but "12" is not listed in the CYCLES setting, you will be out of luck (SOL). The ems_prep and ems_autorun routines will default to the most recent cycle time.

Alternatively, if you include cycle time for which no data set exists then you will have problems with your real-time downloads. Just don't do it.

There is a caveat though, please see the DELAY parameter for more information.

Note that the CYCLES setting can be overridden with the --cycle command line option.

Advanced Complex Stuff:

The CYCLES parameter in the gribinfo.conf file may be used to override the default INITFH, FINLFH, and FREQFH parameter values. If you do not want to use default settings for every model cycle, try using:

CYCLES = CYCLE[:INITFH:FINLFH:FREQFH]

Example: CYCLES = 00:24:36:06,06,12,18:12:36

Note that in the above example the individual cycle times are separated by a comma (,) and the INITFH, FINLFH, and FREQFH values are separated by a colon (:).

Interpretation:

From the 00Z Cycle run (00:24:36:06), obtain the 24 to 36 hour forecasts every 06 hours. Note these values override the INITFH, FINLFH, and FREQFH default values!

From the 06Z Cycle run (06) use the default values of INITFH, FINLFH, and FREQFH specified above.

From the 12Z Cycle run (12) use the default values of INITFH, FINLFH, and FREQFH specified above.

From the 18Z Cycle run (18:12:36), the 12 to 36 hour forecasts every FREQFH hours.

There are even a few other options when using the --cycle option in ems_prep and ems_autorun so you might want to review the information provided in Chapter 7.

A.2.8 DELAY

What I Do: Define the delay (hours) from the cycle time of the model run to when it is available.

Description:

The DELAY parameter defines number of hours, following a cycle time, before the GRIB files are available. In most cases, a lag exists from the time that the operational model is initialized to when the run is completed and the data are processed and ems_prep needs to know about this delay to operate correctly.

For example, if DELAY = 3, then ems_prep will not look for the 12Z cycle run files until after 15Z (12+3). The 06Z cycle would be used as the current cycle (default) between 9 and 15Z. This behavior can be overridden with the --nodelay option in ems_prep.

Example: DELAY = **05**

Note that is you set the value too low then you will be hitting the ftp server for data when the files are not available, which is not good.

A.2.9 SERVER-METHOD

What I Do: Lists local, http, and ftp the sources of the data set

Description:

The SERVER-METHOD specifies the method used to download the data files, the location of the files along with the filenames files joined by a colon (:).

SERVER-METHOD = **SERVER ID:/<path to data>/<filename convention>**

Important: The SERVER ID must have a corresponding IP/hostname defined in the SERVER section of the **uems/data/conf/ems_prep/prep_global.conf** configuration file.

Note that the following placeholders will be replaced with the appropriate values in ems_prep:

YYYY	4 digit year
YY	2 digit year
MM	2 digit month
DD	2 digit day
CC	Model cycle hour
FF	Forecast hour [0-99]
FFF	Forecast hour [100-999]
NN	2-digit Forecast minute

Also, there are occasions when one or more of the reserved place holder character strings is (are) part of the actual filename or path. An example of this issue would be "nam.tCCz.awip3dFFF" where the filename for a 12 hour forecast from a 00 UTC run is "nam.t00z.awip3dF12". Note that an "F" precedes the forecast hour, which is replaced by the "FF" (or "FFF") place holders in the SERVER-<METHOD> entry. This might lead to problems as ems_prep would not know which "FF" to replace and the "F" in the filename may be overridden.

To remedy this problem, a "\" may be placed **before each character** to be retained in the filename. In the above example, the file naming convention specified in the SERVER-<METHOD> would be "nam.tCCz.awip3d\FFF" and the leading "F" in "FFF" will be preserved in the filename.

METHOD

The METHOD indicates the method to use to acquire the data. Currently ftp, http, or nfs are supported indicated by SERVER-FTP, SERVER-HTTP, and SERVER-NFS respectively.

Examples:

SERVER-HTTP = **STRC:/data/YYYYMMDD/nam/grib.tCCz/nam.tCCz.awip3dFF.tm00.bz2**

SERVER-FTP = **NCEP:/pub/data/gfs/prod/gfs.YYYYMMDD/gfs.tCCz.pgrb2\FFF.bz2**

SERVER-NFS = **KIELBASA:/data/YYYYMMDD/grib/grib.tCCz/nam.tCCz.awip3dFF.tm00**

In the first example above, STRC is the ID of the http server and has a corresponding STRC = <hostname> entry in the prep_global.conf file. The files are located in the /data/YYYYMMDD/nam/grib.tCCz directory on the server and nam.tCCz.awip3dFF.tm00.bz2 is the naming convention, with space holders.

Note that ems_prep will automatically unpack ".gz" and ".bz2" files. If you are using a data source that is packed then make sure you include the appropriate suffix.

In the second example above, NCEP is the ID of the ftp server and has a corresponding NCEP = <hostname> entry in prep_global.conf. In order to use the SERVER-FTP (HTTP) option the data files must be available via ftp (http).

NFS USERS

In the SERVER-NFS example above, KIELBASA is the server ID of the system where the data reside and there is a corresponding KIELBASA = <hostname> entry in the prep_global.conf file. Unlike the FTP and HTTP options, either SERVER ID or actual hostname ([user@]<hostname>:) may be used to identify the server.

If a SERVER ID is used, it must be in **ALL CAPS** in both the "SERVER-NFS =" line and the **prep_global.conf** file. For example:

SERVER-NFS = KIELBASA:/data/YYYYMMDD/gfs/YYMMDDCC.gfs.tCCz.pgrb2fFF

Where in **prep_global.conf**:

KIELBASA = roz@kielbasa (Note the all capital letters for kielbasa)

And

SERVER-NFS = roz@kielbasa:/data/YYYYMMDD/gfs/YYMMDDCC.gfs.tCCz.pgrb2fFF

Are basically the same thing. So why then allow for both options? Specifying a server ID in the "SERVER-NFS =" line will allow users to specify a server when passing the --dset <dset set>:nfs:<server> flag to ems_prep. So if you had:

SERVER-NFS = SERVER_A:/data/YYYYMMDD/grib/grib.tCCz/nam.tCCz.awip3dFF.tm00.gz
SERVER-NFS = SERVER_B:/data/YYYYMMDD/grib/grib.tCCz/nam.tCCz.awip3dFF.tm00.gz

With SERVER_A and SERVER_B defined in **prep_global.conf**, then you can specify a server to access:

% ems_prep [other options**] --dset <data set>:nfs:server_b** (upper or lower case)

The default behavior with just "ems_prep --dset <data set>:nfs" will result in ems_prep looping through each of the servers listed (first SERVER_A and then SERVER_B).

Important! - The ems_prep routine uses secure copy (scp) to access the data on those servers identified by either the SERVER ID or actual hostname ([user@]<hostname>), so you MUST have passwordless SSH configured between the machine running ems_prep and the server.

But what is your data reside on the same machine as ems_prep and you don't want to use SCP? In that case set the SERVER ID to "LOCAL" or leave blank:

SERVER-NFS = LOCAL:/data/YYYYMMDD/grib/grib.tCCz/nam.tCCz.awip3dFF.tm00.gz
Or
SERVER-NFS = /data/YYYYMMDD/grib/grib.tCCz/nam.tCCz.awip3dFF.tm00.gz

In which case **ems_prep** will use the standard copy command (cp) to access the requested file from a locally-mounted partition.

Finally, if there is more than one server listed below and you do not specify a server or method, i.e., "% ems_prep --dset <data set>", then ems_prep will attempt to connect each (ftp, http, nfs) server listed until all the requested files have been downloaded.

So in summary:

% ems_prep --dset <data set>:<method>:<server>

Attempts to get <data set> via <method> from <server>

% ems_prep --dset <data set>:<method>

Attempts to get <data set> from all the <method> servers listed in the gribinfo.conf file.

% ems_prep --dset <data set>

Attempts to get <data set> via all the methods and servers listed in the <data set>_gribinfo.conf file.

A.2.10 LOCFIL

What I Do: Define local file naming convention of the data set

Description:

The LOCFIL parameter is file-naming convention to be used on the *local* system. This filename is usually the same as that on the remote server; but hey, you have the power. The primary purpose for this parameter is so filenames on the local machine do not change when failing over to a different remote server, which may not use an identical naming convention for the same data set. The filename uses the same YYYY, MM, DD, CC, FF, and TT placeholders listed in the SERVER-METHOD section.

Message about GRIB 2 <-> 1 CONVERSION

The UEMS will automatically convert between GRIB formats if requested by the user. This is necessary when using some data sets such as the NCEP tiles, which are available from NCEP in GRIB 2 format but must be converted to GRIB 1 format before processing. The ems_prep routine keys off the differences between the filenames on the remote and local systems. If the files on the remote server contain "grib2" or "grb2", or "gr2" in the filename but are missing in the local filename, then the files will be converted to GRIB 1 format. Conversely, if the remote file uses a GRIB 1 naming convention but a GRIB 2 name is used locally, then a GRIB 1 -> 2 conversion will occur.

Example: LOCFIL **= YYMMDDCC.gfs.tCCz.pgrb2foFF**

A.2.11 VTABLE and LVTABLE

What I Do: Defines the Vtable(s) to use for unpacking the GRIB file

Description:

The VTABLE parameter defines the Vtable.<MODEL ID> to use when processing the GRIB grids into the WPS intermediate format. All tables are located uems/data/conf/tables/vtables directory and define what fields to pull from the GRIB file for processing and initialization in your run. Note that Vtables are quasi-independent of the data set. The table just describes the available fields and not the navigation information so a single Vtable table may be used for multiple data sets.

The LVTABLE parameter defines the Vtable to use should this data set be accessed with the --lsm <data set> option, in which case the user likely wants a subset (near surface fields) of the fields available in the Vtable specified by LVTABLE. So, LVTABLE should point to a file that contains just the near surface fields. Both VTABLE and LVTABLE may be specified in the file.

Examples:

VTABLE = **Vtable.NAM**
LVTABLE = **Vtable.NAMLSM**

A.2.12 METGRID

What I Do: Specifies the version of METGRID.TBL for use with ems_prep

Description:

METGRID is used to specify an alternate METGRID.TBL file when running the metgrid routine as part of ems_prep.pl. If METGRID is not specified, then the default tables will be used which should be fine for most applications. There are some datasets such as the NNRP, that require an different file, which has been modified specifically for that dataset. If you with to modify an existing METGRID.TBL file for the dataset described by this file, it is recommended that you first make a copy of the original file, make your changes, and then specify the new filename with the METGRID parameter below. For example, setting:

METGRID = METGRID.TBL.NNRP.ARW

instructs ems_prep.pl to use METGRID.TBL.NNRP.ARW instead of the default METGRID.TBL.ARW file when doing horizontal interpolation from the donor to the computational grid.

All metgrid tables files reside in the uems/data/conf/tables/wps directory.

There is a single placeholder, "CORE", that may be used should there be a different metgrid table file for each model core supported by the UEMS. This option is a hold-over from the time when the NMM shared space with the ARW core in the EMS. Those days are long gone but for sentimental reasons the "CORE" placeholder remains and will be populated with the "ARW" string in the current release. No harm, no foul.

A.2.13 AGED

What I Do: Define the maximum age of data from To to use in initialization

Description:

AGED is the number of days prior to the initialization date/time of your simulation before this dataset is considered "too old" to be used. The UEMS will attempt obtain the most current dataset

file available, but should this file not be available, the UEMS will step backwards in time up to AGED days until another acceptable data file is found.

Only after acquiring a suitable date/time from the requested dataset has failed will the UEMS fail-over to an alternate dataset. If no alternate surface datasets are located, then the fields from the primary initialization dataset will be used.

Note that this parameter only applies to datasets containing static surface fields (--sfc)

A.2.14 TIMEVAR

What I Do: Whether LSM dataset is time-variant

Description:

Set TIMEVAR to Yes if the desired fields from this dataset are to be used as time-variant. This parameter only applies to datasets being used to provide land surface model (LSM) fields that are missing from the primary initialization datasets. Some dataset files, such as NNRP require another source for the sub-surface and surface fields necessary for initialization. Setting TIMEVAR = Yes instructs ems_prep to download and process the required fields for each boundary condition update time. The TIMEVAR parameter is only necessary for the few datasets being used to provide the missing fields. There currently there is no way to override this value from the command line. It is best that leave TIMEVAR commented out unless you know what you are doing, unlike the UEMS developer.

Appendix B: Running the UEMS benchmark simulation

Appendix Contents:

B.1 An introduction

The UEMS package includes a preconfigured WRF ARW core domain to test the installation and evaluate the performance of your system when running a simulation. All the initialization data are provided, and the configuration has been well-tested, so running the benchmark test should be a priority for new users or anyone else with the desire to taste the sweet nectar of success. Running the benchmark is straightforward provided you follow the guidance provided below.

The benchmark case consists of a primary (outermost) domain with two nested sub-domains. The purpose of this configuration is to provide an adequate measure of performance across a wide range of computer systems. If you are using a stand-alone workstation with a modest amount of physical memory (8Gb minimum), you should start by running only over the primary domain. This recommendation is because increasing the number of computational domains, and thus the number of grid points, with increase your system memory requirements.

BTW - If you don't know how much memory you have on your system, then just run the UEMS provided "sysinfo" utility:

```
% sysinfo
```

After you have familiarized yourself with the benchmark results, feel free expand your numerical weather prediction horizons by testing the sensitivity of the simulation to changes in the model physics or dynamics. The configuration files are found within the conf/ems_run directory. However, keep in mind that this exercise is intended to be a learning experience, so failure is always an option.

B.2 Benchmark case background

Note: Some of the information below has been "liberated" from various sources such as Wikipedia, NOAA/NWS Storm Prediction Center, and the Weather Channel. Any spelling or grammatical errors are those of the author.

The benchmark case is a 30-hour simulation of a major tornado outbreak that occurred from April 26-28 2011, across the southeast US. The simulation covers the period of 0600 UTC 27 through 12 UTC 28 April, during which time more than 300 tornadoes were reported.

A summary of the domain and run configuration is provided below.

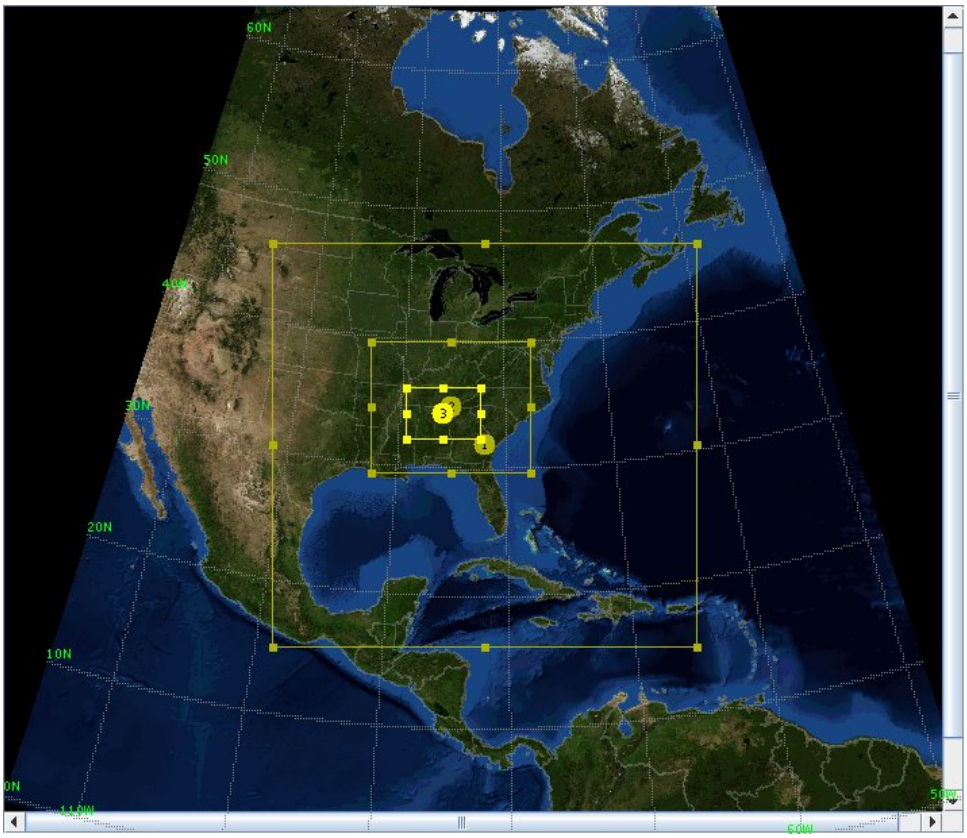
```
    Active Domains                 Domain 01              Domain 02                   Domain 03
*************************************************************************************************************************

     Domain & Run Information

       Domain Type               :  Limited Area           Limited Area                Limited Area
       Primary Time Step         :  90 Seconds             30 Seconds                  10 Seconds
       Grid dimensions (NX x NY) :  211 x 201              238 x 196                   334 x 232
       Vertical Layers (NZ)      :  45                     45                          45
       Grid Spacing              :  18.00km                6.00 km                     2.00 km
       Top of Model Atmosphere   :  50mb                   50mb                        50mb
       Parent Domain             :  NA                     Domain 01                   Domain 02

Run Information

       Start Date                :  2011 Apr 27 06:00 UTC       2011 Apr 27 06:00 UTC       2011 Apr 27 06:00 UTC
       End Date                  :  2011 Apr 28 12:00 UTC       2011 Apr 28 12:00 UTC       2011 Apr 28 12:00 UTC
       Simulation Length         :  30 Hours                    30 Hours                    30 Hours
       Boundry Update Freq       :  06 Hours
```

A complete listing of the configuration along with the physics and dynamics options can be found by using the "runinfo" utility - after you have run ems_prep (see below). To view this information for any simulation, run the following from the top of any run-time domain directory:

% runinfo [--domain 2[,3[,...]]]

Where the "--domain" flag is only necessary when requesting information about nested domains.

If you prefer looking at pictures, and who doesn't, a depiction of benchmark domain is provided by 27april2011/static/projection.jpg, but to save you the time it is also presented here:



Finally, before running the benchmark simulation, be sure to check the number of CPUs to be used in the conf/ems_run/run_ncpus.conf file and edit the values to reflect your system. And no, you cannot use any "virtual" processors made available when hyper-threading is turned on.

B.3 How to run the benchmark case

Running the benchmark case for each WRF core with the UEMS is straightforward:

Step I. From the util/benchmark/27april2011 directory, run the ems_prep

% ems_prep --benchmark

Or if you wish to include the first nested domain:

% ems_prep --benchmark --domain 2

Or if you are going for the "whole kielbasa":

% ems_prep --benchmark --domain 2,3 (no spaces between 2,3)

Note: If you request domains 2 and 3 when running ems_prep, you do not have include them when running the simulation; however, they must be included if you plan on using them.

Step II. Run ems_run to begin the simulation

% ems_run

Or if you wish to include the nested domain:

% ems_run --domain 2[,3]

Following completion of the simulation the output in netCDF format will be located in the wrfprd directory. You may use the "ncview" utility (provided with the EMS) to view the results:

% ncview <netCDF filename>

Or you can continue and post-process the files into a secondary format.

Step III. (Optional) Convert the output files and view the results

The netCDF simulation output files will be located in the wrfprd directory along with any files from the nested domains (if selected). You can convert the files to grib 2 format (and GrADS, GEMPAK) by running the ems_post routine.

% ems_post --grib (For GRIB2 files)

Or if you want to process the nested domain:

% ems_post --grib --domain 2 (or 3)

Note that you can currently process only one domain at a time. All processed data files will be located in the emsprd directory.

There are additional post-processing options available to you. Please see Chapter 9 or "ems_post --help" for the gory details.

Appendix C: 3D analysis and spectral nudging

Appendix Contents:

C.1 What is 3D analysis and spectral nudging?

Traditionally, a limited area simulation is run with the initialization and lateral boundary condition fields provided by a large-scale data set. The model is allowed to integrate forward in time with a limited amount of control provided by the regular boundary condition updates. This approach introduces a variety of potential error sources, and the solution can be sensitive to the domain size and time of year. Moreover, because errors within the model run typically increase with time, this approach has limited viability in long term forecasting and regional climate applications.

The use of 3-dimensional Analysis Nudging techniques has shown to reduce some of the issues associated with limited area models. Analysis nudging provides additional control over a simulation by forcing the model towards a predefined solution. This "solution" is typically provided by a series of analyses that are used to nudge the interior points within a computational domain during model integration. This method has been shown to benefit numerous applications of numerical models including the creation of 4-dimensional dynamically consistent data sets, the improvement of lateral boundary conditions within multi-scale simulations, dynamic initialization, and the downscaling of large-scale data sets for regional climate applications.

3-dimensional Analysis Nudging is not without its limitations, however. Most analyses used for nudging are of a lower resolution, both spatially and temporally than the computational domain to which they are being applied. Consequently, nudging a simulation towards a more course solution will degrade the effective resolution of the domain, which tends to smooth out features that may otherwise be resolved. One method of ameliorating this problem is to limit nudging to regions within the free atmosphere, i.e., above the model planetary boundary layer. This practice allows smaller-scale lower-tropospheric features such as a low-level jet or sea breeze circulation to evolve while not encumbered by non-physical nudging tendencies. In the case of nested simulations, where the child domain resolution is greater than that of the parent, it is advisable to turn off analysis nudging for the inner domains. An exception might be made if the grid spacing of the child domain is similar to that of the analysis data set, but that is typically not the case. Additionally, it is recommended that feedback is turned off when running nested simulations with nudging.

So, why provide you with this information? Well, integrated into UEMS is the ability to run simulations utilizing the 3-dimensional analysis nudging capabilities of the WRF. The 3D analysis

nudging is part of the four-dimensional data assimilation (FDDA) system in the WRF but may also be used to support multi-scale simulations that are initialized solely from a series of analyses or forecasts. And it's all yours!

If you want to learn more about the analysis nudging capability in the WRF, you are encouraged to review the PDF article in the uems/docs/wrf directory.

C.2 How to run a simulation with analysis nudging

Analysis nudging is typically conducted using a series of 3D analyses, such as the ERA reanalysis, the North America Regional Reanalysis (NARR), or the Climate Forecast System Reanalysis (CFSR) data sets. However, it is also possible to use forecasts from a previous run of the NAM or GFS, or even a series of 00-hour forecasts from an operational model. It's all up to you. The UEMS doesn't care (that much). You don't even need any additional files or fields as the same data you use to initialize your UEMS for a traditional run will be used for analysis nudging. Pretty sweet, Hey!

The following section steps you through the process of running a simulation with 3D analysis nudging. Should you have a problem anywhere in your journey to nudging satisfaction, the Council Headmaster for Universal Modeling Perfection (CHUMP) suggests that you "return to Step 1".

C.2.1 Nudging with ems_autorun.pl

Step 1. Edit the conf/ems_run/run_nudging.conf file

Nudging with ems_autorun requires that you configure the **run_nudging.conf** file prior to starting the simulation. This is because the "--nudge" flag will be automatically passed to ems_run without any arguments. Consequently, all the configuration options will be obtained from the file.

Step 2. Tell ems_autorun to include 3D analysis nudging in your simulation

Telling ems_autorun to use analysis nudging can be done by either:

a. Edit the **conf/ems_auto/ems_autorun.conf** files and set **NUDGING = Yes**

Or

b. Pass the "--nudge" flag to ems_autorun

```
% ems_autorun [any other options and flags] --nudge
```

C.2.2 Nudging from the command line with ems_prep.pl and ems_run.pl

Step 1. Run ems_prep with "--nudge"

To include analysis nudging in your simulation, you must tell the UEMS to process the initialization data for this purpose. This step is pretty trivial as all you need to do is include the "--nudge" flag when running ems_prep:

% ems_prep [all the other options and flags] --nudge

And all the magic gets taken care of behind the scenes.

Step 2. Run ems_run with "--nudge"

Following successful completion of ems_prep, you should be ready to proceed to ems_run. If you plan to include nested domains in your simulation, then you should have also passed the "--domains" flag to ems_prep, in which case you will see some additional initialization files located in the wpsprd/ directory that will be used for the analysis nudging. Take a look in the wpsprd directory if you care; otherwise, just tap out the following to start your simulation with 3D analysis nudging:

% ems_prep [all the other options and flags] --nudge

Again, all the magic will get taken care of behind the scenes.

C.3 Witness the power of nudging

Three nested simulations were conducted in which nudging was selectively applied to examine the impact on a 36-hour simulation to illustrate the utility or futility of 3D analysis nudging. The domain and model configuration was the same for all experiments, which included an 18km primary domain with a single 6km nest. The only difference between the simulations was that experiment "**A**" did not include any analysis nudging (turned off), experiment "**B**" employed analysis nudging only to the primary (outermost) domain and experiment "**C**" used nudging for both the primary and nested domains. Nudging within the PBL was turned off for all domains and experiments. Feedback between the primary and child domain was turned off (1-way nesting) for all experiments.

C.3.1 The Case: 14 - 15 December 1987

Each experiment was initialized from 0.5 degree, 6-hourly Climate Forecast System - Reanalysis data set (CFSR) beginning at 00 UTC 14 December 1987 and run for 36 hours. This case was selected due to the abundance of sexy dynamics, which is of interest to the UEMS Committee to Further Local Modeling through Sexy Dynamics Simulations (UCFLMSDS), and to demonstrate the quality of the CFSR data set for investigating historical events (No committee yet). This particular case included the generation and propagation of gravity waves across the US upper Midwest, as part of a period of explosive cyclogenesis, so there was a lot of stuff going on.

C.3.2 Who ya going to believe, the UEMS or your own lying eyes?

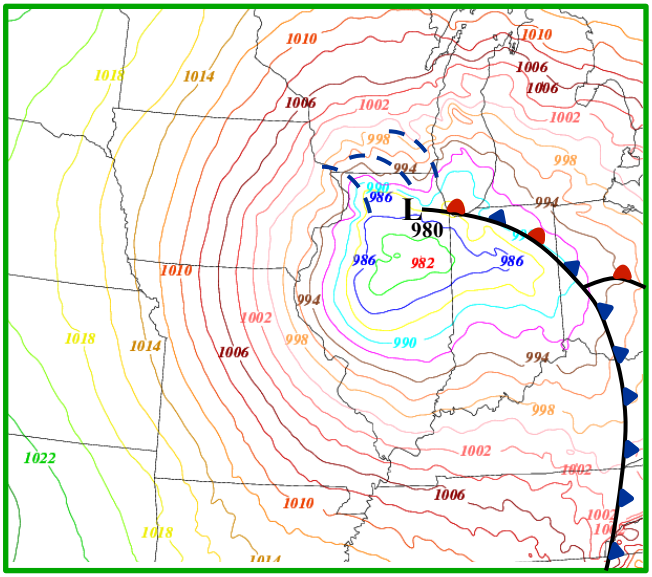
The limited results from the simulations are shown in this section to demonstrate the impact of 3D analysis nudging on a simulation. Since this is not a peer-reviewed article nothing has to be proven here. You must have faith in your documentation. If you don't trust the results, then feel free to run the case yourself, and it's easy to do with the UEMS!

The figure below depicts the 36-hour forecast of mean sea level pressure field from the three experiments along with an analysis of the observed surface low and frontal locations. Also included for comparison is the CFSR analysis for the same date and time. At 12 UTC 15 December, the observed system was occluded with a 980 hPa surface low located in northern Illinois and an accompanying front extending eastward across northern Indiana. At this time there was an observed gravity wave train propagating northeastward from northern Illinois into Wisconsin, which is crudely depicted in the three images by the dashed lines. Note that the exact placement and intensity of the observed surface system is encumbered by limited observations and the influence of the gravity waves in the surface pressure field. Here at UEMS World Headquarters, we (I) are (am) scientists and miracle workers, not graphic artists.

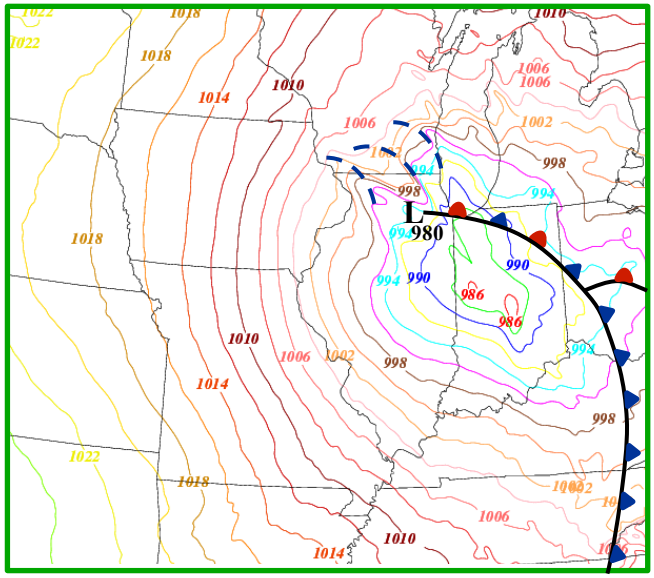
Figure A, from the experiment **without analysis nudging,** shows a primary low center located in north-central Illinois with the accompanying frontal system extending eastward, similar to that observed although located further to the south. Some evidence for the simulated gravity waves exists in the surface pressure; however, they are not well pronounced. The simulated central pressure of 982hPa is the closest to the analyzed 980hPa value in the three experiments.

Figure B, from the experiment with **analysis nudging turned ON for the primary domain only,** shows an ill-defined low center located further to the east of that depicted in experiment **A** and analyzed. The region of lowest pressures (986 hPa) is elongated with a northwest to southeast orientation extending from northern Illinois into Indiana. The surface pressure trough used to determine the location of the simulated frontal system is also not well defined but is still easily identified. This simulation depicts the most pronounced gravity wave signature in the surface pressure field among the three experiments. Multiple simulated waves are evident north and northwest of the low center. These waves formed during the simulation along the Missouri-Illinois border seven hours before the time depicted (~ 05 UTC) and propagated northeastwards while maintaining a coherent structure. Again, you must believe the documentation.

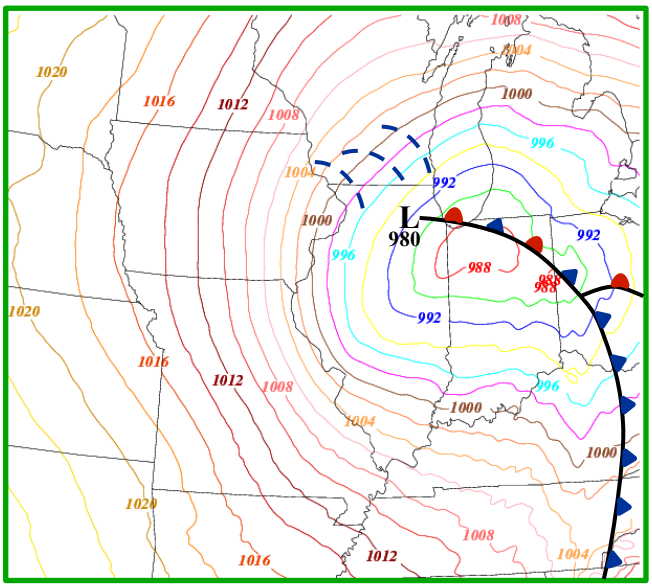
Figure C shows the results of the experiment in which **analysis nudging was employed for all domains.** Remember that unlike experiment **B**, the nested domain in experiment **C** is also being nudged towards the lower resolution CFSR analysis valid at 15 UTC 15 December 1987 (**Figure D**). Even though nudging was turned off within the PBL for both domains, the effects of the nudging in the free atmosphere are evident in the surface pressure field. The simulated central pressure of 987hPa is the furthest from the analyzed 980hPa value in the three experiments. The low center is positioned further to the east in northwest Indiana with the frontal system extending eastward. Overall, the pressure field appears less noisy, and evidence of any gravity wave activity is missing.



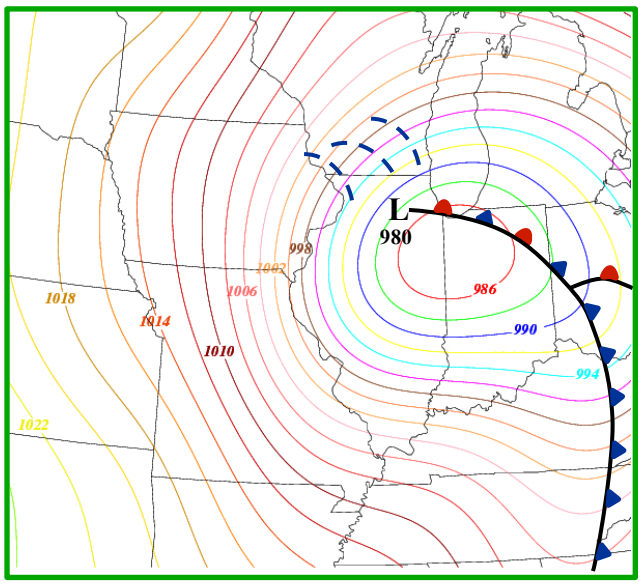
(A) No Analysis Nudging: Domain 2 mean sea level pressure (hPa) valid at 12 UTC 15 December 1987



(B) Analysis Nudging Domain 1 Only: Domain 2 mean sea level pressure (hPa) valid at 12 UTC 15 December 1987



(C) Analysis Nudging Domains 1 & 2: Domain 2 mean sea level pressure (hPa) valid at 12 UTC 15 December 1987



(D) CFSR Analysis: Mean sea level pressure (hPa) valid at 12 UTC 15 December 1987

Appendix D: A Summary of precipitation fields in the UEMS

Appendix Contents:

D.1 Sometimes you get stuff you never knew you deserved

WRF simulation output from the UEMS includes many fields that are not part of the standard NCAR package. These additional fields include precipitation type, sensible weather, and column-based products, all of which were handcrafted to enhance your UEMS experience. A listing of all data fields written to a WRF primary simulation output file (wrfout_*) may be viewed by running:

```
% rdwrfnc -m <wrfout file>
```

Within the information written to the screen you should see the following fields UEMS included fields listed:

TACC_<type>	Simulation Total Accumulation of <type> (mm)
INST_<type>	Instantaneous <type> precipitation rate (mm s-1)
MAX_<type>	Maximum <type> precipitation rate between output times (mm s-1)

Descriptions of the individual precipitation types are provided in the next section.

D.2 Accumulated precipitation fields - Quantities accumulated over the entire simulation

The UEMS-provided precipitation fields are calculated at each timestep based on the fraction of precipitation predicted for each type; rain, freezing rain, ice, snow, graupel, and hail. Additionally, the UEMS leverages the information from the microphysics scheme to determine how to partition the total precipitation (MP+CU+SHCU) over that timestep. The primary assumption in these calculations is that the ratio of each hydrometeor type to total precipitation from the microphysics scheme can be applied to the total precipitation over each timestep. Some additional calculations are made to diagnose freezing rain, liquid equivalent snowfall, and snow depth.

While all the fields listed below can be found in the UEMS output, whether or not they contain non-zero values depends upon your choice of model physics. If a scheme does not support a specific hydrometeor type, such as hail, then the field will contain all zeros.

Below is a summary if each precipitation field output from the UEMS. Fields that are part of the standard NCAR package are indicated by a red asterisk (*).

D.2.1 RAINC*, RAINSH*, and RAINNC*

Total accumulated precipitation (kg m-2 or mm; liquid water equivalent)

The **RAINC**, **RAINSH**, and **RAINNC** fields contain the total accumulated precipitation from the Convective (CU), Shallow Convective (SHCU), and Microphysics (MP) schemes respectively. Think "**RAIN C**umulus, **RAIN Sh**allow, and **RAIN N**on-**C**onvective".

Each field contains the liquid water equivalent amount from that scheme regardless of whether it falls as liquid or frozen stuff. Consequently, the total accumulated precipitation is simply the sum of these three fields:

Total accumulated precipitation = **RAINC** + **RAINSH** + **RAINNC**

If a convective or shallow convective scheme is not used during a simulation, then the associated field will contain all zeros. Fields will also contain zeros at the start of a simulation (TØ output file), unless the data are from a child domain that began integration after the start of the parent. In that case, any accumulated precipitation from the three schemes will be interpolated to the child domain at TØ, regardless of the physics scheme used in the nested simulation. Consequently, the TØ output from a child may contain non-zero accumulate precipitation amounts.

D.2.2 SNOWNC*

Total accumulated snow+ice from the microphysics scheme (kg m-2 or mm; liquid water equivalent)

The **SNOWNC** ("**SNOW N**on-**C**onvective") field contains the simulation accumulated snow+ice amounts as predicted by the microphysics scheme. This field should not be confused with **ACSNOW**, even if you are running without a cumulus scheme. It also should not be used as a surrogate for liquid equivalent snowfall since it does not include the accumulated graupel, which some microphysics schemes favor.

In the UEMS, some of the WRF microphysics routines have been modified for consistency so that **SNOWNC** = snow+ice only, which is not always true in the NCAR WRF release.

As with the other **NC** fields, **SNOWNC** will contain zeros at the beginning of a simulation (TØ wrfout file) The exception is when the data are from a child domain that begins integration after the start of the parent. In this case, any accumulated precipitation from the three schemes will be interpolated to the child domain at TØ, regardless of the physics scheme is used in the nested simulation. Consequently, the TØ output from a child may contain non-zero accumulate precipitation amounts.

D.2.3 GRAUPELNC*

Total accumulated graupel from the microphysics scheme (kg m-2 or mm; liquid water equivalent)

The **GRAUPELNC** ("**GRAUPEL N**on-**C**onvective") field contains the simulation accumulated graupel amounts as predicted by the microphysics scheme. If your choice of microphysics does not support graupel, then **GRAUPELNC** will contain all zeros, and you will like it.

Remember that all the **NC** fields, including **GRAUPELNC**, contain the information from the microphysics scheme only. If you are also using a cumulus or shallow cumulus scheme, then those fields will only contain total liquid equivalent precipitation.

D.2.4 HAILNC*

Total accumulated hail from the microphysics scheme (kg m-2 or mm; liquid water equivalent)

The **HAILNC** ("**HAIL N**on-**C**onvective") field contains the simulation accumulated hail amounts a predicted by the microphysics scheme. If your choice of microphysics does not support hail, then **HAILNC** will contain all zeros, and you will like it (again).

Remember that all the **NC** fields, including **HAILNC**, contain the information from the microphysics scheme only. If you are also using a cumulus or shallow cumulus scheme, then those fields will only contain total liquid equivalent precipitation.

Depending upon which way you lean, you may include accumulated **HAILNC** in your total liquid equivalent snowfall calculation. After all, it's just frozen stuff.

D.2.5 SR*

Snow ratio from the microphysics scheme (Non-Dimensional)

The Snow Ratio (**SR**) is important in that it is used in the calculation of some NCAR WRF accumulated snow fields. The **SR** field is the ratio of liquid equivalent frozen precipitation to total precipitation at each timestep from the microphysics scheme, i.e.,

SR = Frozen H2O / Total H2O (Liquid Equivalent)

The **SR** value is calculated within each microphysics scheme with the only difference between the hydrometeor types that compose the "Frozen Stuff." For example, if a scheme predicts

rain, ice, snow, and graupel, the **SR** is calculated by:

Frozen Stuff **All Stuff**

SR = (ice+snow+graupel)/(ice+snow+graupel+rain) At each MP timestep

The value of **SR** is not sensitive to a particular microphysics scheme's proclivity for favoring one frozen precipitation type over another; however, it will obviously be impacted by schemes that have a bias toward liquid vs. frozen precipitation.

Note that **SR** is determined within the microphysics and does not care about what is falling out of the cumulus or shallow cumulus schemes. This nugget of information may be important because **SR** values are used to compute **ACSNOW**.

For obvious reasons the value of **SR** equals zero for rain-only MP schemes (Kessler)

D.2.6 ACSNOW*

Total accumulated snowfall since simulation start (kg m-2 or mm; liquid water equivalent)

The **ACSNOW** field contains an estimate (one of a few) of the liquid equivalent snowfall since the start of the simulation. *Unlike* **SNOWNC**, **ACSNOW** *includes the contributions from the convective and shallow convective schemes*. While it does contribute to the snowpack fields, **SNOW** & **SNOWH**, it is not affected by snowmelt.

Additions to **ACSNOW** are calculated at each time step by using the **SR** value at a grid point. If the value of **SR** > 0.50, i.e., 50% or greater frozen stuff from microphysics scheme at that timestep, then ALL the precipitation is assumed to be snow; otherwise, it is rain.

So,

ACSNOW = **ACSNOW** + (**RAINC** + **RAINSH** + **RAINNC**) If **SR** > 0.50

This bulk method of calculating snowfall (all or nothing) is called into question when **SR** values remain near 0.50 for an extended period, but hey, what else are you going to do?

D.2.7 SNOW*

Total snow on ground or snowpack (kg m-2 or mm; liquid water equivalent)

The NCAR WRF **SNOW** field contains the amount of liquid equivalent snow water on the ground during the simulation. If you initialize your simulation with data containing a snow field (**SNOW** in the WPS output), then those data will be included in the simulation **SNOW** field values at the initial time (TØ).

Additions to the **SNOW** field are calculated in the LSM scheme and any changes during a run, i.e., increase or decrease from **SNOW**@TØ, result from both new snowfall and snowmelt.

D.2.8 SNOWH*

Depth (physical) of snowpack or snow on the ground (meters)

The **SNOWH** field contains the physical depth of the snowpack during a simulation. The

calculation depended upon the value of **SNOWH** during the preceding timestep plus any new snowfall:

SNOWH = **SNOWH** + **Depth of New Snow**

The actual depth of new snow (**Depth of New Snow**) added to the snowpack at each timestep is calculated within the LSM. For the NOAH LSM, this value is determined from the estimated density of new snowfall as a function of near-surface temperature, where:

Depth of New Snow = New Snow (Liq.Equiv.)/Snow Density

Where,

Snow Density =

0.05	**When T < -15C (20:1 SLR)**
0.05 + 0.0017*(T+15.)1.5**	**When T >= -15C**

The value of **SNOWH** is further modified within the LSM to account for additional effects such as melting.

D.2.9 ACSNOM

Total accumulated snowmelt since simulation start (kg m-2 or mm; liquid water equivalent)

The **ACSNOM** field contains the accumulated snowmelt as diagnosed by the LSM scheme. Within the LSM the amount of melted liquid water is calculated and used to update the snowpack (**SNOW** & **SNOWH**). There is nothing much to see here unless you are into that sort of thing.

D.2.10 SNOWC*

Snow Coverage (Non-Dim; Value range from 0 to 1.)

The **SNOWC** is the ratio of surface area covered by snow within a grid box. The calculation is made within the LSM scheme. For the NOAH LSM, the value is dependent upon the liquid snow equivalent of the snowpack (**SNOW**) and the vegetation type. That is all you need to know about **SNOWC.**

D.2.11 TACC_PRECIP

UEMS total accumulated precipitation (mm; liquid water equivalent)

The **TACC_PRECIP** field, which is provided to you by the singular folk at UEMS world headquarters, contains the total accumulated precipitation from the convection, shallow convection, and microphysics schemes. It is similar to the sum of the **RAINC**, **RAINSH**, and **RAINNC** fields; however, precipitation from a parent domain is not included in the field at the start of a child domain simulation (TØ) when the child integration begins after the parent.

So, TACC_PRECIP(@TØ) = Ø. Always.

The Total Accumulation, Instantaneous and Maximum Precipitation Rate fields

D.2.12 TACC_RAIN

UEMS total accumulated rainfall (mm; liquid water equivalent)

The **TACC_RAIN** field contains information about the total rainfall from the convection, shallow convection, and microphysics schemes. Only precipitation that remains wet after hitting the ground is used, so freezing rain is not included, as that would violate the UEMS law of wet stuff.

D.2.13 TACC_ZRAIN

UEMS total accumulated freezing rainfall (mm; liquid water equivalent)

The wet stuff that freezes contact like the chocolate shell on your ice cream - Yummy!

The **TACC_ZRAIN** field contains information about liquid precipitation (rain) that freezes when it hits the ground. After an initial rain fraction is calculated, a test is applied to determine whether the amount of precipitation should be categorized as liquid or freezing rain. This step is accomplished by taking the mean of the skin and 1st model layer temperatures. If the value is less than 0C the entire precipitation amount is categorized as freezing rain; otherwise, it is just rain.

D.2.14 TACC_SNICE

UEMS total accumulated ice (mm; liquid water equivalent)

The **TACC_SNICE** field contains information about the snow+ice precipitation types as determined by the microphysics scheme. It's important to understand that this field is calculated directly from the snow/ice quantity predicted by the MP scheme at each timestep (SNOWNCV). These ice fields are not necessarily a surrogate for snowfall (**TACC_SNOW** and **TACC_SNOWD**) because they do not include graupel. If the MP scheme you are using doesn't support graupel, then please ignore the previous sentence.

D.2.15 TACC_GRAUPEL

UEMS total accumulated graupel (mm; liquid water equivalent)

The **TACC_GRAUPEL** field contains information about the fraction of accumulated graupel in the total accumulated precipitation. If the MP scheme does not support the graupel precipitation type, this field will contain zeros.

D.2.16 TACC_HAIL

UEMS total accumulated hail (mm; liquid water equivalent)

The **TACC_HAIL** field contains information about the fraction of accumulated hail in the total accumulated precipitation. If the MP scheme does not support the hail precipitation type, this field will contain zeros.

D.2.17 TACC_SNOW

UEMS total accumulated snowfall - All the white stuff (mm; liquid water equivalent)

The **TACC_SNOW** field contains the fraction of snow+ice+graupel from the MP scheme applied to the total precipitation. This field is what you should use when looking at simulated snowfall amounts because it includes the major frozen hydrometeor fractions as predicted by the MP scheme (Hail is excluded).

So, **TACC_SNOW** = **TACC_SNICE** + **TACC_GRAUPEL**

The **TACC_SNOW** field differs from **ACSNOW** in that while the calculation of **ACSNOW** uses the snow ratio (**SR**) field to determine frozen or liquid precipitation type, calculations of the **TACC_** fields assume that mixed precipitation can exist. The reality is that most microphysics schemes do a good job at not spreading the total predicted precipitation over many types on a regular basis.

D.2.18 TACC_SNOWD

UEMS measured depth of snowfall (frozen) from the start of simulation (mm)

The **TACC_SNOWD** field attempts to predict the total physical depth of snowfall since the start of the simulation, just like you were to measure it with your "Hello Kitty snow depth measurement stick." There are two methods used for this calculation depending on your choice of MP scheme:

Method #1 - The Milbrandt-Yau method

Calculation of snowfall depth using this method only applies if you use the double-moment Milbrandt-Yau microphysics scheme (9). Buried within the bowels of the scheme is the code necessary to calculate the ratio between the volume of frozen precipitation (ice+snow+graupel) to the volume of total precipitation. Unlike a similar mass based calculation, using volume allows for the explicit diagnosis of the snow liquid ratio (SLR) at the surface, which may then be applied to the liquid equivalent snowfall to get the predicted snow depth.

While the calculation is correct, it does not mean the field is without caveats:

a. The scheme makes the spherical hydrometeor assumption, which is fine unless the snow consists of needles and other non-spherical snow shapes.

b. There is no accounting for the effects of surface phenomena such as melting, compacting, settling, drifting, etc. Consequently, the Milbrandt-Yau **TACC_SNOWD** field will likely be an over-prediction of the same liquid snowfall amounts if you measured it yourself with the well-calibrated Kitty stick.

c. A UEMS-implemented attempt to account for the break-up of large snowflakes during high wind conditions is included. I like to think of it as a sausage-making analog to snowfall prediction.

Method #2 - The LMS hacked method

Because I didn't want the other MP schemes to feel left out of the simulated snow depth party, a separate method of estimating the depth for all those microphysics schemes not named Milbrandt-Yau. This method was liberated from the NOAH LSM scheme (See **SNOWH**) and is applied to the predicted liquid equivalent snow amount at each timestep:

TACC_SNOW = TACC_SNOW + New Snow (Liq.Equiv.)/Snow Density

Where,

Snow Density =

0.05	**When T < -15C (20:1 SLR)**
0.05 + 0.0017*(T+15.)1.5**	**When T >= -15C**

The field is different from the LSM derived **SNOWH** values in that it uses the **TACC_SNOW** increment at each time step rather than applying the Snow Ratio (**SR**) threshold test. It also does not include the effects of melting, compacting, settling, drifting, or shoveling. A yellow snow coverage field may be added in a future release.

D.3 Instantaneous precipitation fields - Valid at the time of output file date/time stamp

All data units are liquid-equivalent mm s^{-1} unless otherwise noted.

INST_CURATE	Instantaneous precipitation rate from the convective scheme
INST_NCRATE	Instantaneous precipitation rate from the microphysics scheme
INST_PRATE	Instantaneous total precipitation rate
INST_RRATE	Instantaneous rainfall rate
INST_ZRATE	Instantaneous freezing rainfall rate
INST_IRATE	Instantaneous snow+ice precipitation rate
INST_GRATE	Instantaneous graupel precipitation rate
INST_HRATE	Instantaneous hail precipitation rate
INST_SRATE	Instantaneous snow (snow+ice+graupel) rate
INST_SFRATE	Instantaneous snowfall (depth) rate

D.4 Period maximum precipitation fields - Valid over the period from previous output file date/time stamp

All data units are liquid-equivalent mm s^{-1} unless otherwise noted.

MAX_CURATE	Period Maximum precipitation rate from the convective scheme
MAX _NCRATE	Period Maximum precipitation rate from the microphysics scheme
MAX _PRATE	Period Maximum total precipitation rate
MAX _RRATE	Period Maximum rainfall rate
MAX _ZRATE	Period Maximum freezing rainfall rate
MAX _IRATE	Period Maximum snow+ice precipitation rate
MAX _GRATE	Period Maximum graupel precipitation rate
MAX_HRATE	Period Maximum hail precipitation rate
MAX _SRATE	Period Maximum snow (snow+ice+graupel) rate
MAX _SFRATE	Period Maximum snowfall (depth) rate