

EMS Version 3



A Nearly Complete Guide to the WRF EMS



**National Weather Service/Training Division
Forecast Decision Training Branch**

EMS Version 3

Weather Research & Forecasting



A Nearly Complete Guide to the WRF EMS



National Weather Service/Training Division
Forecast Decision Training Branch



Table of Contents –Millions of Fingers and Toes Saved from Paper Cuts

Table of Contents for the Nearly Complete Guide to the WRF EMS	i
Acknowledgements	v
Preface	vii
Chapter 1: Introduction to the WRF EMS	1
1.1 WRF EMS: What is it?.....	1-1
1.2 Why should you care?.....	1-1
1.3 Summary of WRF EMS features	1-2
1.4 How much computer power is necessary to run the EMS?	1-3
1.5 Is support for the WRF EMS available?	1-3
1.6 What if I have a brilliant idea that must be incorporated into the EMS?	1-3
Chapter 2: Installation of the WRF EMS	2
2.1 Introduction -- Because you have to start somewhere	2-1
2.2 System Requirements for installing the EMS	2-1
2.3 Installing the WRF EMS.....	2-2
2.3.1 The basics - The stuff you should know	2-2
2.3.2 What happens during the installation process?	2-2
2.3.3 Installing from DVD	2-3
2.3.4 Installing from the internet	2-4
2.3.5 A local network installation	2-4
2.3.6 Making sure your installation was a success.....	2-5
2.3.7 Additional, potentially useless, installation options	2-6
2.4 Updating your WRF EMS with passion.....	2-8
2.4.1 More basics - The stuff you sorta gotta know.....	2-8
2.4.2 More potentially useless updating options.....	2-8
2.5 Additional information that may only be of limited use	2-10
2.6 Target binaries and you	2-11
Chapter 3: A self-guided tour through the WRF EMS	3
3.1 WRF EMS: What is in this thing?	3-1
3.2 The WRF EMS Tour: A view from the tippy top	3-1
3.3 The all-important EMS.cshrc file	3-2
3.4 All the environment variables you love to hate	3-4
3.5 WRF EMS run-time routines	3-4

Table of Contents –Millions of Fingers and Toes Saved from Paper Cuts

3.6	The many utilities you would have missed had they not been listed here.....	3-6
3.6.1	Some additional WRF EMS utilities.....	3-6
3.6.2	Non EMS-developed tools and utilities.....	3-7
3.7	A WRF EMS flow diagram	3-8
Chapter 4:	Just Getting Started? Read this Chapter!	4
4.1	Just the basics.....	4-1
4.2	Help! Help! - How Do I Run this WRF EMS?.....	4-1
4.3	Additional information you might want to know.....	4-4
Chapter 5:	Just Because You Can - Mastering Your Model Domain	5
5.1	Just the mastering basics	5-1
5.2	Be the “Wizard of Your Domain” with the Domain Wizard.....	5-2
5.2.1	Starting the Domain Wizard.....	5-2
5.2.2	The “Choose Domain” window.....	5-4
5.2.3	The “Horizontal Editor” window.....	5-5
5.2.4	Defining your ARW and NMM core primary domain.....	5-7
5.2.5	Creating nested domains for the NMM and ARW core	5-11
5.2.6	Creating domain localizations	5-15
5.2.7	Visualizing the terrestrial data	5-16
5.3	Domain creation from the command line.....	5-18
5.3.1	Creating a new domain with <code>ems_domains.pl</code>	5-18
5.3.2	Manually configuring the navigation for a new domain.....	5-18
5.3.3	Manually localizing your new domain.....	5-19
Chapter 6:	What’s in each WRF EMS run-time domain directory?	6
6.1	What is a “run-time directory”?	6-1
6.2	A tour of a run-time directory	6-1
6.2.1	The run-time routines.....	6-1
6.2.2	The domain sub directories	6-2
Chapter 7:	Meet the WRF EMS <code>ems_prep.pl</code> routine.....	7
7.1	“What is this <code>ems_prep.pl</code> thing”?	7-1
7.2	“How do I make <code>ems_prep</code> work for me”?.....	7-1
7.3	Some handy dandy <code>ems_prep</code> option flags.....	7-2
7.4	Meet your <code>ems_prep</code> configuration files.....	7-2
7.4.1	The one and only <code>prep_global.conf</code>	7-2
7.4.2	The <code><data set>_gribinfo.conf</code> files	7-3

Table of Contents –Millions of Fingers and Toes Saved from Paper Cuts

7.4.3	Meet the Vtable files	7-3
7.5	What are “personal tiles” data sets?.....	7-3
7.6	All the <code>ems_prep</code> command line options you want to know and love	7-4
7.7	Just a few <code>ems_prep</code> examples	7-18
7.7.1	The very basics with <code>--dset</code>	7-18
7.7.2	Using the <code>--cycle</code> option with emotion	7-19
7.7.3	Witness the power of <code>--length</code>	7-20
7.7.4	An example using the <code>--date</code> option	7-20
7.7.5	Working with multiple initialization data sets.....	7-20
7.7.6	“What do we want? NARR and NNRP data! When do we want it? Now!”	7-21
Chapter 8:	Meet the WRF EMS <code>ems_run.pl</code> routine	8
8.1	A taste of the <code>ems_run.pl</code> life	8-1
8.2	Meet your <code>ems_run</code> configuration files.....	8-1
8.3	Managing output from the model.....	8-4
8.4	Running <code>ems_run</code> from the command line.....	8-4
8.5	The <code>ems_run</code> command line options.....	8-5
Chapter 9:	Meet the WRF EMS <code>ems_post.pl</code> routine	9
9.1	A taste of the <code>ems_post.pl</code> life	9-1
9.2	How do I process thee? Let me count the ways	9-1
9.3	Welcome to your <code>ems_post</code> configuration files.....	9-2
9.4	The <code>ems_post</code> command line options	9-3
9.4.1	General <code>ems_post</code> options and flags	9-4
9.4.2	Options for the creation of GRIB files	9-5
9.4.3	Options for the creation of GEMPAK grid files.....	9-7
9.4.4	Options for the creation of GrADS grid files	9-7
9.4.5	Options for the creation of BUFR, GEMPAK, and BUFKIT sounding files.....	9-8
9.5	Managing the fields in your WRF EMS GRIB files.....	9-9
9.5.1	Controlling GRIB file contents for different domains.....	9-9
9.5.2	Deciphering the <code>wrfpost</code> control file.....	9-10
9.6	A message regarding the NMM core to GRIB processing	9-11
Chapter 10:	Meet the WRF EMS <code>ems_autorun.pl</code> routine	10
10.1	Living the <code>ems_autorun.pl</code> life.....	10-1
10.2	Meet your <code>ems_autorun</code> master	10-1
10.3	Setting up the EMS for real-time forecasting applications	10-1

Table of Contents –Millions of Fingers and Toes Saved from Paper Cuts

10.4	Running <code>ems_autorun</code> from the command line	10-3
10.5	The <code>ems_autorun</code> command line options	10-3
10.6	Concurrent post processing with EMS <code>autopost</code>	10-16
Appendix A: Adding or Modifying Initialization Data Sets in the WRF EMS		A
A.1	Welcome to the <code>gribinfo.conf</code> files!	A-1
A.2	Anatomy of a <code>gribinfo.conf</code> file	A-1
Appendix B: Running the WRF EMS Benchmark Cases		B
B.1	A bit of an introduction is in order	B-1
B.2	Benchmark case background information	B-1
B.3	How to run the benchmark case	B-1
B.4	What can I do with the benchmark results?	B-2
Appendix C: A Description of the EMS <code>wrfpost</code> output fields		C
C.1	A bit of an introduction is in order	C-1
C.2	A list of fields available from the EMS <code>wrfpost</code> routine	C-1
Appendix D: Getting your WRF EMS output displayed in AWIPS		D
D.1	Ingesting GRIB 1 files in AWIPS	D-1
D.2	Ingesting GRIB 2 files in AWIPS	D-9
Appendix E: WRF EMS Exercises		E
E	Information regarding the WRF EMS workshop exercises	E-1
E.1	Some information for the caretaker of the WRF EMS	E-1
E.2	System Requirements	E-1
E.3	Data Sets Provided for the Exercises	E-1
E.4	A brief summary of the experiments	E-2
E.5	A brief summary of the experiments	E-2
E.6	Exercise #1 - Installing and Benchmarking the WRF EMS	E1-1
E.7	Exercise #2 - Introduction to the WRF EMS	E2-1
E.8	Exercise #3 - Sensitivity of a Simulation to Model Configuration	E3-1
E.9	Exercise #4 - Running a Nested Simulation	E4-1
E.10	Exercise #5 - Real-time NWP with the WRF EMS	E5-1

Acknowledgements – Returning some love to those who have contributed

Greetings to all who dare to read!

This is a **partial list** of WRF EMS contributors, only because I can't possibly recall everyone who has provided suggestions and assistance over the years. If you are reading this and feel you are being slighted, then please contact me and I'll add you to the list. If you are embarrassed to ask for inclusion, just send the request through a surrogate. A note from your doctor will suffice as well. You will really be noticed and appreciated if you write the request on the side of a Pinarello Dogma (Italian users), Storck Fasenario 0.7 (German users), Look 595 (French users), or a Trek Madone 6.5 (everyone else) road bike frameset (size 54). No harm in asking.

Contributors in to particular order other than alphabetical

National Weather Service Field Offices

Brandt Maxwell	WFO SGX	Matt Foster	WFO OUN
Charlie Paxton	WFO TBW	Pablo Santos	WFO MIA
Christopher Mello	WFO CLE	Thomas Hultquist	WFO MPX
Dan Leins	WFO CLE	Warren Snyder	WFO ALY
Jeff Medlin	WFO MOB		

National Center for Environmental Prediction (NCEP)

Geoff DiMego Wesley Ebisuzaki Matt Pyle

Other Individuals whose contributions and patience I greatly appreciate:

Alan Fox	Fox Weather
Ben Baranowski	NWS/WDTB
Brian Hoeth	NASA
Carlo Colarieti Tosti	Italy
David Smart	University College London
Emanuele Rizzi	Italy
Jamie Wolff	NCAR/DTC
Jan Horak	Czech Republic
Jeff Smith	NOAA/ESRL
Jonathan Case	NASA
Leonard Montenegro	
Paula McCaslin	NOAA/ESRL
Scott Dembek	NASA
Tony Mostek	NWS/FDTB

And finally:

The wonderfully enthusiastic, devastatingly intelligent, and exceptionally good-looking yet suspiciously anonymous people of "wrfhelp"

And to those users who have sent kind words of love and encouragement wrapped in email that read "This %\$&%!@ thing doesn't work!!!". They too deserve some acknowledgement, but just not here.

Preface – For those users looking for “reading materials”

Caveat Lector!

The Weather Research and Forecasting (WRF) Environmental Modeling System (EMS) is a complete and relatively easy to use state-of-the-science numerical weather prediction (NWP) modeling package. Nonetheless, running any NWP model still requires a basic understanding of meteorological and computer science. Even the most NWP-savvy users do encounter problems, some of which are their own fault whether they want to admit it or not. Occasionally it is the fault of others, but they’re not going to admit it either. Other times, stuff just happens for completely unexplained reasons that are outside the control of anybody or anything. This is real science, and while it ain’t perfect, it’s getting very close. Consequently, the National Oceanic and Atmospheric Administration (NOAA), National Weather Service (NWS), Forecast Decision Training Branch (FDTB), and the Science Operations Officer (SOO), Science and Training Resource Center (SOO/STRC) will take no responsibility for the accuracy of your forecasts, even if you ask nicely.

In addition, numerical weather prediction can be dangerous regardless of the forecast. Understand that when running the WRF EMS, stuff can happen and things may break that can’t always be fixed by the lone support person. That’s just the way it is. Life is sometimes unfair. But fear not, when you have the WRF EMS, your beer glass is always half full.

When asking for help from WRF EMS support, be kind and patient. Platitudes and offers of baked goods sometimes work, but not always. If you are concerned then stop reading now, close this guide, wash your hands, and run away; otherwise, if you want the thrill and adventure of running your own numerical weather prediction system, then carry on citizen modeler!

This document provides limited guidance for using the WRF EMS. While nearly every attempt has been made to ensure that this document is complete, it should be noted that this prose was written while under the influence of sleep deprivation, a highly precocious 6-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, exist.

Any questions or suggestions for improvement should be sent directly to the author, just because he cares.

And as the WRF EMS marketing and campaign manager reminds us to say:

“Think Globally, Model Locally”

Second printing Release 3.1.1, February 2010
Author: Robert Rozumalski – NOAA/NWS/FDTB/STRC
SOO/STRC WRF EMS Website: <http://strc.comet.ucar.edu/wrfems>

Contact information:

Robert A. Rozumalski, PhD
NOAA/NWS National SOO Science and Training Resource Coordinator
Forecast Decision Training Branch
COMET/UCAR PO Box 3000
Boulder, CO 80307-3000
Phone: 303.497.8356

Robert.Rozumalski@noaa.gov

Chapter 1: Introduction to the WRF EMS

Chapter Contents

- 1.1 WRF EMS: What is it?**
- 1.2 Why should you care?**
- 1.3 Summary of WRF EMS features**
- 1.4 How much computer power is necessary to run the EMS?**
- 1.5 Is support for the WRF EMS available?**
- 1.6 What if I have a brilliant idea that must be incorporated into the EMS?**

1.1 The WRF EMS: What is it?

The Weather Research and Forecast Environmental Modeling System (WRF EMS) is a complete, full-physics, state-of-the-science numerical weather prediction (NWP) package that incorporates dynamical cores from both the National Center for Atmospheric Research (NCAR) Advanced Research WRF (ARW) and the National Center for Environmental Predictions' (NCEP) non-hydrostatic mesoscale model (NMM) releases into a single end-to-end forecasting system. All the capability of the NCEP and NCAR WRF packages are retained within the WRF EMS; however, the installation, configuration, and execution of the cores have been greatly simplified to encourage its use throughout the operational, private, and University forecasting and research communities.

Nearly every element of an operational NWP system has been integrated into the WRF EMS, including the acquisition and processing of initialization data, model execution, output data processing, and file migration and archiving. Even tools for the display of forecast and simulation data are provided. Real-time forecasting operations are enhanced through the use of an automated process that integrates various fail-over options and the synchronous post processing and distribution of forecast files. In addition, the WRF EMS can be easily configured to run on a single, stand-alone workstation or a cluster of Linux computers (Beowulf cluster).

More information regarding the WRF EMS as well as much of the same stuff written here can be found on the NWS Science Operations Officer (SOO) Science and Training Resource Center (STRC) web site:

<http://strc.comet.ucar.edu/wrfems>

1.2 Why Should You Care?

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

1. To improve the knowledge and use of NWP models and issues at the local level
2. To advance the forecasting process through an improved understanding of mesoscale atmospheric processes and the use of non-traditional diagnostic tools
3. To increase participation among the WFOs and other agencies in developing and running NWP systems to examine local forecast problems

Running the WRF EMS locally will serve to provide:

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's a good thing

1.3 A Summary of the WRF EMS features

- The WRF EMS is a complete, full-physics, NWP package that incorporates dynamical cores from both the NCAR ARW and the NCEP NMM WRF models into a single end-to-end forecasting system.
- The EMS is easy to install and configure. Users should be able to run simulations within 30 minutes of installation. And with wonderful documentation such as this, why shouldn't it be that easy?
- The rumors are correct, no compilers are necessary for running the WRF EMS. The system includes pre-compiled binaries optimized for 32- and 64-bit Linux systems running shared or distributed memory Linux environments. The MPICH2 executables are also included for running on local clusters across multiple workstations.
- The installation tool does just about everything for the user including the creation of a user account (if necessary), installation of the software, configuration of the system to run on multi-CPU workstations, and it even determines which binaries are best for you.
- Auto-updating capability has been integrated into the WRF EMS. When an update or patch becomes available, it is downloaded and installed automatically.
- The EMS includes preconfigured WRF ARW and NMM core benchmark cases so you can compare the performance of your system to others.
- The WRF EMS is designed to give users flexibility in configuring and running NWP simulations, whether it is for local research or real-time forecasting purposes.
- The WRF EMS system allows for the acquisition of multiple initialization data sets via NFS, FTP, and HTTP. Just like the developer, the system is semi-intelligent, in that it will determine which data sets from different model runs are available for ingestion at a given time.
- The system has the capability to reduce the likelihood of missed forecasts during real-time operational by incorporating multiple "fail-over" options that include alternate servers, data sets, or initial forecast hour. Should a run fail, the WRF EMS will also send e-mail to the user(s) alerting of a problem.
- All configuration parameters have been organized and documented in easy to read files that contain default settings for each dynamical core.
- The EMS can be configured to automatically calculate an appropriate simulation timestep for a given dynamical core and horizontal grid spacing.
- The EMS supports the automated processing of forecast files concurrent with a model run. The user can view forecast fields while the model is running.

- The post processor supports a wide variety of display software including AWIPS, BUFKIT, NCL, GrADS, GEMPAK, NAWIPS, and netCDF.
- The WRF post can process forecast fields on 81 different pressure levels from 10 to 1025mb.
- Forecast files may be exported to remote systems via SFTP, FTP, cp, and/or SCP depending upon the user's need.

1.4 How much computer power is necessary to run the EMS?

The answer to this question depends upon whether you will be executing the WRF EMS 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 4GB of physical memory. The amount of memory should be commensurate with CPU performance. That's just the nature of NWP as chances are that you will always want to run the model at higher resolutions over a larger 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 as speed of the machine is not as critical of an issue since you are not up against any deadlines to get the model forecast completed. Finally, if you plan on running nested simulations consider increasing the amount of physical memory even further.

The binaries compiled for the WRF EMS should 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 ~2 Gb, it is strongly suggested that machines have a minimum of 4 Gb for real-time modeling to avoid paging and swapping issues. In general, if you have less than 4 Gb of memory then consider increasing your system resources. Besides, memory is relatively inexpensive.

1.5 Is support for the WRF EMS available?

The SOO/STRC WRF EMS modeling package is fully supported by the NWS Science and Training Resource Coordinator (SOO STRC). US NWS and other NOAA agencies may request help from the SOO STRC 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 WRF EMS; 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 *all* WRF EMS activities are conducted by a single person. This includes testing, package design, development, support, research, computer maintenance, EMS real-time data server upkeep, web site development (or lack thereof), DVD burning, labeling, and mailing. So be kind and understanding as nothing gets done as quickly as it should.

1.6 What if I have a brilliant idea that must be incorporated into the EMS?

The WRF EMS 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 section 1.5 before doing so.

Chapter 2: Installation of the WRF EMS

Chapter Contents

- 2.1 Introduction -- Because you have to start somewhere**
- 2.2 System requirements for installing the EMS**
- 2.3 Installing the WRF EMS**
 - 2.3.1 The basics - The stuff you should know
 - 2.3.2 What happens during the installation process?
 - 2.3.3 Installing from DVD
 - 2.3.4 Installing from the internet
 - 2.3.5 Local network installation
 - 2.3.6 Making sure your installation was a success
 - 2.3.7 Additional, potentially useless, installation options
- 2.4 Updating your WRF EMS with passion**
 - 2.4.1 More basics - The stuff you sorta gotta know
 - 2.4.2 More potentially useless updating options
- 2.5 Additional information that may only be of limited use**
- 2.6 Target binaries and you**

2.1 Introduction -- Because you have to start somewhere

Installation of the WRF EMS requires the use of `ems_install.pl`, which was developed to tackle all the challenges that you might encounter if you were to attempt a manual installation of an NWP system. It is highly recommend that you use the most current version of this routine as your life will become less complicated and more rewarding if you do. *And that's a statement only a few modeling systems can make!*

So, "Where might I get this WRF EMS `ems_install.pl` utility thing" you ask?

All NOAA-affiliated users of the WRF EMS may simply request the current version of the routine from the SOO Science and Training Coordinator (SOO STRC). All non-NOAA users, i.e., everyone else, must register for the WRF EMS on the SOO/STRC site:

<http://strc.comet.ucar.edu/wrfems>

Registration allows the user to receive notification of updates as they become available and also allows the developer to advance his world geography knowledge when a request from Mauritius is submitted.

Note that there may be a delay in receiving the `ems_install.pl` utility following registration as the submitted information is checked for some semblance of legitimacy and to avoid advertisements for "personal enhancement" products from being sent to valid users. Once the EMS concierge is able to confirm your registration, which typically takes a few minutes but sometimes as much as 24 hours, you will receive a second message with the `ems_install.pl` routine attached. Enjoy!

2.2 System requirements for installing the EMS

There are a few system requirements for installing and running the WRF EMS, which are listed below:

- **A relatively current Linux distribution, but not too current** – The WRF EMS has been tested on Red Hat Enterprise, Fedora, CentOS, SuSe, and Ubuntu distributions, although the use of Ubuntu has required some changes to the (dash/bash) interpreter in the past. Other distributions will probably work; however, it's simply too difficult to keep up with all the releases and updates. There is typically a lag before the EMS developer can install a newly released OS and run the tests. So just stick with what works.
- **8 Gb of available disk space** – This requirement pertains to the installation of the EMS only. Of course, running an NWP model can use up a significant amount of disc space so this requirement should be considered as an absolute minimum.
- **The T|C shell must be installed on the system** – Linux purists may get all upset about this requirement but that's just the way it is for now; however, it will likely change in the future. Note that in many current Linux releases, the default is not to install **tcsh**, so you may have to install it separately. The `ems_install.pl` routine will check whether it is installed and provide an error message.
- **The EMS user must be using a T|C shell** – If you are installing the EMS under a known user then they must be running **csh** or **tcsh**; otherwise horrible things may happen such as the EMS failing to run. And you don't want that to happen. There are ways around this requirement but we're not going there right now.
- **Root permission and/or write permission on a large disc partition.** The `ems_install.pl` routine was designed to be run a root user. You will have the opportunity to assign ownership to an existing or new user during the processes. That said, the EMS can be installed by an existing user provided the user has write permission on the drive were the EMS will reside.

2.3 Installing the WRF EMS

2.3.1 The basics - The stuff you should know

Whether or not you have an existing EMS release on your system, you can do a fresh install with `ems_install.pl`. The novice `ems_install.pl` user should embrace the most basic of installation commands:

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

Note that the release version is optional, as indicated by the [square brackets], since the default is the most current release. But if you are looking to install a previous release, say version 3.1.1.4.22, you can try:

```
% ems_install.pl --install 3.1.1.4.22
```

The above example is for demonstration purposes only since release 3.1.1.4.22 may not actually exist, so don't try it at home. There is additional information on determining the available releases Section 2.5.

2.3.2 What happens during the installation process?

Regardless of what option you choose when installing the EMS, the process will include the following:

- A greeting, just because that's important and you're worth the effort

- 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 EMS installation resides at that location:
 - Get the version of existing installation
 - Ask whether you want to rename the existing installation to wrfems.<release>
- Prompt you for the name of the user to assign ownership of the package
- Create a new account and home directory if the user does not exist
- Check that the user's login shell is either tcsh or csh
- Prompt you for a password if a new user was created
- Finally install the EMS from the specified source
- Do the post-install configuration
- Congratulate you on yet another wise decision

Note that all sorts of useful information will be printed to the screen while the installation is running so don't leave the room.

During the installation process, the `ems_install.pl` routine will also attempt to:

- Determine the appropriate run-time binaries based on the CPU type on your system. See section 2.6 for more on target binaries.
- Attempt to determine the number of physical CPU and cores per CPU on your system. Should a problem occur, you will be prompted to give up this information and you will be powerless to resist.
- Install an entry (disabled) in the user's crontab file to automate the process of updating the EMS. You will be able to configure the `ems_install.pl` utility to automatically download and install updates from the SOO/STRC EMS servers and notify user(s) of any changes. More on the updating capabilities utility later.
- Install an entry (again, disabled) in the user's crontab file to assist in the automation of real-time forecasting. All you need to do is make the appropriate changes to this entry and you're off!

See, isn't that simple? The WRF EMS practically installs itself.

Note that the downloading WRF EMS tarfiles from the SOO/STRC server may require a considerable amount of time depending upon the speed of your connection. Be patient, as your effort will be well rewarded.

2.3.3 Installing from DVD

If you are lucky enough to possess one of the fabulous DVDs from the WRF EMS collector series then consider yourself blessed, because life doesn't get any better. Besides being a great coaster, the DVD can be used to install the EMS. This capability does come with a few caveats however so don't get too excited. Ok, go ahead and jump up and down a little.

In order to install the EMS from DVD you will need to actually mount the DVD, which may require root privilege on your system. For the sake of this guidance it will be assumed that you have mounted the EMS DVD under “**/media/cdrom**”, although the exact location may differ on your machine. If you're actually reading these instructions then it's assumed that you can figure out what to do.

Step a. Load WRF EMS DVD

Step b. Change directories to DVD drive, e.g., “`cd /media/cdrom`”

At this point you should see a copy of the install routine on the DVD. Use this version for the installation unless told otherwise by a rainbow or small domesticated farm animal with fur, not feathers. You can't trust the feathered ones.

Step c. Run the `ems_install.pl` routine like you want something good to happen:

```
# ./ems_install.pl
```

With any luck something good *will* happen and the installation process will begin. On some systems however, you may see an error such as:

```
# /ems_install.pl
```

```
bash: ./ems_install.pl: /usr/bin/perl: bad interpreter: Permission denied
```

Or

```
% ems_install.pl
```

```
ems_install.pl: Permission denied.
```

The above errors likely indicate that your DVD is mounted “**noexec**”, meaning you can't run an executable file from the DVD. Silly security restrictions.

All is not lost, besides the time required interpret the error message that is, because there is a work-around available:

Step c1. Copy `ems_install.pl` from the DVD to another location such as “`/tmp`”.

Step c2. Run `ems_install.pl` from that location with the following flags:

```
# ./ems_install.pl --dvd /media/cdrom (or wherever you mounted the DVD)
```

That should solve your EMS installation-related problems for now.

2.3.4 Installing from the internet

Unless you are installing the EMS directly from DVD (section 2.3.3), the default behavior of the `ems_install.pl` routine is to attempt to download the necessary files from the SOO STRC data sever. There is no need to include any special arguments or flags as everything, including the IP address of the SOO STRC server, are hardcoded into `ems_install.pl`; however, this may have to be changed in the future. Just follow the basic guidance provided in section 2.3.1.

2.3.5 A local network installation

If you don't have direct access to the SOO STRC server you can still install the EMS. This method requires that the EMS tar files be manually downloaded from the server and placed in a directory where they are accessible. Simply follow these few steps:

Step a. Create a temporary directory where the files are to be downloaded. This can be called anything provided that you have at least 3Gb of space on that partition. For this example the directory will be named “`/usr1/emsfiles`”.

```
# mkdir /usr1/emsfiles
```

Step b. Open up the SOO STRC site in your browser for full releases:

<http://soostrc.comet.ucar.edu/wrfems/releases>

and determine which release you want to install, which should be the most current one if you know what's good for you. The releases are identified by such silly names as "**3.1.1.4.99.beta**" or "**3.1.1.5.0**", because the developer was born without an imagination.

Step c. Create a second sub directory below `"/usr1/emsfiles"` with the name of the release to be downloaded. For example:

```
# mkdir /usr1/emsfiles/3.1.1.5.0
```

Step d. Again open up the SOO STRC site to the desired release:

<http://soostrc.comet.ucar.edu/wrfems/releases/3.1.1.5.0>

And download all the tar files to your `"/usr1/emsfiles/3.1.1.5.0"` directory. There may be quite a few and some are rather large, so while you are waiting you can take up waterfall kayaking or some other worthwhile activity.

Step e. Once all the files are downloaded and your injuries have healed, run the `ems_install.pl` routine as follows (Replacing the example with the actual location of the downloaded files on your system):

```
# ./ems_install.pl --install --locdir /usr1/emsfiles/3.1.1.5.0
```

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

2.3.6 Making sure your installation was a success

Following a successful and care-free installation, you should log out and return as the WRF EMS user. Make sure your EMS environment is set correctly by attempting the following commands:

```
% cd $EMS
```

And you should be located at the top level of the WRF EMS

Then try

```
% ls $EMS_STRC
```

There you will see the contents of the `$EMS/strc` directory. If both of the above tests are successful then try running the "**sysinfo**" command provided with your EMS (and it IS yours now.):

```
% sysinfo
```

You should see a summary of your system configuration that includes the Linux distribution and additional information such as your blood pressure (just kidding). Please make sure that the following values are correct:

Physical CPUs : # The number of CPUs or mice that you would see if you opened up the computer case
Cores per CPU : # The number of cores on each physical CPU, the stuff you can't see
Total Processors : # This value should be just Physical CPUs * Cores per CPU

If either the number of Physical CPUs or the Cores per CPU is incorrect, you will have to change these values in the \$EMS/EMS.cshrc file. If everything appears to be correct then your installation is complete and you are ready to become a modeler. If not, just give your not-so-local SOO STRC a call or send a message along with some baked goods.

2.3.7 Other additional potentially useless installation options

There are many command-line flags and options that may be passed to the `ems_install.pl` routine, not all of which are listed here simply because I've long forgotten what many of them do. However, I can recall enough to impart upon you sufficient knowledge to meet most of your WRF EMS installation desires. And that's really all you want out of life, isn't it?

Option [Optional] Argument

a. `--reldir` SOMEDIR

What this option can do for you:

Override the default location (directory) of the downloaded EMS release tarfiles. When installing the EMS, the tarfiles are placed in the "`<path>/wrfems/releases/<release number>`" directory unless you pass the `--reldir <full path>` option. The default is best unless you have a good reason to change.

b. `--domtrans` [SOMEDIR]

What this option can do for you:

Use this option if you want to transfer your computational domains from an existing EMS installation. Including the optional [SOMEDIR] tells the `ems_install.pl` routine where to look for the domain directories should they not be located in the default location. Not passing an argument to `--domtrans` means that a previous EMS is already installed, which will be replaced by the one that you are installing, and the domains located in the previous "runs" directory will be moved to the new "runs" directory. Additionally, all the configuration files will be updated and the domains will be relocalized unless you passed the `--nolocalize` option.

c. `--scour`

What this option can do for you:

Delete the existing EMS installation rather than saving it. It was old and full of holes anyway.

d. --nogeog

What this option can do for you:

Don't download and install the large WRF static GEOG files. Remember that you actually need these data sets to run the model, but if you already have them locally you may not need to download them again. You can just copy the files over to the new EMS installation. However, there are times when this data set changes and you may need the updated files. This is just a warning.

e. --locdir SOMEDIR

What this option can do for you:

Override the directory where the EMS releases and/or updates reside locally (SOMEDIR/<release>). You would use this option if you were installing the EMS from a local source rather than from the STRC server. The directory location, SOMDIR, should be the path to the directory where the individual releases are located. You do not include the release number in the pathname specified by SOMEDIR. THE install routine will figure that part out.

f. --[no]install

What this option can do for you:

[Do Not] Install the downloaded WRF EMS files (Default is to install them). Just let them sit in the wrfems/release/<release number> directory until they cool off.

g. --nolocal

What this option can do for you:

Don't use tar files found locally in wrfems/release/<release number> directory. Obtain them from the SOO/STRC server fresh and overwrite those in the directory.

h. --x64|x32|piv|px

What this option can do for you:

Download and install the x64 64-bit|x32 32-bit|xp4 32-bit|generic i386 binaries regardless of the system architecture. See the "Target binaries and you" section (2.5) below.

i. --[no]match STR,STR

What this option can do for you:

The argument to --[no]match is a list of comma separated character strings used to limit which files to download. You would use this option if you were being very selective in the WRF EMS tarfiles that you wanted to download and install. For example if you have a truncated file and needed to replace it. Other than that there is no reason why you need this option but it is available just in case.

j. `--[no]localize`

What this option can do for you:

[Do Not] relocalize existing computational domains. You would only use this flag along with the `---domtrans` option; otherwise it's just silly. Default is `--localize`.

k. `--curl|wget`

What this option can do for you:

Use only the `curl|wget` routine for http requests to the STRC server. You would use this option if you have a problem with the other Linux utility.

2.4 Updating your WRF EMS with passion

You will eventually need to update your old, crusty, buggy, and yet well-loved WRF EMS with a newer and shinier model containing new bugs and that new WRF smell. When that moment arrives you can call upon your trusty EMS installation tool (ya, it's a tool now) to tackle the job. It will someday be known as the Swiss army knives of EMS installation tools but for now it's just the leather awl of installation tools.

As you may or may not recall, when you installed the EMS, an entry was placed in your crontab files that will allow you to automatically download and install EMS updates. If you fancy this auto-update option then you must enable the crontab entry by removing the comment in front of the command line. If automation makes you nervous then keep it disabled and just fly manually while following the guidance provided you Luddite.

2.4.1 More basics - The stuff you sorta gotta know

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

```
% ems_install.pl --update [update release version]
```

Note that the update release version is optional, as indicated by the [square brackets], as the default is the most current release. It is recommended that you not include the update release and allow the `ems_install.pl` tool to do its magic. If you have been a slacker and find yourself a few dozen updates behind, no problem, `ems_install.pl` will figure out what updates you need to bring your system up to date. With any (a lot of) luck, the tool will download and install each missing update until you are current and just as brilliant as the day you first installed the EMS. At least that's the plan.

2.4.2 More additional potentially useless updating options

Below are some additional options and flags that you can pass to `ems_install.pl` while updating your system. Just as with the installation processes, some options require an argument and others take an [optional argument], while there are a few that require nothing at all.

Option	[Optional] Argument
--------	---------------------

a. **--updir** SOMEDIR

What this option can do for you:

Override the default location (directory) of the downloaded EMS update tarfiles. When updating the EMS, the tarfiles are placed in the "<path>/wrfems/updates/<release number>" directory unless you pass the `--updir <full path>` option. The default is best unless you have a good reason to change.

b. **--force**

What this option can do for you:

Force the installation of an update even though it's already installed. Maybe you messed something up and can't figure out where you went so terrible wrong, or maybe you want get your frustration out. I know, I've been there, which is why I included `--force` instead of the `--pickadaisy` option.

c. **--locdir** SOMEDIR

What this option can do for you:

Override the directory where the EMS releases and/or updates reside locally (SOMEDIR/<release>). You would use this option if you were updating the EMS from a local source rather than from the STRC server. The directory location, SOMDIR, should be the path to the directory where the individual release directories are located. You do not include the release number in the pathname specified by SOMEDIR. THE install routine will figure that part out.

d. **--[no]install**

What this option can do for you:

[Do Not] Install the downloaded WRF EMS files (Default is to install them). Just let them sit in the `wrfems/release/<release number>` directory until they cool off.

e. **--binonly** ARCH

What this option can do for you:

Download/install only the x64|x32|xpiv|px runtime binaries (`wrfm.exe` & `real.exe`).

f. **--nolocal**

What this option can do for you:

Don't use tar files found locally in `wrfems/release` directory- Get them from the SOO/STRC server fresh and overwrite those in the directory.

g. --x64|x32|piv|px

What this option can do for you:

Download and install the x64 64-bit|x32 32-bit|xp4 32-bit|generic i386 binaries regardless of the system architecture. See the "Target binaries and you" section (2.5) below.

h. --[no]match STR,STR

What this option can do for you:

The argument to --[no]match is a list of comma separated character strings used to limit which files to download. You would use this option if you were being very selective in the WRF EMS tarfiles that you wanted to download and install. For example if you have a truncated file and needed to replace it. Other than that there is no reason I why you need this option but it is available just in case.

i. --[no]localize

What this option can do for you:

[Do Not] relocalize existing computational domains. Use this option sparingly as you may actually need your domains relocalized for the updated EMS to work. You would only use this flag along with the --domtrans option; otherwise it's just silly.

j. --curl|wget

What this option can do for you:

Use only the curl|wget routine for http requests to the STRC server. You would use this option if you have a problem with the other Linux utility.

2.5 Additional information that may only be of limited use

If you take anything away from this section, it should be that you can always pass the "--help" flag to any of the WRF EMS utilities in hopes that it will provide some semblance of clarity. Whether it actually does "help" anything will depend upon the ability of the developer (that be me) to communicate an idea, and the user (that be you) to comprehend the information presented. So go ahead and feel free to give the "--help" option a test drive and behold the wonders of the EMS installation routine.

```
% ems_install.pl --help
```

A couple of other semi-useful options are the "--list" and "--listall" flags. These flags are used to interrogate the EMS server to find out the available releases and updates. For a simple listing of the available beta, experimental, and official full WRF EMS releases try

```
% ems_install.pl --list
```

And for an overly-verbose listing of the available WRF EMS releases and updates use:

```
% ems_install.pl --listall
```

The primary difference between the two options is that while “--list” will give you a summary, the “--listall” flag will provide you all the names of the individual files that comprise each release and update, which is probably more than you wanted to know anyway.

2.6 Target binaries and you

As stated previously, the EMS installation routine will attempt to determine which binaries are best for your system. It accomplishes this feat by checking the architecture of your system, either x64 or i386, and then running the wrfems/util/bin/cupid routine that determines the level of CPU on your system based upon the supported instruction sets. The EMS takes this information and determines the best fit from the various precompiled binary sets, which include:

- An INTEL/AMD unified binary for 64-bit systems (--x64)
- An INTEL Penryn or equivalent AMD level binary for 32-bit systems (--x32)
- An INTEL P4 or equivalent AMD level binary for 32-bit systems (--piv)
- A generic i386 INTEL and AMD binary for 32-bit systems (--px)

While there are many additional target CPUs for which EMS binaries can be created, an attempt is made to keep the number of possible options down while not sacrificing performance. This is done to save the developers sanity, which is probably long gone.

Amazingly, the routine that determines the CPU level of your system is not perfect and sometimes makes mistakes. Ya, it’s hard for me to believe but it’s true. A common indication that the installed binaries are not supported on your system is appearance of an “Illegal instruction” error when running the WRF EMS. Don’t worry if this error message ever appears, all is not lost as you can always install one of the other binary data sets by using the `ems_install.pl` tool and one of the target flags above.

For example, when doing a fresh install,

```
% ems_install.pl --release --piv
```

This forces the installation of the P4 binaries on your machine rather than the ones it thinks you should have. That’s show’n the EMS install tool who’s boss!

What happens when you have already installed the EMS and get the dreaded “Illegal Instruction” error? Well, you can try,

```
% ems_install.pl --update --force --piv --binonly
```

This instructs EMS install to update your system with the current P4 binaries. The --force option is necessary because by default, the install routine will not attempt to install same release that is already on your system, unless the --pickadaisy option is used.

Chapter 3: A self-guided tour through the WRF EMS

Chapter Contents

- 3.1 WRF EMS: What is in this thing?**
- 3.2 The WRF EMS Tour: A view from the tippy top**
- 3.3 The all-important EMS.cshrc file**
- 3.4 All the environment variables you love to hate**
- 3.5 WRF EMS run-time routines**
- 3.6 The many utilities you would have missed had they not been listed here**
 - 3.6.1 Some additional WRF EMS utilities
 - 3.6.2 Non EMS-developed tools and utilities
- 3.7 A WRF EMS flow diagram**

3.1 WRF EMS: What's in this thing?

At this point you should have successfully installed the WRF EMS on your computer, because if you have not, then this chapter will be of little use to you. So, if the EMS is not yet installed go back to Chapter 2 and return here only after you have completed that task.

At first glance it may appear that the EMS consists of numerous directories, subdirectories, and files that require the constant attention of the user. The reality is that you can just ignore the man behind the curtain. While it is true that there are many directories and files, users for the most part will not have to modify their contents. The EMS was designed so that both the novice and well-seasoned NWP user could feel comfortable with the system. Nearly everything comes pre-configured for most applications. Nonetheless, there are always exceptions, and thus this chapter provides an overview of the system should you feel it necessary to dig deeper into the bowels of the EMS.

3.2 The WRF EMS Tour: A view from the tippy top

There are a number of environment variables that may be used to navigate the WRF EMS, many of which will be explained in this chapter. The most important variable is \$EMS, which defines the top level of the EMS installation. As you recall from Chapter 2, one test of a successful installation is the ability to change to the \$EMS directory, i.e., “cd \$EMS”. Go ahead and try it now as this is where your self-guided tour will begin.

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

- bin** The **bin** directory contains the 32- and 64-bit Linux binaries provided with the WRF EMS. These binaries have been specially compiled for use with the system. If you would like to use your own binaries please contact the SOO STRC first; otherwise, you will fail.
- data** The **data** directory is the location for the large terrestrial data sets, default model configuration files, default namelist files, and tables used when running the system. There are quite a few files and directories beneath data; however, you probably won't need to concern yourself too much unless you are trying to make permanent changes to the system.

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

- | | |
|----------------|--|
| geog | Contains the various terrestrial data sets used with the WRF |
| domains | Contains sample NMM and ARW domains |
| narr | Contains land mask fields for running simulations with NARR data |
| tables | Contains the default EMS configuration files and tables |
- docs** The **docs** directory contains assorted pieces of documentation on the WRF and WRF EMS. Not all of the information is relevant to the EMS but users may find some value in the resources.
- dwiz** The **dwiz** directory contains the Java libraries and binaries used when running the WRF Domain Wizard (DW).
- logs** The **logs** directory contains the installation and Domain Wizard (dwiz) log files. It is also the default location for the `ems_autorun.log` files. The log files from the individual domain simulations are **not** located in “**wrfems/logs**” but can be found in “**wrfems/runs/<domain>/log**”.
- runs** The **runs** directory is where all your magic happens. This is where your newly-created domain directories will reside after you have completed a successful localization with the Domain Wizard. It is also where you will run your simulations with the EMS.
- strc** The **strc** directory and subdirectories contain all of the Perl routines and modules used to power the WRF EMS. It is unlikely that you will ever need to edit the files in this directory. These files are also the most likely to be updated with each new EMS release.
- util** The **util** directory contains many of the utility programs and packages provided with the EMS that help to make it a “system”. The preconfigured domains used for running the benchmark cases reside here as well as the 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 “**wrfems/util**” has to offer.

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

- | | |
|------------------|--|
| bin | Various precompiled non-EMS developed utilities (Section 3.6.1) |
| benchmark | The NMM and ARW core benchmark domains |
| grads | GrADS binaries, scripts, and data files |
| mpich2 | MPICH2 routines, Python scripts and documentation |
| nawips | NAWIPS model data manipulation and display package |
| ncview | NeView binaries, scripts, and data files |
| workshop | Contains the data, html files, and documentation for the EMS exercises |
| wrfems | The EMS-modified WPS and WRF release files |
| wrfutils | Various precompiled NCAR-provided WRF utilities |

3.3 The all-important EMS.cshrc file

When you log in as a WRF EMS user, a number of environment variables are set that are used by the Perl modules and executable routines included with the system. All of these variables are defined in the `EMS.cshrc` file, which resides in the top level of the WRF EMS installation (`$EMS`). Some of the variables in this file may be modified by the user if necessary, but for the most part the EMS install tool should have done an adequate job during the configuration process. Nonetheless, it's always a good to make sure you are happy with the settings.

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

NCPUS and CORES

The NCPUS environment variable defines the number of physical processors that reside on your local system. NCPUS **does not** define the total number of processors (NCPUS x cores), nor does it define the processors used when running a simulation. That assignment is accomplished in the `run_ncpus.conf` file, which is one of the configuration files for each computational domain created. The NCPUS 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 physical CPUs although some most exotic and expensive mainboards can support 8 or more, but you probably don't have one of those.

The CORES environment variable defines the number of cores contained within each of those touchable CPUs. The description of the processor type in the `/proc/cpuinfo` file might provide a clue as to the number of cores on each CPU such as "dual core", "Duo", "quad core", or "6-Core". If you read a "tri-core" or "hepta-core" then you got other problems. If you really don't know how many cores there are on each CPU then try running the EMS "**sysinfo**" utility; otherwise, look up your CPU on that internet thing.

The NCPUS and CORES environment variables are used by the EMS to define the *total number of processors available on your system* (NCPUS * CORES). It is important to have these values correct as many of the binaries are compiled to run on multiple shared memory processors and the EMS needs to know just how many processors are available. Note that with the exception of the WRF "real.exe" and the NMM and ARW core executables, EMS binaries can only utilize the processors available on the local system. You cannot, for example, distribute the processing of initialization data across a cluster of workstations. This minor limitation was introduced to simplify EMS development since the advantage in exporting computational load to other systems was minimal.

Also note that setting either NCPUS or CORES greater than the true values for your system will result in degradation of EMS 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 EMS installation tool indicates that you have twice as many physical CPUs and cores on your computer than you think it does. If hyper-threading is turned ON in the BIOS then the system can mis-report the correct number processors and cores by 2-fold. In that event you will have to manually assign the correct values to NCPUS and CORES in `EMS.cshrc` as your performance will be compromised. You are probably better off turning hyper-threading OFF in your BIOS but that is up to you.

DSKCHEC

The DSKCHEC environment variable allows the user to monitor disk space usage on the partition where the EMS resides. Running an NWP model can generate lots and lots of Giga-chunks of data and, if not monitored, a partition can fill up with less-than-desirable consequences. Setting DSKCHEC to "Yes" will result in the EMS checking available space before the start of each EMS run-time routine. If space is limited (greater than 95% usage), a warning message will be printed to the screen. The routine will also automatically terminate your simulation and provide you with a message (via email too) when your usage reaches 100%. That's the way it's supposed to work at least.

MAILEXE

The MAILEXE environment variable defines the mail routine to use when sending you informative

messages or pearls of nonsensical wisdom, which tends happen from time to time. Just like a fortune cookie that provides lottery numbers. Note that while the email recipients for disk space warnings are defined by the `USERS` variable, there is a separate `USERS` configuration available for each user-created domain located in the `ems_autorun.conf` configuration file. More about that file in Chapter 10.

USERS

The `USERS` environment variable provides a list of email address that will be used in the event of a disk space problem only (See `DSKCHEC`). This variable is not to be confused with the other `USERS` parameter, available in each domain `ems_autorun.conf` configuration file, which allows you to send email to a separate list of users in the event of a failed model run. Similar, but different.

Individual email addresses are separated by commas. Leave `USERS` blank if you don't want mail sent because it just gets in the way of your solicitations for “male enhancement” products.

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 193. Why is the number “193” important? Well, because there is an internal check in `MPICH2` for the number of environment variables set. If a user has more than 200 (193+ 7 that `MPICH2` includes) the `MPICH2` routines will fail to start. I don't know why. I didn't write `MPICH2` but it's there.

3.4 All the environment variables you love to hate

There are additional environment settings that are defined in the `EMS.cshrc` file that are used by the `EMS` routines and utilities, even a few of which that may be of use to you. Those variables that may be of interest are identified below:

- \$EMS** The top levels of the WRF EMS installation (`wrfems`)
- \$EMS_BIN** The primary directory where the EMS binaries reside (`wrfems/bin`)
- \$EMS_DATA** The directory where the static data sets reside (`wrfems/data`)
- \$EMS_STRC** The location of the EMS `STRC` Perl modules and routines (`wrfems/strc`)
- \$EMS_RUN** The directory where all your computational domains are located (`wrfems/runs`)
- \$EMS_UTIL** The utility (closet) directory (`wrfems/util`)
- \$EMS_CONF** The location of the default configuration and grib information files (`wrfems/conf`)
- \$EMS_LOGS** Location of the general EMS log files (`wrfems/logs`)
- \$EMS_DOCS** Location of the EMS and other WRF documentation (`wrfems/docs`)

3.5 WRF EMS run-time routines

The run-time routines are the heart and soul of the WRF EMS. These are the tools that you will need to acquire and process initialization data, run simulations, and post-process the model output files. They are also used to automate the entire process when running a real-time forecast system. You

should be kind to the run-time routines and they will be kind to you.

All the run-time routines and ancillary modules are located below the `wrfems/strc` (`$EMS_STRC`) directory; however, **they should 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 `wrfems/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 `wrfems/strc`. Taking this approach makes it easier for the EMS to keep things tidy. And a tidy house is a happy house.

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

The primary (big) four (Kind of like the “BCS” of the EMS):

ems_prep	Used to identify, acquire, and process the external data sets for use as initial and boundary condition information in the WRF EMS.
ems_run	Ingests the output from <code>ems_prep</code> , created the initial and lateral boundary conditions for the model run, and then executes the simulation.
ems_post	Processes all the model output and exports the data files to exotic locations of your dreams.
ems_autorun	Automates the process of executing <code>ems_prep</code> , <code>ems_run</code> , and <code>ems_post</code> in succession for the purpose of creating a real-time simulation experience.

There are two additional run-time utilities that you will likely use but are not part of the “EMS BCS”. They do not have links from the domain directories.

ems_clean

The `ems_clean` routine is used to return your run-time domain directory to some user-requested state. This process includes the removal of unnecessary files and directories and reestablishing symbolic links. The routine is automatically run whenever you execute one of the run-time routines above but may also be run manually from within a run-time domain with the level 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 settings (level 6). Most users will employ the level 1, 3, or 4. A more detailed explanation of the power that is `ems_clean` is provided by running “`ems_clean --help`”.

ems_autopost

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

3.6 The many utilities you would have missed had they not been listed here

The WRF EMS includes a number of utilities that may be used to enhance your local modeling experience, and then there are others that are simply useful. Below is a summary of those utilities and the location where they reside.

3.6.1 Some additional WRF EMS utilities

The following utilities are located in the `wrfems/strc/ems_bin` directory but should be in your path when you log in so there is no need to point to them directly

ems_domain

The `ems_domain` routine allows users to relocalize their computational domains should there be a problem running the Domain Wizard (`dwiz`) utility. It may be run directly from the command line or indirectly through `ems_clean` with “`ems_clean --level 5`” or “`ems_clean --level 6`”. More information is provided in the “Domain creation from the command line” section of Chapter 5.

ems_guide

The `ems_guide` routine is a command-line quasi- user’s guide which basically falls somewhere between this document and the information provided when passing the “`--help`” flag to any of the EMS run-time routines. Lots of good information and guidance provided.

runinfo

The `runinfo` routine provides a summary of the model configuration settings for the computational domain in which it is run. Basically, running “`runinfo`” will give you a description of the domain configuration, forecast file format and output frequency, physics and dynamic settings for the simulation, as well as a few other less interesting tidbits of information. If you plan on including nested domains in your run then you can include the “`--domains`” flag, i.e.

USAGE: % `runinfo --domains 3, 5`

As always, additional information is provided by passing “`--help`”.

sysinfo

The `sysinfo` utility is used to provide 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: % `wrfems@kielbasa-> sysinfo`

System Information for `kielbasa.comet.ucar.edu`

```
System Date       : Fri Feb 30 19:54:23 2012 UTC
Alternate Hostname : None
Machine Address   : 10.0.1.3
```

```
Alternate Address : 128.117.110.37
Machine OS       : Linux
Kernel          : 2.6.18-128.7.1.el5
Linux Distribution : CentOS release 5.3 (Final)
CPU Name        : Six-Core AMD Opteron(tm) Processor 2435
CPU Type        : x86_64
CPU Instructions : barcelona
Physical CPUs   : 2
Cores per CPU   : 6
Total Processors : 12
CPU Speed (MHz) : 2600.000
System Memory   : 24177 Mb
Binaries        : STRC Compiled x64
```

Information for User wrfems on kielbasa.comet.ucar.edu

```
User ID          : 8010
group ID         : 1750
Home Directory   : /home/wrfems
User Shell       : /bin/tcsh
Shell Installed  : Yes
~/mpd.conf Exists : Yes
Shell Login Files : .cshrc
EMS.cshrc Sourced : .cshrc
EMS Home Directory : /usr1/wrfems
EMS Util Directory : /usr1/wrfems/util
```

gribnav

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

For users interested in sending your data into AWIPS, gribnav replaces the “**wgrib --A**” option provided with EMS V2.

netcheck

The netcheck routine should be used prior to running the WRF EMS on a cluster of Linux computers. It was designed to identify potential problems that may arise with MPICH2, specifically local networking issues, as MPICH2 is rather picky about the networking and communication configuration in your cluster. To use, run netcheck from the master node on your system:

USAGE: % netcheck master node1 node2 node3 ... nodeN

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

grib2cdf

The grib2cdf routine reads the contents of a grib file and writes out a netCDF formatted file along with a companion “cdl” file. The netCDF file will have a name similar to that of the grib file except with a “.nc” extension.

USAGE: % grib2cdf <grib file> ... <grib file>

This utility is simply a front end to a variety of programs that are used to convert the data from one format to another and then extract the necessary information. There is nothing fancy going on here.

3.6.2 Non EMS-developed tools and utilities

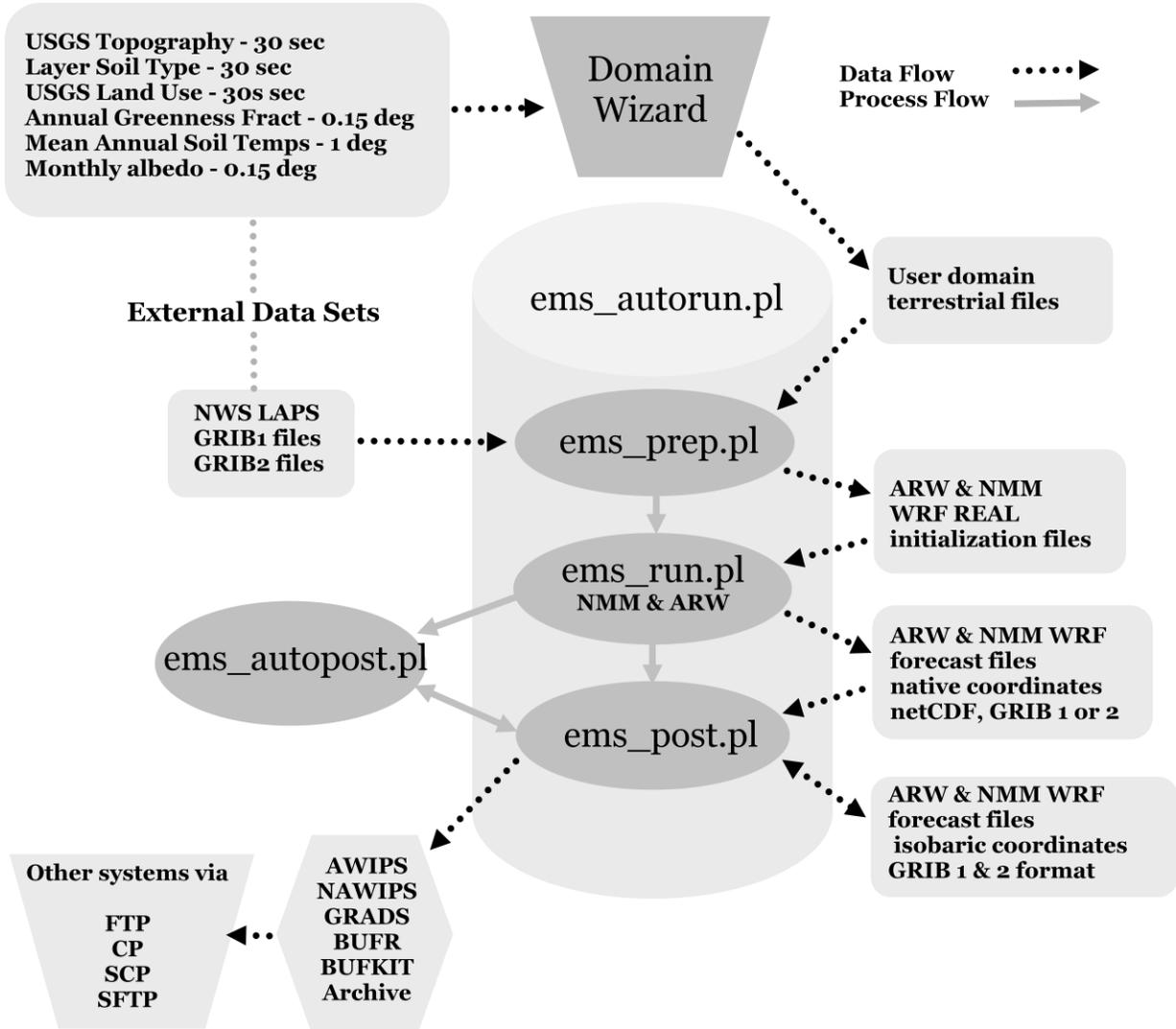
Here is a brief summary of the “value added” utilities that are provided with the WRF EMS, many of which are used by the run-time scripts so it’s in your best interest to not remove them. Besides, a few may be of use to you:

Located in the wrfems/util/bin directory:

- a) **cnvgrib** Converts between GRIB 1 and GRIB 2 formats (Goes both ways)
- b) **copygb** Remaps GRIB 1 files to new grid navigation/projection
- c) **cupid** Provides information of about your CPU
- d) **g1print** Prints out information in GRIB 1 files for creating a Vtable
- e) **g2print** Prints out information in GRIB 2 files for creating a Vtable
- f) **gribdump** A GRIB 1 file interrogation tool
- g) **ncdump** A netCDF file interrogation tool
- h) **ncview** A netCDF file visualization tool
- i) **rdwrfin** Prints out summary of WRF intermediate format files
- j) **rdwrfnc** Prints out summary of WRF netCDF format files
- k) **wgrib** A very special GRIB 1 file interrogation tool (EMS-modified)
- l) **wgrib2** A very, very special GRIB 2 file interrogation tool (EMS love added)

3.7 A WRF EMS flow diagram

So here is a diagram depicting the process flow through the WRF EMS. There is lots of stuff going on here and it really doesn't look pretty in pictures but it serves as a good summary



Chapter 4: Just Getting Started? Read this Chapter!

Chapter Contents

- 4.1 Just the basics
- 4.2 Help! Help! - How Do I Run this WRF EMS?
- 4.3 Additional information you might want to know

4.1 Just the basics

This chapter will be short and sweet, because there is nothing like a jumble of nonsensical, incomprehensible verbiage to dissuade potential users of a software package to turn off their computers in favor of needlepoint. You will get enough blather in the remaining chapters of this guide. This is the time for parental guidance and handholding, which we all need from time to time whether we admit it or not.

So, if you want to know the basics of running the WRF EMS then read this chapter. If you are a well-seasoned EMS user then you can skip this information and move on to something else.

4.2 Help! Help! - How Do I Run this WRF EMS?

Now that you have successfully installed the EMS and hopefully completed one or both of the benchmark cases (Appendix B), it is time to set up and run a simulation over your own domain. Since the EMS comes preconfigured you should not have to make any changes, although there are plenty of options with which to experiment as you become more comfortable with the system. However, we will stick to the basics for now.

Running the WRF EMS is straight forward 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 the “**dwiz**” command and the GUI should 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 computational domain if you are a “*Instructions? I don’t need no stinking instructions*” type of person. It’s fairly straight forward and few problems should be encountered provided you remember to:

- a. Select “Rot Lat-Lon” under the “Projection options” menu for the WRF NMM core, anything else for the ARW core
- b. ***If running the NMM core, the S-N grid dimension (NY) must be an even value!*** Even if the DW allows you to enter an odd integer, which is a bug in the code. Using an odd integer will cause the localization to fail!
- c. Be sure to localize your domain by pushing the “Localize” button at the top of the localization window.

Review Chapter 5 if you should have any questions. Once this step has been successfully completed you can move on to Step 2.

Step 2 – Prepare to run a simulation

While running the DW, a new domain was created and placed in “**wrfems/runs/<domain name>**”. It is from this directory that you will run any simulations.

Chapter 04 – Just Getting Started? Read This Chapter!

In the newly minted domain directory you will find a number of links and sub-directories that may be new to you. If you want to learn more about these files and directories then spend some time in Chapter 6, but just not now because you have other work at hand.

The EMS default configuration should be sufficient for running simulations with a grid spacing of 10 to 25km. Nonetheless, there are two configuration files that you should check just to make sure everything is correct, because, you know, stuff sometimes happens.

In the “**conf/ems_run/run_ncpus.conf**” configuration file, make sure that the number of CPUs is 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 *times* the number of cores per CPU. Check the following parameters in this file:

```
REAL_NODECPUS = local:#  
WRFM_NODECPUS = local:#
```

Where the “#” should be replaced with the number of processors. You can also use:

```
REAL_NODECPUS = local  
WRFM_NODECPUS = local
```

Without the “: #” wherein the EMS will use the **CORES** and **NCPUS** environment variables (Chapter 3) to calculate the total number of available processors on your system.

The other configuration file that you want to inspect is “**conf/ems_run/run_physics.conf**”, which contains all the model physics settings for your simulation. The only parameter that you may consider changing for now is **CU_PHYSICS** and only if your grid spacing is less than 8 km. If this is the case than you can change:

```
CU_PHYSICS = 1, 0  
To  
CU_PHYSICS = 0, 0
```

But you do not have to make this change. It’s just a recommendation.

While you are inspecting the physics configuration file you are encouraged to familiarize yourself with the various parameters and options, only because you will be seeing them again when you really get into modeling, which is inevitable at this point so don’t try to resist. Other than those two possible changes you don’t have to modify the default configuration.

Once this step has been successfully completed you can move on to Step 3.

Step 3 – Processing your initialization data

This step assumes that you have an internet connection so you can access the initialization data sets. You don’t need to have a connection to the outside world provided that the data files already reside on your local system (Chapter 7) but for this example it is assumed that you are connected.

If you have good bandwidth you can try running the following command:

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

If you have limited bandwidth then try running:

```
% ems_prep --dset gfsptiles --length 24
```

Running either command causes the `ems_prep` routine (Chapter 7) to download and process the first 24 hours of forecast files from the most recent Global Forecast System (GFS) operational model run. These files will serve to provide your initial and boundary condition data for your 24 hour simulation. If you want to include and sub (nested) domains then you would also include the “`--domains`” (Chapter 7) option but we are not there yet. We are taking baby steps here.

Once this step has been completed successfully you can move on to Step 4.

Step 4 – Running your simulation

This next step is easy, simple run:

```
% ems_run
```

At which time the WRF EMS will create the WRF-formatted initial and lateral boundary condition files and start the simulation. Some semi-valuable information will be printed to the screen so don't be distracted by anything going on outside your window like a passing fire truck. You are a modeler now. Besides, real modelers don't have windows. So if you do then please cover them with Star Trek or Albert Einstein posters or something. This should be step 4b – “cover windows”.

Running the simulation may take a while depending on the size of your domain, the grid spacing used, and the performance of your machine. Remember that the time required to run a simulation will depend upon your available computer resources and is proportional to the number of grid points in your domain and inversely proportional to your grid spacing. Just for your modeling education, *decreasing the grid spacing by 1/2, say from 20 to 10km, 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 watching that fire truck.

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

Step 5 – Processing the model output files

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

Following a successful simulation, all your WRF output files will be located in the “`wrfprd`” directory. Unless you have changed the default format type, these data should be in netCDF format and may be viewed directly with any utility that reads netCDF.

For example, if you want a summary of the file contents:

```
% rdwrfnc -m <netCDF file>
```

Or to (albeit limited) the various fields in the file:

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

Now, if leaving the output files in netCDF is not in your plans then you can use “`ems_post`” to process the data into another format, generate images, or export the files to another system. There are many options available. It's really up to you. Much more information on using `ems_post` is available in Chapter 9; however, for now you can simply take comfort in converting your netCDF files into GRIB 1 format by running:

% **ems_post**

Once your processing has successfully completed you can show your friends and/or significant other what you have been doing all alone in that room with the windows blocked.

4.3 Additional information you might want to know

Once you become more familiar with running the model you can simplify the entire process by simply using the **ems_autorun** routine, which is discussed in Chapter 10. This tool is primarily designed for real-time simulations but may be used for case studies as well. Also, if you decide to use **ems_autorun** for real-time forecasting, then you might also want to consider using the EMS autopost option, which is detailed in Chapter 11. Good Luck.

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

Chapter Contents

5.1 Just the mastering basics

5.2 Be the “Wizard of Your Domain” with the Domain Wizard

- 5.2.1 Starting the Domain Wizard
- 5.2.2 The “Choose Domain” window
- 5.2.3 The “Horizontal Editor” window
- 5.2.4 Defining your ARW and NMM core primary domain
- 5.2.5 Creating nested domains for the NMM and ARW core
- 5.2.6 Creating domain localizations
- 5.2.7 Visualizing the terrestrial data

5.3 Domain creation from the command line

- 5.3.1 Creating a new domain with `ems_domains.pl`
- 5.3.2 Manually configuring the navigation for a new domain
- 5.3.3 Manually localizing your new domain

5.1 Just the mastering 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 includes two steps, 1) Defining the navigation of your domain(s), and 2) Localizing the domain to generate the static terrestrial data sets. The creation and localization of the computational domains can be achieved either through use of the Domain Wizard (DW) GUI or pseudo-manually by running the `ems_domains.pl` utility from the command line.

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

- a. The map projection you select will determine the WRF core (NMM or ARW) used. Select the Rotated Latitude-Longitude projection (Rot Lat-Lon) for the NMM core or Lambert Conformal, Polar Stereographic, Mercator, or Lat-Lon projections for the ARW core.
- b. If you are using the ARW core, it is recommended that you use:
 - i. Lambert Conformal projection for mid latitudes
 - ii. Polar Stereographic high latitudes
 - iii. Mercator or Lat Lon for low latitudes
- c. You must run a successful localization in order to proceed with your run.

The localization step is handled by the WPS “geogrid” routine. The purpose of geogrid is to define the simulation domains, and interpolate various terrestrial data sets to the model 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 to the model grids by default. Global data sets for each of these fields are provided with the EMS. Several of the data sets 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. New or additional data sets may be interpolated to the simulation domain through the use of the table file, `GEOGRID.TBL`. The `GEOGRID.TBL` file defines each of the fields that will be 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 data set for that field is located. The various GEOGRID.TBL files are located in the wrfems/data/tables/wps directory. Output from geogrid is written in the WRF I/O API format, and thus, by selecting the NetCDF I/O format, geogrid can be made to write its output in NetCDF for easy visualization using external software packages, including ncview. (*Liberally adapted from the official ARW users guide*)

5.2 Be the “Wizard of Your Domain” with the Domain Wizard

WRF Domain Wizard is the successor to the WRFSI GUI and is a graphical user interface (GUI) for the WRF Preprocessing System (WPS). It enables users to easily define and localize domains by selecting a region of the Earth and choosing a map projection. Users can also define nested domains using the nests editor, run the WPS geogrid through the GUI, and visualize the NetCDF output. (*Also liberally borrowed from the DW web site*)

Note: The WRF EMS includes a precompiled modified 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 you should be able to run the GUI:

```
% dwiz (Domain Wizard GUI)
```

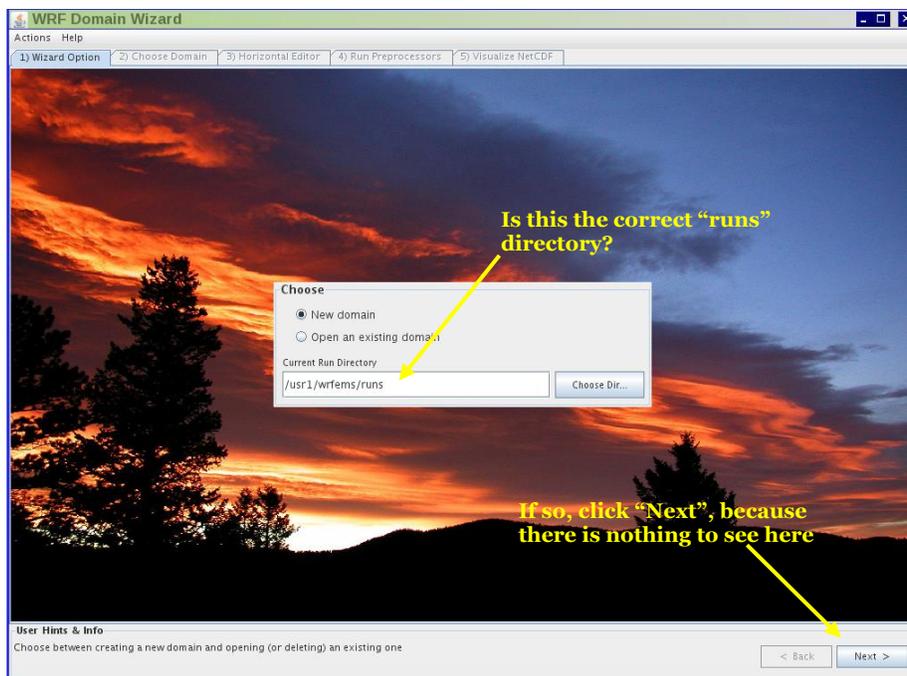
At which time the DW window should appear (see image below or next page depending on the formatting of this document). Always present at the top of the GUI window is the standard menu bar at the top of the window. The user can exit the application or query the help pages and version information by selecting options found under the Actions and Help buttons of the menu bar, respectively. 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 been completed.

FYI - The best way to learn how to use the DW is by playing with the tool while an expert looks over your shoulder. Unfortunately, most of you do not have this luxury and will have to be more proactive in educating yourself on the fundamentals of the DW.

The initial window gives the user an opportunity to select from creating a new domain and modifying an existing domain. The “Current Run Directory” is defined from the \$EMS_RUN environment variable so if the default location is not to your liking you can either:

- a. quit DW, setenv EMS_RUN <your directory>, start DW again
- Or
- b. reset the value in the window by selecting “Choose Directory”

Chapter 05 Just Because You Can – Mastering Your Model Domain

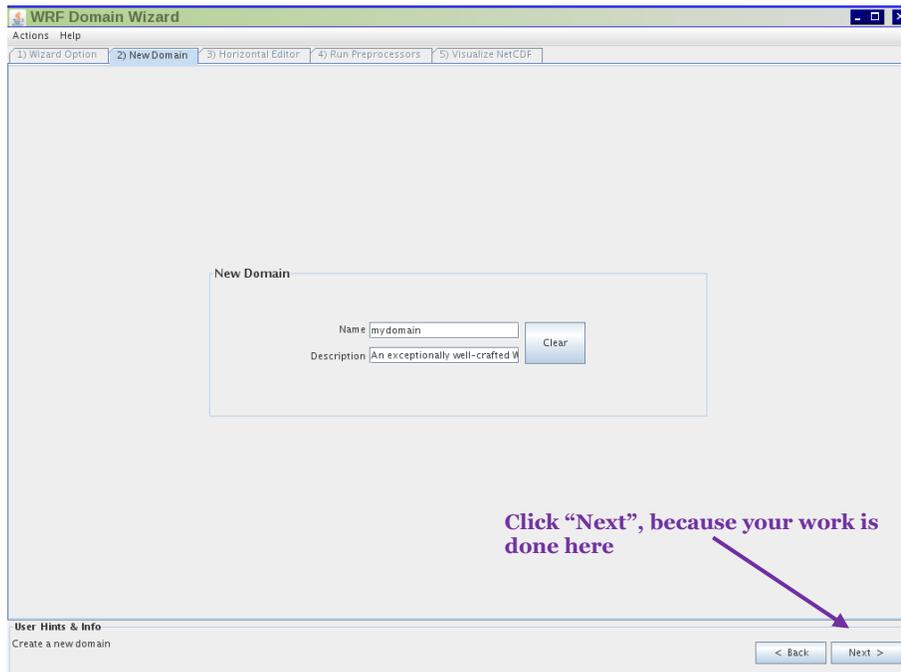


The above image depicts the initial DW window. When creating a new domain simply select “Next” at the bottom right and move along to the “Choose Domain” challenge.

5.2.2 The “Choose Domain” window

This window allows the user to give a name to the domain that is being created. The name should not include spaces although you may use a mix of upper and lower case (**see exception below**). Something snappy and clever is highly recommended, such as that used in the image below.

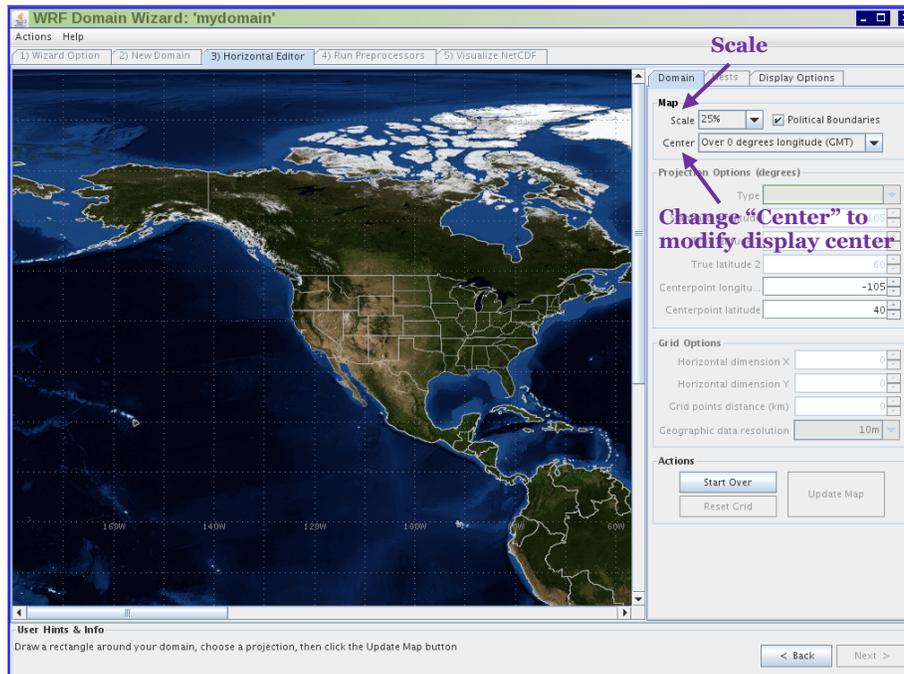
Note: *If you plan on using GEMPAK with the data files in this domain it is recommended that you use all lower case. This is because GEMPAK does not recognize upper case.*



Once you have given your domain a name then select the “Next” button and move to the “Horizontal Editor” window. Go ahead and click it – Now.

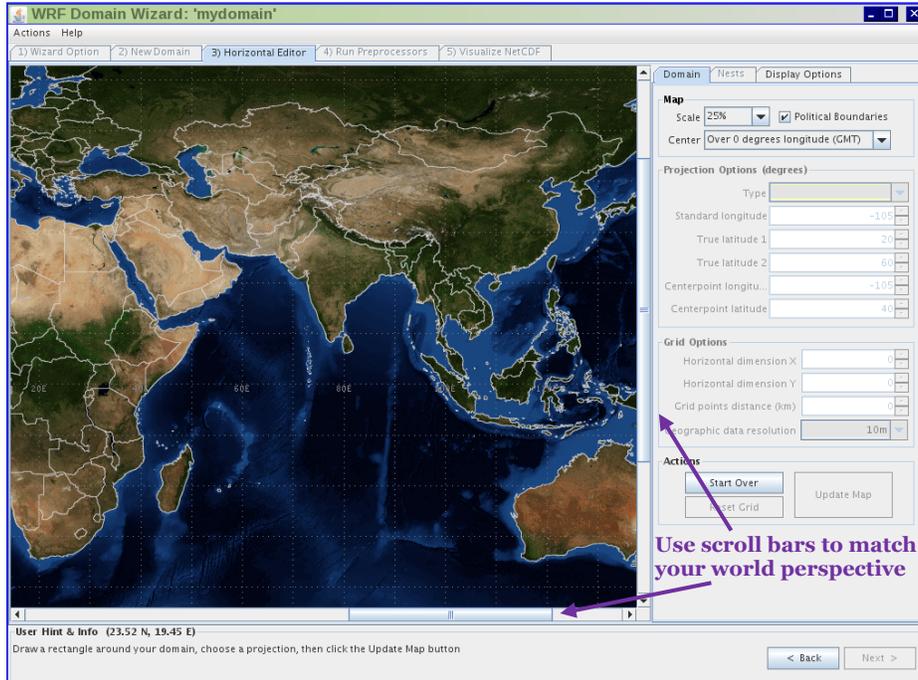
5.2.3 The “Horizontal Editor” window

The Horizontal Editor window allows the user to define a model domain by drawing a bounding box on a map (left side) and editing map projection variables (right side). You will use this window to draw a box around the general area that will define your computational domain and then choose a map projection. You will then be able to fine-tune the domain shortly so there is no need to be exact here.

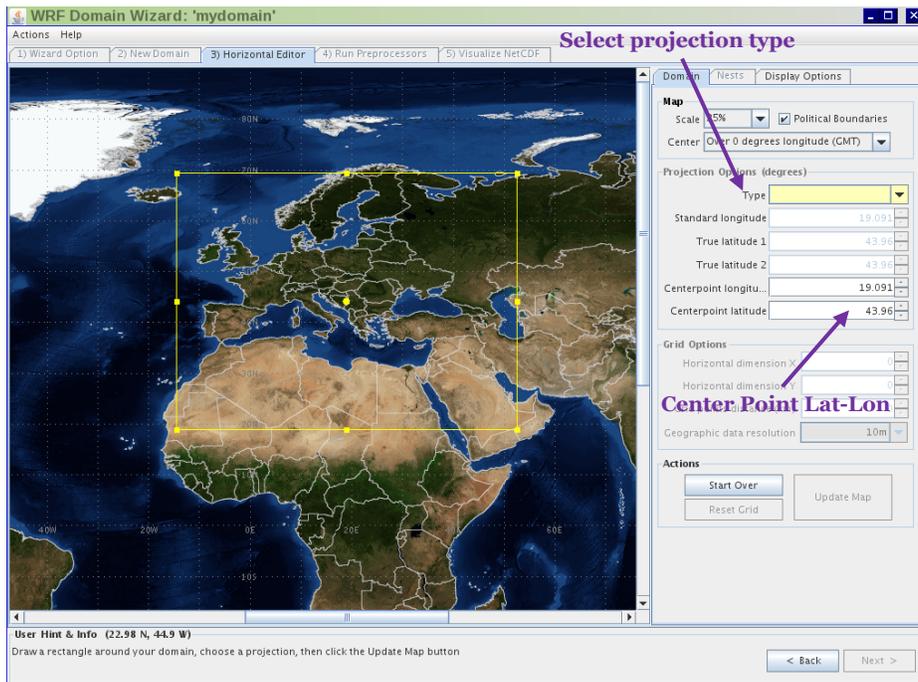


The initial global map image above shows a Cylindrical Equidistant projection of the world centered over North America. The map image can be increased or decreased in size by changing the “Scale” in the upper right of the window. The entire global map is not displayed all at once, but the panel has sliding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

Chapter 05 Just Because You Can – Mastering Your Model Domain



Initially, only the global map is active and all other interface buttons are disabled. As the user presses mouse button 1 (left click) on the global map and drags the mouse to a new location, a yellow domain-bounding box will be drawn that specifies a domain. Qualitative status information will be displayed in the User Hints & Information panel, specifically the lower left (LL) and upper right (UR) corner latitudes and longitudes in the selected domain.

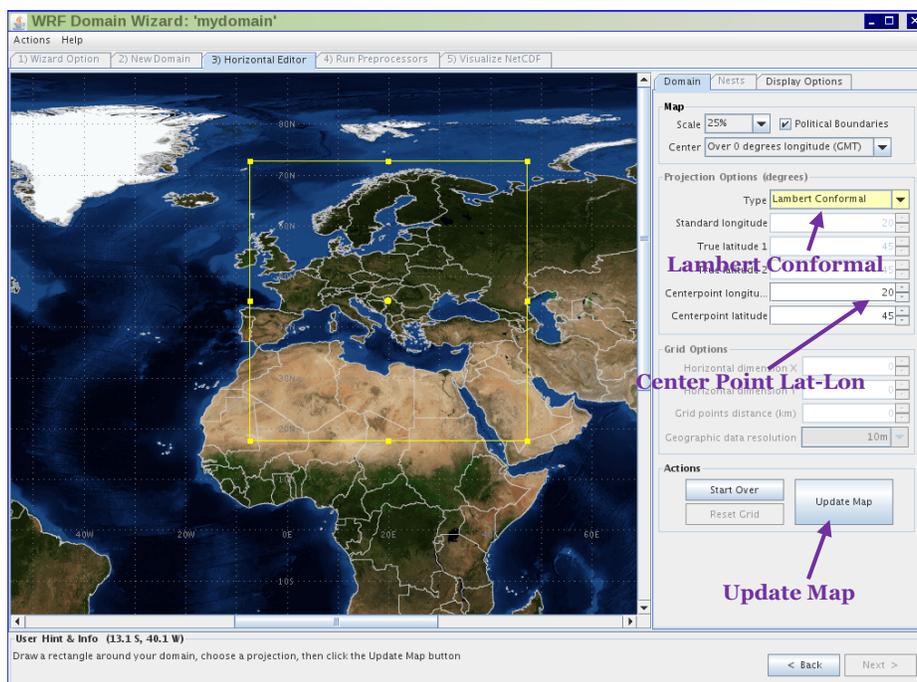


5.2.4 Defining your ARW and NMM core primary domain

As stated previously, your choice for map projection will determine whether you are running the NMM or ARW core of the WRF model. This selection is made from the “**Horizontal Editor**” window in the “**Projection Options**” section. There, click on the “**Type**” menu and choose:

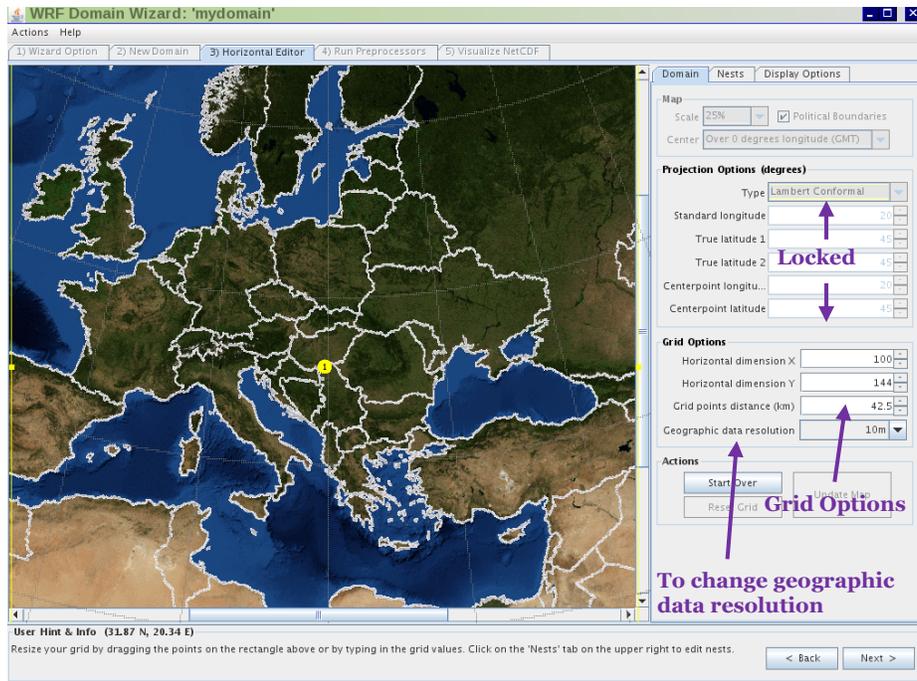
- “Rot Lat-Lon (NMM)” for the NMM core
- “Lambert Conformal”, “Polar Stereographic”, “Lat-Lon”, or “Mercator” for the ARW core

You can also modify the center point location of your primary domain (Domain 1) at this time. These values can be changed either by highlighting and editing the settings in the box or by using the increase/decrease buttons. As the changes are made to the primary domain the bounding box that defines the domain also changes.



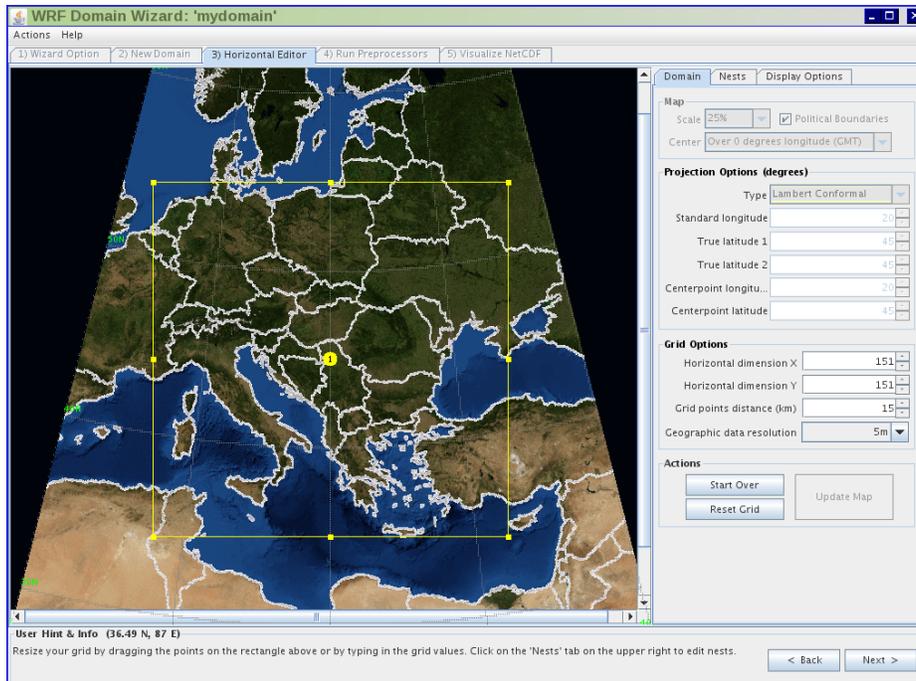
After the projection has been selected and changes to the center point of the primary domain have been made select “**Update Map**”. The DW will pause for a short time while it is “**Rendering Map**”. When the DW returns you see that the window only depicts the primary domain and the projection while the center point controls are locked.

Chapter 05 Just Because You Can – Mastering Your Model Domain



The above image shows a primary domain that has initially been defined with estimated values for the number of horizontal grid points in NX (100) & NY (144) and grid spacing (42.5km). The default number of NX grid points for a new domain is 100 for ARW and 51 for the NMM core. The number of NY grid points is calculated as a function of NX and the surface area covered by the domain-bounding box. These area calculations also determine an initial value for grid spacing.

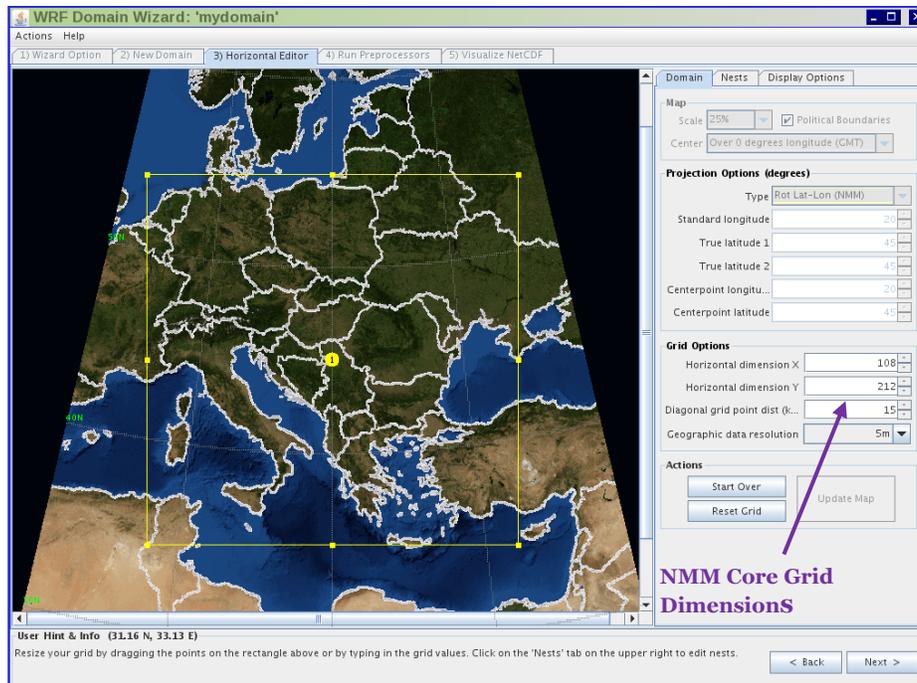
At this point users will likely fine-tune the computational domain for their grid dimensions and spacing. This can be achieved through the “**Grid Options**” section of the domain menu. These values can be changed either by highlighting and editing the settings in the box or by using the increase/decrease buttons. As the changes are made to the primary domain the bounding box that defines the domain also changes. Note that the “**Update Map**” button is no longer available to the users. *If you decrease the size of your domain then the box in the window will also decrease; however, if you increase the domain size the bounding box may extend outside of the areal coverage depicted in the window. That is a bug in DW but your domain is fine.*



The above image shows a primary ARW Lambert Conformal computational domain centered at 20E, 45N with 151 NX and 151 NY grid points and 15 km grid spacing.

Important information regarding the creation of NMM domains: When creating an NMM domain, number of grid points used in the South-North (NY) direction **must always be even**. If the DW allows you to use an odd value, don't do it. Using an odd value for NY will cause your localization to fail, choke, die, and decay into snakes before your eyes. So don't use an odd value for NY!

Due to the rotated Lat-Lon grid (Arakawa-E) used in the NMM core, the relationship between NX, NY, grid spacing, and the areal coverage of the computational domain might not be as intuitive as with the ARW core, which uses a more conventional grid. For example, below is an image of an NMM core domain configuration depicting nearly the same areal coverage and grid spacing (15km) as that was used for the ARW core above.



Notice that while the center point, grid spacing, and total number of grid points is approximately the same (22801; ARW, 22896; NMM) the NX & NY dimensions are significantly different for the NMM core compared to the ARW. This is due to how the grid spacing is defined for the NMM grid. I'll just leave it there rather than going into an explanation of the Arakawa-E grid.

Important information regarding the resolution of static terrestrial data sets:

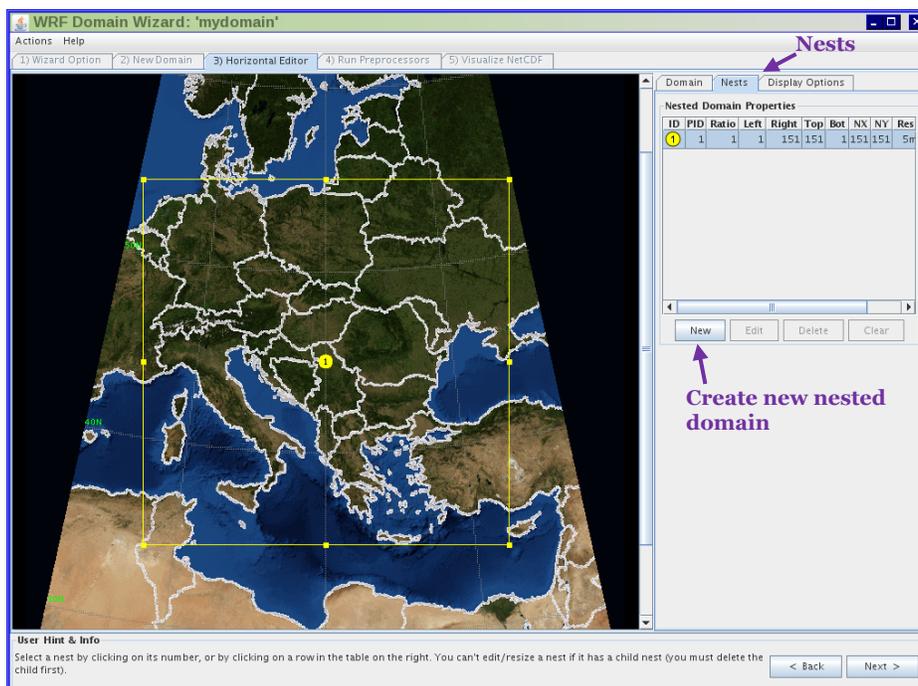
Users may also edit the resolution of the terrestrial data sets being used to generate the static geographic files such as topography and land/sea mask. By default, the WRF EMS DW will select a resolution that is appropriate for the grid spacing used to define your computational domain. Options include 10 minute (~18.5 km), 5 minute (~9.2 km), 2 minute (~3.7 km), and 30 second (~1 km) data sets. If you wish to change the resolution data set used simply select a different value from the “**Geographic data resolution**” menu.

5.2.5 Creating nested domains for the NMM and ARW core

The ARW and NMM cores support concurrent 1- and 2-way nesting, which is neatly integrated into the WRF EMS. For the uninitiated, 2-way nesting allows for information from a higher resolution child domain to be fed back to the 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.

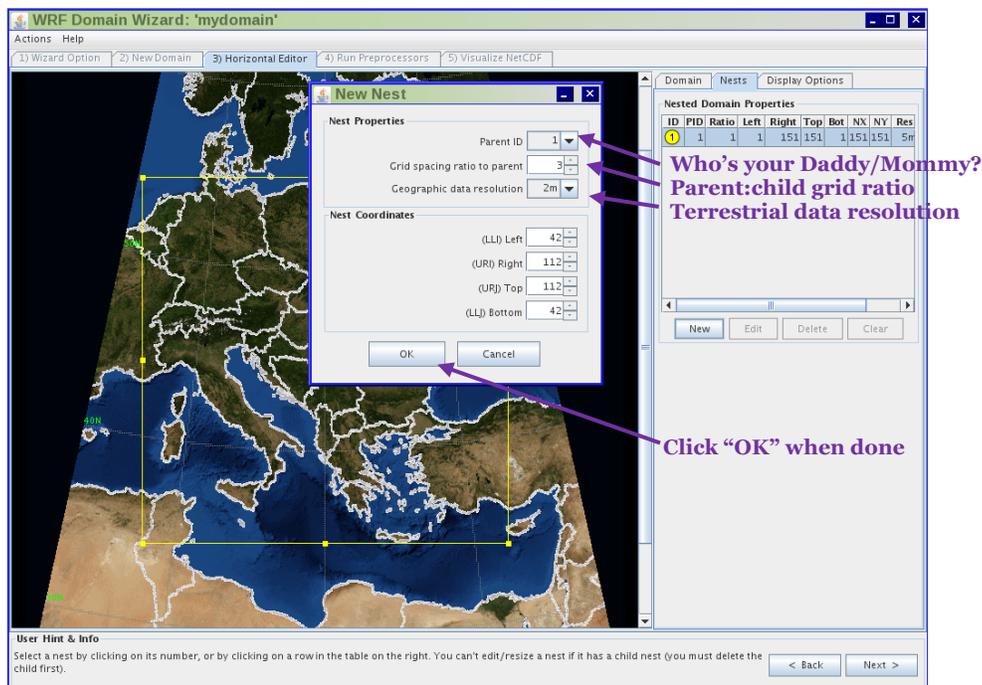
When running the ARW and NMM cores, all nesting is concurrent with the parent domain, meaning the nest runs at the same time as the outer domain. The benefit of concurrent nesting is that the boundary conditions for the nested domain are provided at every parent domain time step. The limitation of this method is that it requires more physical memory on your workstation in order to run multiple domains simultaneously.

The DW may also be used to create nested or sub-domains for your primary computational domain. Once you have settled on your primary domain select the “Nests” tab to display the nested domain configuration tool.



Here you can create new nested domains or edit/delete existing domains. If there are no currently defined nests then only the “New” button will be activated. To create a new domain simple click on the “New” button to bring up the “New Nest” configuration window.

Chapter 05 Just Because You Can – Mastering Your Model Domain

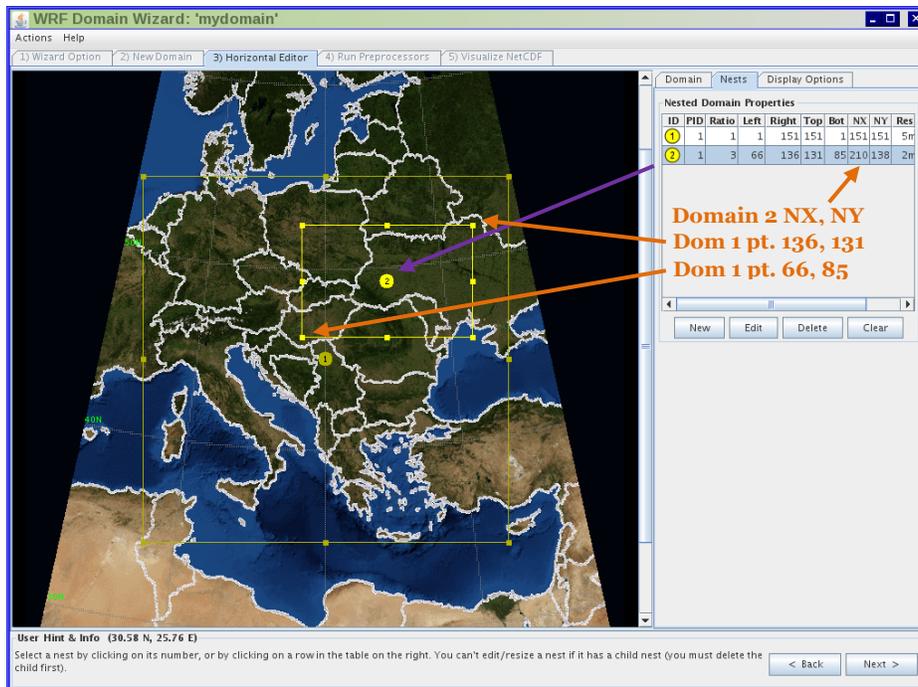


When creating a nested domain, first specify the parent domain with the Parent ID menu. The parent provides the boundary condition updates for the child. Since the domain ID of the primary domain is always 1, the first nested domain you create will have an ID of 2. The second nest you create will have an ID of 3, etc. Note that most values defining a child domain are specified relative (pun intended) to the parent domain. In the above example, the new domain will be 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 of the parent domain. Since the grid spacing for the primary domain (parent) is 15km, the child domain will have a grid spacing of 5km. **Important:** *This value (3) is the recommended grid space ratio although 5 may also be used with reasonable results for all ARW core nested domains. Using an even ratio, i.e., 2, 4, 6, etc. is ok, but may cause problems with some of the feedback within the model and some data processing utilities. Stick with a ratio of 3 or 5; you will be glad you did.*

For the NMM core the only available child:parent ratio is 3:1.

Finally, select the “**Geographic Data Resolution**” for your nested domain. By default, an appropriate value should be used but if you don't like it then change it.



Once these values are chosen, the graphical interface will allow you to manipulate the nested domain using the mouse cursor. A sub domain must be at least 5 grid points inside the parent’s grid boundary. The GUI should ensure that this limit is not exceeded. Each nest is constrained to have its corner points coincident with grid points of its parent domain, which the GUI will also ensure. 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, 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. Be sure to take note of the number of grid points being used by the nested domains. In the above example domain 2 is located with domain 1, which has 151x151 points; however, domain 2 has 210x138 points since for every domain 1 grid point there are 3 for domain 2. *Making nested domains can quickly use up the available memory on your system!*

Here is another example with multiple nested domains

Chapter 05 Just Because You Can – Mastering Your Model Domain

The screenshot shows the WRF Domain Wizard interface. On the left is a map of Europe with four nested domains highlighted in yellow and numbered 1 through 4. On the right is a table titled 'Nested Domain Properties' with columns: ID, PID, Ratio, Left, Right, Top, Bot, NX, NY, Res. Below the table are buttons for 'New', 'Edit', 'Delete', and 'Clear'. At the bottom right, there are '< Back' and 'Next >' buttons. A purple arrow points to the 'Next >' button with the text 'Click "Next" when done'. A 'User Hint & Info' box at the bottom left contains text about selecting and editing domains.

ID	PID	Ratio	Left	Right	Top	Bot	NX	NY	Res
1	1	1	151	151	1	151	151	5m	
2	1	3	66	136	131	85	210	138	2m
3	1	3	20	90	75	18	210	171	30m
4	2	3	56	156	102	38	301	193	30m

User Hint & Info
Select a nest by clicking on its number, or by clicking on a row in the table on the right. You can't edit/resize a nest if it has a child nest (you must delete the child first).

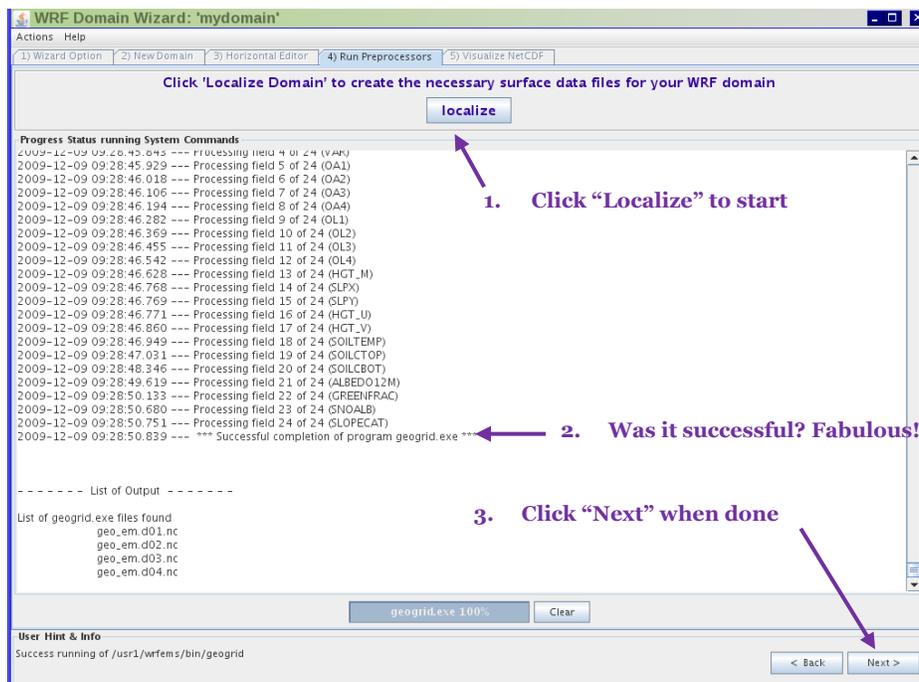
There are a total of 4 domains in the above example, the primary domain, which is always domain 1, and 3 nested domains. Nested domains 2 and 3 are both the children of domain 1 and thus have 5km grid spacing. Nested Domain 4 is the child of domain 2 and thus has $5/3$ or 1.6667 km grid spacing. When you are done creating your domain empire, click on "Next" to run the localization.

5.2.6 Creating domain localizations

Before you can go ahead and use our domains you must run the localization from the “**Run Processors**” window. Localizing the domains serves to extract the terrestrial data over the area defined by the computational domain(s) from the large global files. These localized domain files will reside in the “static” directory and described further in Chapter 6.

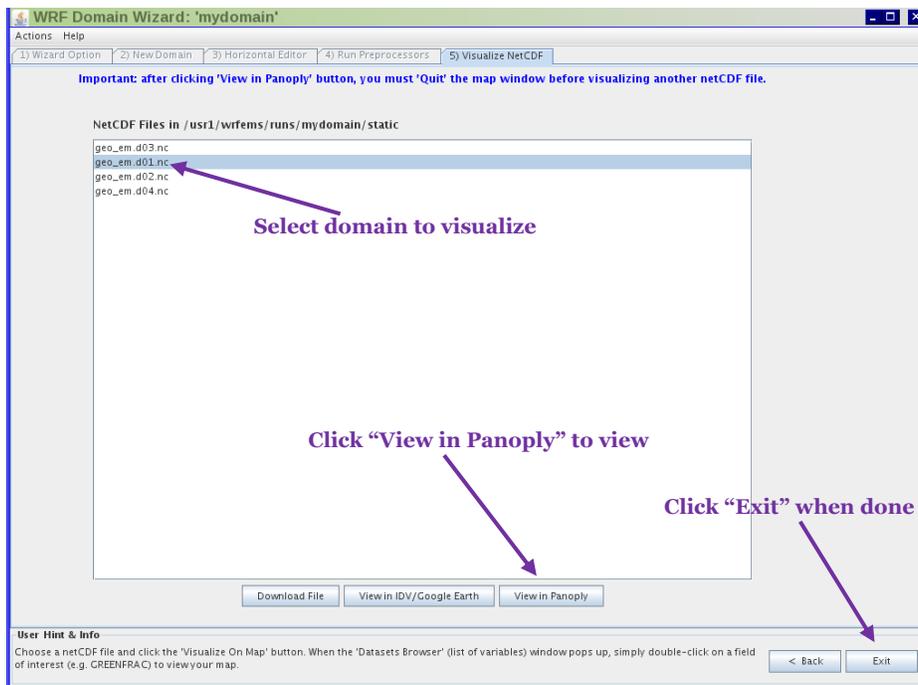
The steps to localization are:

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

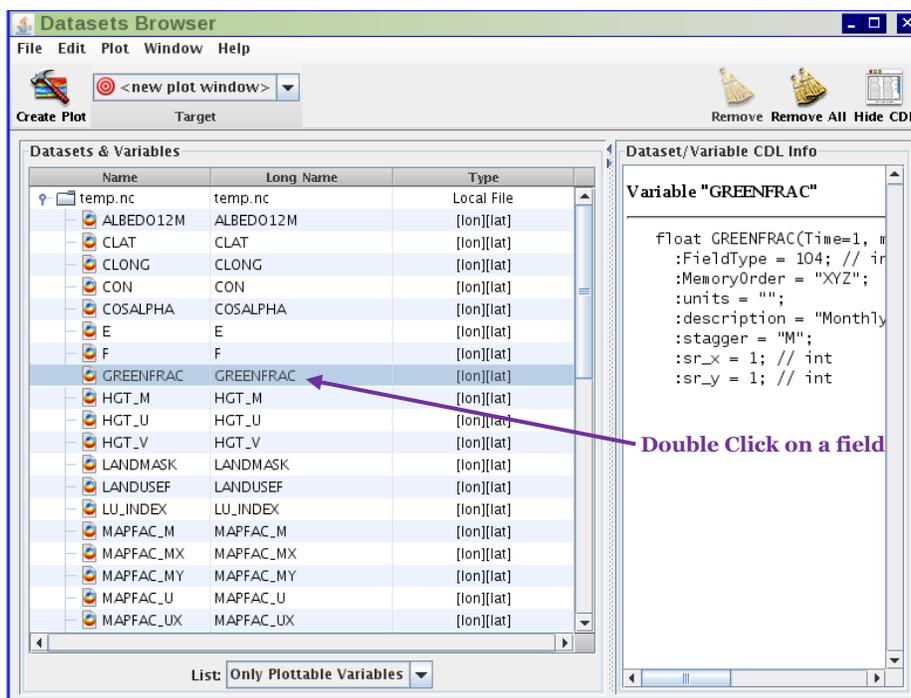


5.2.7 Visualizing the terrestrial data

Provided your localization was successful, once you select the “Next” button you will be able to 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 the purpose of visualizing these data.

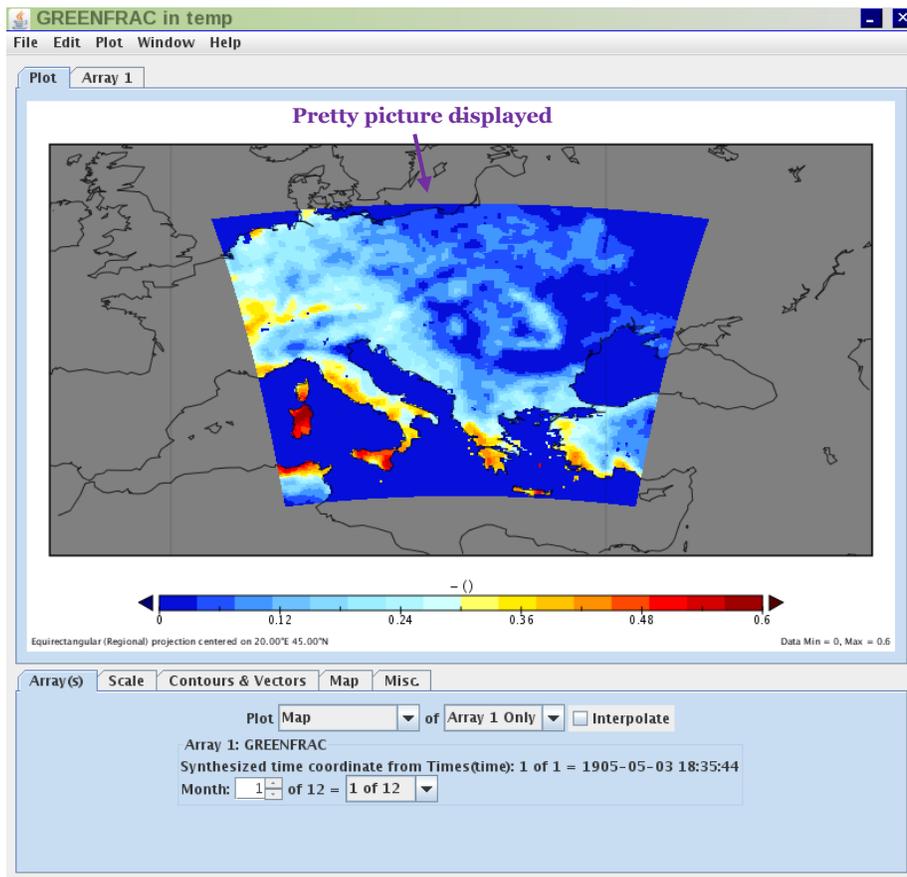


After selecting domain, click “View in Panoply” you should get the following window:



Chapter 05 Just Because You Can – Mastering Your Model Domain

To view your data, simply select the data set and then double click on a field to display:



5.3 Domain creation from the command line

So, running a fancy GUI to create your computational just ain't your thing? You like things "the old fashioned way"? Well, partner, the WRF EMS has you covered.

The WRF EMS includes the `ems_domains.pl` utility that can be used to create an EMS domain and then localize the domain after you have manually configured the grid navigation. Note that this option requires that you know something about defining a WRF domain by editing the namelist files. Some guidance about this is provided here but you may need to consult the ARW and/or NMM users guide located in the `wrfems/docs` directory.

5.3.1 Creating a new domain with `ems_domains.pl`

If you need to manually create a new computational domain you can either copy an existing domain in the "**runs**" directory or use the `ems_domains.pl` utility with the `--newdom` option. The basic syntax for the command is:

```
% ems_domain.pl --newdom <domain name> --core <arw | nmm>
```

A real world example might look like:

```
% ems_domain.pl --newdom mydomain --arw
```

Assuming that the above command was successful you will see a new directory called "mydomain" located in "**wrfems/runs**". In this directory are the default configuration and localization files. This domain is actually created from the ARW|NMM benchmark domain so if that navigation and configuration is OK with you then your work is done here; otherwise, you will need to modify the grid information and relocalize.

5.3.2 Manually configuring the navigation for a new domain

If you have an existing domain that needs some navigation work then you have come to the right place. Note that after you have made any changes to the grid definition or navigation you must localize your domain. See section 5.3.3 for additional details.

Note that all grid definition or navigation changes are made in the "**runs/<domain>/static/namelist.wps**" file.

The following has been "borrowed" with minor edits from the WPS user's guide.

The fields that may be edited are in bold face below. You don't necessarily have to edit all the fields, just those that meet your needs.

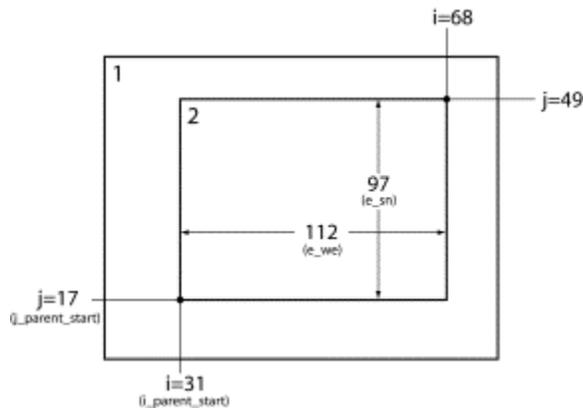
```
&share
  wrf_core           = 'ARW'
  max_dom           = 2
  start_date         = '2008-03-24_12:00:00','2008-03-24_12:00:00'
  end_date           = '2008-03-24_18:00:00','2008-03-24_12:00:00'
  interval_seconds   = 21600
  io_form_geogrid    = 2
/
&geogrid
```

```
parent_id      = 1,1
parent_grid_ratio =1,3
i_parent_start = 1,31
j_parent_start = 1,17,
s_we          = 1,1
e_we          = 74,112
s_sn          = 1,1
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 = '/mmm/users/wrfhelp/WPS_GEOG/'
```

The namelist variables that are affected by nests are shown in the (partial) namelist records above. The example shows namelist variables for a two-domain run (the coarse domain plus a single nest), and the effect on the namelist variables generalize to multiple nests in the obvious way: rather than specifying lists of two values, lists of N values must be specified, where N is the total number of model grids.

In the above example, the first change to the “share” namelist record is to the `max_dom` variable, which must be set to the total number of nests in the simulation, including the coarse domain. Having determined the number of nests, all of the other affected namelist variables must be given a list of N values, one for each grid. The only other change to the “share” namelist record is to the starting and ending times. Here, a starting and ending time must be given for each nest, with the restriction that a nest cannot begin before its parent domain or end after its parent domain; also, it is suggested that nests be given starting and ending times that are identical to the desired starting times of the nest *when running WPS*. This is because the nests get their lateral boundary conditions from their parent domain, and thus, only the initial time for a nest needs to be processed by WPS, except when grid nudging, also called analysis nudging, is used in WRF. It is important to note that, *when running WRF*, the actual starting and ending times for all nests must be given in the WRF namelist.input file.

The remaining changes are to the “geogrid” namelist record. In this record, the parent of each nest must be specified with the `parent_id` variable. Every nest must be a child of exactly one other nest, with the coarse domain being its own parent. Related to the identity of a nest's parent is the nest refinement ratio with respect to its parent, which is given by the `parent_grid_ratio` variable; this ratio determines the nominal grid spacing for a nest in relation to the grid spacing of the its parent.



Next, the lower-left corner of a nest is specified as an (i, j) location in the nest's parent domain; this is done through the `i_parent_start` and `j_parent_start` variables, and the specified location is given with respect to the unstaggered grid. Finally, the dimensions of each nest, in grid points, are given using the `s_we`, `e_we`, `s_sn`, and `e_sn` variables. The nesting setup in our example namelist is illustrated in the figure above shows how each of the above-mentioned variables is determined. Currently, the starting grid point values in the south-north (`s_sn`) and west-east (`s_we`) directions must be specified as 1, and the ending grid point values (`e_sn` and `e_we`) determine, essentially, the full dimensions of the nest; to ensure that the upper-right corner of the nest's grid is coincident with an unstaggered grid point in the parent domain, both `e_we` and `e_sn` must be one greater than some integer multiple of the nesting ratio. Also, for each nest, the resolution of source data to interpolate from is specified with the `geog_data_res` variable.

5.3.3 Manually localizing your new domain

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

```
% ems_domain.pl --locdom [domain name]
```

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

Chapter 6: What’s in each WRF EMS run-time domain directory?

Chapter Contents

6.1 What is a “run-time directory”?

6.2 A tour of a run-time directory

- 6.2.1 The run-time routines
- 6.2.2 The domain sub directories

6.1 What is a “run-time directory”?

When you created a computational domain, either with the Domain Wizard (DW; Chapter 5) or manually, a “**run-time**” directory was created and placed in the location defined by the \$EMS_RUN. By default \$EMS_RUN is defined as “<path>/wrfems/runs” but it can actually reside just about anywhere. The name of the run-time directory should correspond to that name you assigned. For example, if you called your domain something clever such as “mydomain”, then you should find a \$EMS_RUN/mydomain (“wrfems/runs/mydomain”) directory.

Note that you will only run the WRF EMS from one of the run-time directories located under \$EMS_RUN.

6.2 A tour of a run-time directory

In each run-time directory you will find a number of links and sub directories that serve a critical function in the EMS. The following is an explanation the important stuff.

6.2.1 The run-time routines

As stated in Chapter 3, the run-time routines are the heart and soul of the WRF EMS. These are the tools that you will be using 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. You should be kind to the run-time routines and they will be kind to you.

In each run-time directory you will find links to the \$EMS_STRC directory where the run-time routines actually reside. Should these links become broken simply use the “ems_clean” routine to fix the links.

ems_prep Used to identify, acquire, and process the external data sets for use as initial and boundary condition information in the WRF EMS.

ems_run Ingests the output from ems_prep, created the initial and lateral boundary conditions for the model run, and then executes the simulation.

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

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

6.2.2 The domain sub directories

Here is a summary of the run-time sub directories and what they contain

conf Contains the configuration files used for the run-time directory simulations. The configuration files for `ems_run` (Chapter 8), `ems_post` (Chapter 9), and `ems_autorun` (Chapter 10) routines are located beneath the `conf` directory.

grib Contains the files (GRIB 1 & 2) that are used to initialize your model run. These files use the EMS naming convention defined in the various `<dataset>_gribinfo.conf` files (Chapter 7). When `ems_prep` is run, the acquired initialization files are placed in the **grib** directory prior to processing.

log Contains the log files generated from the run-time routines

static The static directory contains a variety of files used by the EMS, including:

- a. The domain-specific static terrestrial data sets that were created when you localized the computational domain. For the ARW core, these files are named “**geo_em.d##.nc**”, where the “##” contains the computational sub (nested) domain for which they were created. For the NMM core, the primary (Domain 1) file is called `geo_nmm.do1.nc` while the NMM core files have names such as “**geo_nmm_nest.l##.nc**”. These files are in netCDF format and may be viewed with the “**ncview**” and “**rdwrfnc**” utilities.
- b. The WRF and WPS namelist files. The namelist files (**namelist.***) are generated and stored in the static directory. Initially you will only see the WPS namelist file (**namelist.wps**) but after running the `ems_run` routine the WRF “**real.exe**” (**namelist.real**) and model (**namelist.wrfm**) namelist files will be created. Note that during execution of the various WRF programs a symbolic link is created from “**namelist.input**” to the proper file in the static directory. *For the most part users will never edit the namelist files directly.*
- c. The **wrfpost_cntrl.parm** and **wrfpost_cntrl_aux.parm** files that controls the fields and levels output to the GRIB files generated when running `ems_post` (Chapter 9).
- d. A **bufr_stations_do1.txt** file that controls the locations/stations of the BUFR soundings when running **ems_post**.

wpsprd Contains the output from the WPS routines when running `ems_prep`.

wrfprd Contains the output from the WRF model when running the **ems_run** routine. All data files are placed in the `wrfprd` directory after **ems_run** has completed.

emsprd Contains the output from the post processing routines when running `ems_post`. If an **emsprd** directory does not exist then one will be created by **ems_post**.

Chapter 7: Meet the WRF EMS `ems_prep.pl` routine

Chapter Contents:

- 7.1 “What is this `ems_prep.pl` thing”?
- 7.2 “How do I make `ems_prep` work for me”?
- 7.3 Some handy dandy `ems_prep` option flags
- 7.4 Meet your `ems_prep` configuration files
 - 7.4.1 The one and only `prep_global.conf`
 - 7.4.2 The `<data set>_gribinfo.conf` files
 - 7.4.3 Meet the `Vtable` files
- 7.5 What are “personal tiles” data sets?
- 7.6 All the `ems_prep` command line options you want to know and love
- 7.7 Just a few `ems_prep` examples
 - 7.7.1 The very basics with `--dset`
 - 7.7.2 Using the `--cycle` option with emotion
 - 7.7.3 Witness the power of `--length`
 - 7.7.4 An example using the `--date` option
 - 7.7.5 Working with multiple initialization data sets
 - 7.7.6 “What do we want? NARR and NNRP data! When do we want it? Now!”

7.1 “What is this `ems_prep.pl` thing”?

The primary purpose of the `ems_prep.pl` routine is to identify, acquire, and process data sets for use as initial and boundary condition information for a WRF simulation. It is the first step on the stairway to modeling nirvana. The `ems_prep.pl` routine is the most complex of the run-time scripts as it must sort through a myriad of user options to determine which data to download, where to obtain the files, how to process the files, and then complete the horizontal interpolation to the user's computational domain. There is a lot going on behind the `ems_prep.pl` curtain but it is in your best interest not to look. *Pay no attention to what goes on behind the curtain.*

As with all the EMS run-time routines, the `ems_prep.pl` file is located in the `wrfems/strc` directory. The user never runs `ems_prep.pl` directly however, but rather, executes the routine from within one of the domain directories by using the `ems_prep` symbolic link. For the sake of clarity, `ems_prep` will be used throughout this chapter to refer to the link that exists in each domain directory.

7.2 “How do I make `ems_prep` work for me”?

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

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

The above might appear a bit confusing at first. A more simplified expression looks something like:

```
% ems_prep --dset <data set> [other stuff]
```

Where the “`--dset <data set>`” is the only mandatory option as it defines the data set(s) to be used for the initial and boundary conditions. More information on the “`--dset`” flag is available from:

% `ems_guide --emsprep dset`

Note that most of the available options, i.e., “**the other stuff**” above, serve to override existing defaults that are defined in the various `<data set>_gribinfo.conf` files, which are located in the `wrfems/conf/grib_info` directory. If you are using `ems_prep` for real-time modeling then it is suggested that you modify the default values in the appropriate files so you only have to pass a minimum number of options. Actually, you shouldn't be running `ems_prep` directly for real-time forecasting purposes. That responsibility is better left to `ems_autorun` (Chapter 10), which runs `ems_prep` for you.

The final output files from `ems_prep` are in netCDF format located in the “`wpsprd`” subdirectory and serve as input to the `ems_run` routine. There are also WPS intermediate files created during this process that contain output of the fields extracted from the GRIB files but they are deleted once they are no longer needed. If it is necessary to retain the intermediate files for troubleshooting purposes then you will have to include the “`--nointrdel`” flag when running `ems_prep`. All `ems_prep` log files are located in the “`logs`” subdirectory

7.3 Some handy dandy `ems_prep` option flags

As with all of the WRF EMS run-time routines, `ems_prep` has a built in help menu and guide. The help menu provides a brief summary of all the options available with `ems_prep`.

% `ems_prep --help`

And for a much more detailed description of a specific option try:

% `ems_guide --emsprep <option>`

If you need a list of all the data sets that are available for model initialization:

% `ems_prep --dslist`

And finally, if you want a summary of the default values for a specific data set:

% `ems_prep --dsquery <data set>`

7.4 Meet your `ems_prep` configuration files

Unlike the other EMS run-time routines, there are no `ems_prep` configuration files that are unique to each computational domain. There are some global configuration files that are used however, and it would be wise for the user (that be you) to at least know that these files exist should a problem arise.

Novice users should not feel intimidated by all the options presented in these files. The files are reasonably well-documented, which should help ease a user's “What/How do I do this?” fueled anxiety.

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

7.4.1 The one and only `prep_global.conf`

The `prep_global.conf` configuration file, located in the `wrfems/conf/ems_prep` directory, contains the data server ID assignments. These IDs are used by `ems_prep` to assign an IP/hostname when attempting to acquire data files from exotic locations whether it's the

workstation in another room or from a national forecast center. If you wish to add another data server to the list of data sources you will need to edit this file but for the most part the sources are updated with each new EMS release.

7.4.2 The `<data set>_gribinfo.conf` files

The `<data set>_gribinfo.conf` files define the default attributes of each data set available for model initialization. In the `wrfems/conf/grib_info` directory you will find files named `<data set>_gribinfo.conf`, where `<data set>` is replaced with a unique key word that identifies that data set. So when you run `ems_prep` and include “`--dset gfs`”, `ems_prep` will look in the “`gfs_gribinfo.conf`” file to get all the default values for the GFS 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. Additional Information on modifying these files or adding new data sets is provided in Appendix A.

Note that almost all of these settings may be overridden by command line options and flags, but more about that topic later in this chapter.

7.4.3 Meet the Vtable files

The Vtables are used to determine which fields to extract from a data set for model initialization. When you download data files for use by `ems_prep`, not all the fields contained in those files are relevant for initializing an NWP model. Thus, the Vtables are used to specify which fields to extract from the GRIB 1 and 2 files. Note that users do not generally need to mess with the Vtables since extracting additional fields does not guarantee that they will be used for model initialization.

The Vtables are located in the `wrfems/data/tables/vtables` directory and each data set has an assigned table defined by the `VTABLE` and `LVTABLE` parameters in each `<data set>_gribinfo.conf` file. Note that multiple data sets use the same Vtable so be careful when making changes to these files.

There are two utilities provided with the WRF EMS that can be used when creating a new or modifying an existing Vtable. The `g1print` and `g2print` routines, which are located in the `wrfems/util/bin` directory, will read a GRIB 1 and 2 file respectively, and dump out information that can be transferred to a Vtable.

7.5 What are “personal tiles” data sets?

To facilitate the acquisition and processing of data for model initialization, WRF EMS users may employ “personal tiles” data sets. Personal tiles are dynamically-generated subsets of high resolution regional and global data sets that have been tailored specifically for operational modeling. These data sets retain the full temporal and spatial resolution of the parent data; however, file size, and consequently bandwidth, have been reduced by as much as 98% for a typical mid-latitude domain. This capability is critical when attempting to establish an operational NWP system in areas where bandwidth is limited.

There is no additional configuration that needs to be completed by the user prior to using personal tiles. Everything is dynamic in that all domain sub setting occurs on the STRC server when a request is made. Changed your domain? No problem. The WRF EMS does all the work so you can relax.

The available personal tile data sets are identified by the “`ptile`” string in the filename and are also singled out when running “`ems_prep --dslist`”. The request for the GFS personal tile data set might

look something like:

```
% ems_prep -dset gfsptiles [more stuff]
```

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

The SOO STRC currently has 3 active personal tile servers available for WRF EMS users and additional servers may be added in the future as demand increases. In addition to providing access to real-time data sets, each personal tile server retains a 3 to 4 week running archive of operational GFS and NAM forecasts for EMS users (See the “`--date`” and “`--cycle`” options). Finally, EMS users have access to an on-line archive of North American Regional Reanalysis (NARR) and Global Reanalysis I and II (NNRP) data running historical cases.

7.6 All the `ems_prep` command line options you want to know and love

There are numerous optional flags and arguments that can be passed to `ems_prep`. These options serve to override existing settings in the configuration files and provide you with control over the initialization step of your simulations.

Probably the most useful option is “`--help`”, which should be obvious in its purpose as it provides a simple listing of the options available to the user. Nonetheless, a somewhat more informative description of each option is provided here; because, after all, this is the “Nearly Complete Guide to the WRF EMS”.

First, here is an index listing the options described in this chapter:

- 7.6.1 Options: `--dset`, `--sfc`, and `--lsm`
- 7.6.2 Option: `--domain`
- 7.6.3 Option: `--clean` and `--allclean`
- 7.6.4 Option: `--tiles`
- 7.6.5 Option: `--date`
- 7.6.6 Option: `--cycle`
- 7.6.7 Option: `--besthr` and `--synchr`
- 7.6.8 Option: `--local`
- 7.6.9 Option: `--length`
- 7.6.10 Option: `--previous`
- 7.6.11 Option: `--analysis`
- 7.6.12 Option: `--nodelay`
- 7.6.13 Option: `--hiresbc`
- 7.6.14 Option: `--gribdir`
- 7.6.15 Option: `--rundir`
- 7.6.16 Option: `--[no]gribdel`
- 7.6.17 Option: `--[no]intrdel`
- 7.6.18 Option: `--noprocs`
- 7.6.19 Option: `--nometgrid`
- 7.6.20 Option: `--nodegrib`
- 7.6.21 Option: `--query` or `--dsquery`
- 7.6.22 Option: `--dlist`
- 7.6.23 Option: `--[no]dcheck`

7.6.1 Options: `--dset`, `--sfc`, and `--lsm`

What I Do: Manage the initialization data sets

Usage:

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

Or

```
--sfc dataset [:METHOD:SOURCE:LOCATION],[DATASET[:METHOD:SOURCE:LOCATION]],...
```

Or

```
--lsm dataset [:METHOD:SOURCE:LOCATION],[DATASET[:METHOD:SOURCE:LOCATION]],...
```

Description:

In its simplest usage, the `--dset`, `--sfc`, and `--lsm` options specify the data set to use for initial and boundary conditions (`--dset`), static surface fields (`--sfc`), and land-surface fields (`--lsm`) respectively.

Note that "`--dset dataset`" is the only mandatory option and oxymoron in `ems_prep`; well, there may be other oxymora. Use of the `--sfc` and `--lsm` options are, of course, optional.

The behavior of these options is inextricably tied to the parameters and settings found in the respective `<data set>_gribinfo.conf` files. In particular, the **SERVERS** section (Appendix A), which defines the source(s) for the data files. Thus, to fully understand the functionality of these options, you should review the **SERVERS** section of a `<data set>_gribinfo.conf` file. Just pick one to read, they all say the same thing.

In its simplest usage:

```
% ems_prep --dset dataset
```

Or for example,

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

This tells `ems_prep` to use the `gfs` data set, defined in the `gfs_gribinfo.conf` file, as initial and boundary conditions for your model run. All default settings will be used unless you override them through the use of arguments to `--dset` and/or other command-line options.

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/MT.gfs_CY.CC/RD.YYYYMMDD/PT.grid_DF.gr2/fh.oOFF_tl.press_gr.op5deg

SERVER-HTTP = STRC:/data/grib/YYYYMMDD/gfs/grib.tCCz/YMMDDCC.gfs.tCCz.pgrb2fFF
SERVER-HTTP = TOC:/data/MT.gfs_CY.CC/RD.YYYYMMDD/PT.grid_DF.gr2/fh.oOFF_tl.press_gr.op5deg

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

The above entries in the `gfs_gribinfo.conf` file indicate two FTP sources for initialization files, NCEP and TGFTP, two HTTP sources, STRC and TOC, and two NFS sources, DATA1 and a local source. The server IDs, i.e., NCEP, TGFTP, STRC, TOC, and DATA1 correspond to predefined hostnames or IP addressed located in the `wrfems/data/conf/ems_prep/prep_global.conf` file. The NFS entry without a server ID specifies the location of the `gfs` files on the local system. So why

have entries for remote servers when the data can be obtained locally? Well, this is for demonstration purposes only; closed course with professional driver.

So, if you were to run the "`ems_prep --dset gfs`" command as in the example above, `ems_prep` would use the information provided in the `SERVERS` section of `gfs_gribinfo.conf` file to acquire the initialization files. The `ems_prep` routine would first use the `FTP` command to access data on the servers identified by `NCEP` and `TGFTP`. If unsuccessful, `ems_prep` will attempt at access files via `http` and then finally use secure copy (`SCP`) to obtain data from the `DATA1` server and copy (`cp`) from the local system.

Yes, `ems_prep` will attempt every possible source identified in the `gfs_gribinfo.conf` file. *Working harder so you can slack off would be `ems_prep`'s motto if it had one, which it doesn't.*

Optional Arguments

There are arguments that may be passed to the `--dset` option 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 location and naming convention used at the remote server respectively.

METHOD

`METHOD` is used to override the default behavior of using each method, `ftp`, `http`, and/or `nfs` specified in the `SERVERS` section to acquire data. `METHOD` can take the form of:

```
nfs    Only use the SERVER-NFS entries in the <data set>_gribinfo.conf file.  
ftp    Only use the SERVER-FTP entries in the <data set>_gribinfo.conf file.  
http   Only use the SERVER-HTTP entries in the <data set>_gribinfo.conf file.
```

```
nonfs Don't use the SERVER-NFS entries in the <data set>_gribinfo.conf file.  
noftp Don't use the SERVER-FTP entries in the <data set>_gribinfo.conf file.  
nohttp Don't use the SERVER-HTTP entries in the <data set>_gribinfo.conf file.
```

```
none   Don't use any of the methods listed in the SERVERS section. All files are assumed to  
        reside and correctly named in <domain>/grib directory.
```

Here are a couple 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 located data from `NCEP` and `TGFTP` it will not use the other methods listed.

```
% ems_prep --dset gfs:noftp
```

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

```
% ems_prep --dset gfs:none
```

Translation: Do not attempt to acquire the files as they already reside and are correctly named in the `<domain>/grib` directory. Note that the same behavior may be achieved by commenting out or deleting all the `SERVER-` entries in the `gfs_gribinfo.conf` file OR by passing:

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

SOURCE

SOURCE 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
```

This tells **ems_prep** to only acquire 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 as specified in the `gfs_gribinfo.conf` file.

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
```

To acquire files locally that do not have a **SOURCE** associated with them in the `<data set>_gribinfo.conf`, such as in the last `SERVER-NFS` entry above, use `"local"`:

```
% ems_prep --dset gfs:nfs:local
```

Using **METHOD**, **SOURCE**, and **LOCATION** together, just like one big, happy family, only different

You may also use **SOURCE** to specify a hostname or IP address. This is best done in conjunction with both the **METHOD** and **LOCATION** argument. By using all three arguments you can request that initialization files be acquired from a location not listed in `<data set>_gribinfo.conf`. The format looks similar to the `SERVER-` entries:

```
% 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
```

Notice that all the examples above are equivalent, provided that there is a `NOMAD6` entry as a server ID in the `prep_global.conf` file. In the above example you must specify a **METHOD**; otherwise something will fail.

Using Multiple Data Sets

The `--dset` option can be used to specify separate data sets for initial and boundary conditions in your model runs. For example, if you wish to use the 12km NAM files as the initial and 0.5 degree GFS for your boundary conditions, simply separate the two data sets with a `"%"` in the dataset argument to `--dset`, i.e.,

```
% ems_prep --dset nam218%gfs --length 24
```

In which case `ems_prep` will attempt to acquire a single `nam218` file to use as the initial conditions and the `gfs` will be used for the boundary conditions through 24 hours. Note that the `"--length"` option must be used when specifying multiple data sets.

All the optional arguments detailed ad nauseam above are available for use with multiple data sets as well. For example; knock yourself out with such classics as:

```
% ems_prep --dset nam218:http:gfs::strc --length 36
```

Translation: Only use the `SERVER-HTTP` entries in the `nam218_gribinfo.conf` file to acquire 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 is the classic:

```
% ems_prep --dset nam218:http:strc:/pub/nam.tCCz.pgrbfff:gfs:nfs:local:/data/gfs/YYYYMMDD/gfs.tCCz.pgrb2fff --length 180
```

Translation: Override the `nam218_gribinfo.conf` `SERVER` entries and obtain the `nam218` file via `http` from the source identified by `STRC` in the `prep_global.conf` file and located in the `/pub/nam` directory with the naming convention of `nam.tCCz.pgrbfff`. Also, override the `gfs_gribinfo.conf` `SERVER` entries and copy (`cp`) the `gfs` files through 180 hours that are located in the `/data/gfs/YYYYMMDD` directory.

Using Static Surface (`--sfc`) and Land Surface Data Sets (`--lsm`)

There are two additional options for the acquisition and processing of surface and land-surface data set named `"--sfc"` and `"--lsm"` respectively. These options 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]],...
```

where multiple data sets are separated by a comma (",") and not a "%" as with `--dset`.

When multiple complementary data sets are specified, such as:

```
--sfc sstpile,ssthr
```

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

The `--sfc` option allows users to acquire and process near surface fields that will be used to replace or augment a field or fields in the initialization data set as defined by `--dset`. The fields identified will be held constant throughout the model simulation. An example of using the `--sfc` option would be:

```
% ems_prep --dset gfs --sfc ssthr:ftp:polar
```

Which would replace the SST fields in the GFS data set with the 1/12th degree SST field from the polar ftp server.

7.6.2 Option: `--domains`

What I Do: Provide control over domains included in simulation

Usage: `% ems_prep --domains domain1[:START HOUR],...,domainN[:START HOUR]`
Where $N \leq \text{Max Domains}$

Description:

Passing the "`--domains`" option defines a list of (nested) domain(s) to initialize when running a simulation. Any domain(s) must have been defined and localized previously while running the GUI. If you created any sub domains (multiple layers of nests), then passing the `--domains` option will activate them. You will not be able to run a nested simulation unless you activate the sub domains!

NOTE: Domain 1 is the Mother or Parent of all domains and is always included by default.

If three nested domains were created and you wish to activate all three:

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

Note that by requesting domain 4, domains 2 and 3 will automatically be activated!

So, why specify multiple domains (2, 3, and 4) when only one (4) will accomplish the same task? Because you can also control the start time of the 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).

Here is a hypothetical configuration to demonstrate the power of the `--domains` option. Let's assume that you have created 10 domains (1 Parent of all domains and 9 sub domains with the following configuration:

```

                DOMAIN 1
                -----
    Domain 2    Domain 3    Domain 6
    Domain 4    Domain 9    Domain 7
    Domain 5    Domain 10   Domain 8
```

In the above configuration, Domain 1 is the parent of domains 2, 3, and 6. Domain 2 is the parent of domain 4, domain 9 the parent of 10, etc.

So, if you were to pass "`--domains 9`" to `ems_prep`, then domains (1), 2,3,4,5,6,7,8 would also be included in the initialization.

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 hours AFTER the start of the main simulation, you would then include the `START HOUR` in your argument list to `--domains`:

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

Again, although not explicitly included in the list, domains 3, 4, and 5 are implicitly included and thus will also be initialized.

The above command will initialize domains 2, 4, and 5 to start 9 hours *after* domain 1. Domain 6 will begin 12 hours after domain 1; however, domain 3 will start *at the same time* as domain 1. This is because, unless otherwise specified, included sub domains will be initialized to start at the same time as the parent!

Here is a more complicated example:

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

Ok, lots of stuff going on here. First note that domain 10 is not implicitly or explicitly included in the list so it will not be run. Also:

- **Domain 2** is (explicitly) initialized to start 3 hours after the start of domain 1
- **Domain 4** is (implicitly) initialized to start 3 hours after the start of domain 1
- **Domain 5** is (explicitly) initialized to start 15 hours after the start of domain 1

- **Domain 3** is (explicitly) initialized to start 12 hours after the start of domain 1
- **Domain 9** is (explicitly) initialized to start 6 hours after the start of domain 1; however, since this start hour is BEFORE the parent domain, 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 0 hours after the start of domain 1
- **Domain 7** is (explicitly) initialized to start 6 hours after the start of domain 1
- **Domain 8** is (explicitly) initialized to start 24 hours after the start of domain 1; however, since the total length of the domain 1 simulation is 24 hours, *Domain 8 will be turned off!*

7.6.3 Option: `--clean` and `--allclean`

What I Do: Specifies the amount of directory cleaning before run

Usage: % `ems_prep --clean`
Or
% `ems_prep --allclean`

Description:

The `--clean` flag serves to scour the domain directory of any evidence that a previous simulation had been attempted **and then exit**. All initialization, forecast, and log files will be removed but the files will remain in the grib directory. In addition, new links to the run-time scripts will be created. This option is the same as running “`ems_clean -level 3`”.

The `--allclean` flag behaves similarly to `--clean` except that any files in the “`grib`” directory will all be removed. This option is the same as running “`ems_clean -level 4`”.

The default behavior when running `ems_prep` is similar to passing the “`--clean`” flag except that the routine does not exit and continues on its merry way. This is probably what you want anyway.

7.6.4 Option: `--tiles`

What I Do: Specify the NCEP tiles to acquire and process for model initialization

Usage: % `ems_prep --tiles tile1[,tile2[,tile3[,...]]]`

Description:

The argument list for the "`--tiles`" option is a series of NCEP tile numbers separated by a comma (,) and without spaces. By passing the tiles option you will override the default values in the `<data set>_gribinfo.conf` file. This option will only work if the requested data set uses tiles; otherwise, it will be ignored. As an example:

```
% ems_prep --dset tile12 --tiles 31,32,22,23 (yes)
```

```
% ems_prep --dset tile12 --tiles 31, 32, 22, 23 (no)
```

If you do not know what tiles you should be using then look at the `tile12` or `tile32` images in the `/usr1/wrfems/docs` directory.

7.6.5 Option: `--date`

What I Do: Specify the date for the model initialization

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

Description:

Passing the "`--date`" option defines the initialization date of the simulation for the primary domain (domain 1). This date will also be used for any sub (nested) domain unless it is overridden with the "`--domains`" option.

The argument to `--date` is a 4- or 2-digit year, 2-digit month (01 to 12), and 2-digit day (01 to 36). Not passing the "`--date`" option will cause `ems_prep` to use the current system date.

Important: This option should **not** be used for real-time modeling purposes.

7.6.6 Option: `--cycle`

What I Do: Specify the cycle time, start hour, end hour, and frequency of the initialization data used in your simulation.

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

Description:

Note: This option should NOT be used for real-time modeling purposes

The `--cycle` option defines the cycle time of the model data set to use for initialization of your simulation. The general usage is:

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

Not passing the `--cycle` option will cause `ems_prep` to use the cycle time of the most recent model run from which data are available. To determine the cycle time of the most recent run,

`ems_prep` accounts for the amount of time required to run the operational model and process the forecast files for distribution.

For example, if it takes NCEP three hours to run the model and process grib files then `ems_prep` will not attempt to obtain data from the 12Z run until after 15Z.

The list of available cycles and the delay time for each data set are specified in the `<dataset>_gribinfo.conf` file by the **CYCLES** and **DELAY** parameters respectively.

Users Note: If you are making real-time model runs then you should **not** use `--cycle`.

The `--cycle` option also accepts 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 `<data set>_gribinfo.conf` file as **INITFH**, **FINLFH**, and **FREQFH** respectively.

The format for the argument list is:

```
% ems_prep --dset <dataset> --cycle CYCLE[:INITFH:FINLFH:FREQFH]
```

Notice that the values for **INITFH**, **FINLFH**, and **FREQFH** are separated by colons (:). Here are a few examples. Feel free to make some of your own.

```
% ems_prep --dset <dataset> --cycle 00:00:24:03
```

Translation: Use the 00Z 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. The script will attempt to download a 00, 03,06,09,12,15,18,21, and 24 hour forecast files. All default values are overridden.

```
% ems_prep --dset <dataset> --cycle 06:06:30
```

Translation: Use the 06Z cycle time, the 06 hour forecast for the initialization time, the 30 hour forecast for the final boundary condition time (thus a 24 hour forecast), and use the default value in `<data set>_gribinfo.conf` (**FREQFH**) for the BC frequency.

```
% ems_prep --dset <dataset> --cycle CYCLE:INITFH:36:12
```

Translation: Use the default cycle time (current), the default forecast hour for the initialization time, the 36 forecast for the final boundary condition time, and use 12-hourly BC files.

Finally (phew!), you actually don't need to include the **CYCLE**, **INITFH**, **FINLFH**, or **FREQFH** placeholders. You can just use the colons (:) without anything in between. For example:

```
% ems_prep --dset <dataset> --cycle :::12
```

Or

```
% ems_prep --dset <dataset> --cycle :24::06
```

Translations: Use the default cycle time (current), the default forecast hour for the initialization time, the default forecast hour for the final boundary condition time, and use 12-hourly BC files.

In the second example, Use the default cycle time (current), the 24 hour forecast for the initialization time, the default forecast hour for final boundary condition time, and use 6-hourly BC files.

You have the power, use it wisely.

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

Also note: The **DELAY** setting in `<data set>_gribinfo.conf` file can be overridden with the "**--nodelay**" option.

Important: For the final time, *this option should not be used for real-time modeling purposes.*

7.6.7 Option: **--besthr** and **--synchr**

What I Do: List of static surface data sets to match the surface data set hour with the closest cycle hour

Usage: % `ems_prep` [other stuff] **--synchr** `<sfc data set>`,`<sfc data set>`,...

Description:

Passing **--synchr** tells the WRF EMS to only use the cycle that is closest to the initialization cycle. It will not use the other cycles when looking for data, but rather, will look for the same cycle hour over the period of days defined by AGED parameter defined in the `<data set>_gribinfo.conf` file.

The argument to **--synchr** is the list of static surface data sets, separated by a comma, to apply this restriction and overrides the BESTHR field, if any, in the `<dataset>_gribinfo.conf` file.

Why do this?

Some data sets, such as MODIS SSTs have a diurnal variation that needs to be taken into account when initializing a simulation. It may not be appropriate to use a data set from 00Z for a simulation at 12Z even if that is the most recent time available. If **--synchr** is passed then the WRF EMS will look for 12Z from the previous day. Note that the data set cycle times do not have to match the simulation initialization time. The WRF EMS will determine the closest cycle hour to the initialization hour and use that value.

For use with **--sfc** data sets only!

7.6.8 Option: **--local**

What I Do: Tells `ems_prep` to only look in the "grib" directory for initialization files

Usage: % `ems_prep` [other stuff] **--local**

Description:

Pass the **--local** option if you want the `ems_prep` routine to **only** look in the local grib directory for the initialization files. This option does the same thing as specifying "**--dset <data set>:local**" but I forgot I included that functionality and added **--local**. So enjoy them both together, I would!

7.6.9 Option: `--length`

What I Do: Specifies the simulation length for the primary domain

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

Description:

Passing the "`--length`" option overrides the `FINLFH` (See: "`--cycle`" option) value with a value that would result in the requested forecast length (hours). The `--length` option usurps the forecast length defined by the `--cycle` option, so passing:

```
% ems_prep --dset <dataset> --cycle 00:06:30:03 --length 36
```

Is the same as passing:

```
% ems_prep --dset <dataset> --cycle 00:06:42:03
```

The "`--length`" option overrides everything when defining the length of your forecast. The "`--length`" option is king. All hail "`--length`"!

Important: *The `--length` option must be used when specifying separate data sets for initial and boundary conditions with the `--dset` option. That's all I have to say about length.*

7.6.10 Option: `--previous`

What I Do: Tells `ems_prep` to use the previous cycle time rather than the current one

Usage: % `ems_prep` [other stuff] `--previous`

Description:

Passing the "`--previous`" option will instruct `ems_prep` to use initialization data from the previous cycle time rather than the current cycle. This option is primarily used as a failover when using the `ems_autorun` routine and should not be attempted by mortals unless you think you know what you are doing.

7.6.11 Option: `--analysis`

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

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

Description:

Passing the "`--analysis`" option will result in your simulation being initialized from a series of analyses rather than a previous forecast. If you are using data sets such as the North American Regional Reanalysis (NARR), then you want to include `--analysis`. This option should also work when initializing your simulation from a series of 00-hour forecasts such as from the NAM or GFS.

But wait, there is more. What if you're feeling silly and you want to initialize your simulation from a series of X-hour (non zero-hour) forecasts? Well then you can simply pass an argument to the `--analysis` option like:

% **ems_prep** [other stuff] **--analysis 24**

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 first get a good night's rest as chances are that you are suffering from sleep deprivation. This option will still be there in the morning. Sweet dreams.

7.6.12 Option: **--nodelay**

What I Do: Override the **DELAY** setting in the `<dataset>_gribinfo.conf` file

Usage: % **ems_prep** [other stuff] **--nodelay**

Description:

Passing "**--nodelay**" option will turn off (set to 0 hours) the default **DELAY** value defined in each `<data set>_gribinfo.conf` file being accessed for the current simulation. Using the **--nodelay** option is kind of like saying "I want it now!" even though it might be fruitless.

7.6.13 Option: **--hiresbc**

What I Do: Requests high-resolution boundary condition update files

Usage: % **ems_prep** [other stuff] **--hiresbc**

Description:

Passing the "**--hiresbc**" option request that 1-hourly boundary condition update files be created instead of the default frequency, which would be the same as the frequency of the files downloaded for model initialization. This option serves to provide no benefit to the model simulation, but rather, will allow:

1. The user to run a simulation for a period less than that covered between the 00-hour forecast and first boundary condition update.
2. For the restarting of a simulation between normal boundary condition update times

Confused? An example would be if your boundary condition files were 3-hourly but you wanted to run a 2 hour simulation. Passing **--hiresbc** would allow you to accomplish this task.

7.6.14 Option: **--gribdir**

What I Do: Alternate location of the "grib" directory

Usage: % **ems_prep** [other stuff] **--gribdir <path to grib files>**

Description:

Passing the "**--gribdir**" option overrides the default location of the gridded files, on the local system, used to initialize a model simulation. This is not to be confused with the data directory specified by the "**--dset <dataset>:nfs**" option, but rather, overrides the default location of `wrfems/runs/<domain>/grib`. This option is primarily for internal use by **ems_autorun**.

7.6.15 Option: `--rundir`

What I Do: Specifies the run-time domain to use

Usage: % `ems_prep` [other stuff] `--rundir` <simulation run time domain directory>

Description:

Pass the "`--rundir`" option if you want to define the location of the domain directory. This option is typically used by the `ems_autorun` routine and is not recommended for use by the user, although this suggestion will probably not stop most of you.

Note that the domain directory must exist or `ems_prep` will terminate. Not passing `--rundir` will set the default as the current working directory, which is probably what you want when running from the command line. So again, it is probably in your best interest to leave this option alone.

7.6.16 Option: `--[no]gribdel`

What I Do: [Does not] scour grib files after processing

Usage: % `ems_prep` [other stuff] `--[no]gribdel`

Description:

Pass the `--[no]gribdel` option if you [no not] want the processed grib files scoured from the run directory following successful completion of WPS. Not passing `--[no]gribdel` results in the default behavior on not deleting the GRIB files.

7.6.17 Option: `--[no]intrdel`

What I Do: [Does not] scour WRF intermediate files after processing

Usage: % `ems_prep` [other stuff] `--[no]intrdel`

Description:

Pass the `--[no]intrdel` option if you [no not] want the processed WRF intermediate files scoured from the "`wpsprd`" directory following successful completion of WPS. Not passing `--[no]intrdel` results in the default behavior on not deleting the intermediate files.

The WRF intermediate format files are simply the selected contents on the GRIB files written out in a binary format. Raw LAPS files are in WRF intermediate format. Typically, you do not need to keep the intermediate files around as they are further processed by the `metgrid` program and written out to WRF netCDF and placed in the in the "`wpsprd`" directory. However, there are times when you need to do some troubleshooting and may need these files. Not passing `--nointrdel` will result in the default behavior of the files being deleted.

The contents of the WRF intermediate files may be viewed with the "`rdwrfin`" utility, which is provided in the `wrfems/util/bin` directory.

7.6.18 Option: `--noproc`

What I Do: Tells `ems_prep` to not process GRIB files after acquisition

Usage: % `ems_prep` [other stuff] `--noproc`

Description:

Passing the "`--noproc`" option will instruct `ems_prep` to download the requested initialization files to the local system but do not process them for use with the model.

7.6.19 Option: `--nometgrid`

What I Do: Tells `ems_prep` to exit just before running the `metgrid` program

Usage: % `ems_prep` [other stuff] `--nometgrid`

Description:

Passing "`--nometgrid`" tells `ems_prep` to exit just before running of the `metgrid` routine, which is used to accomplish the horizontal interpolation of the initialization grids to the computational domain. Passing `--nometgrid` will leave all the `metgrid` input files in the domain directory for inspection in the event of a problem. Not for the casual user.

7.6.20 Option: `--nodegrib`

What I Do: Tells `ems_prep` to exit just before running the `degrib` program

Usage: % `ems_prep` [other stuff] `--nodegrib`

Description:

Passing the "`--nodegrib`" option tells `ems_prep` to exit just before running of the `degrib` routine, which is used to unpack the initialization files into the WPS intermediate format. Passing `--nodegrib` will leave all the `degrib` input files in the domain directory for inspection in the event of a problem. Also not for the casual user.

7.6.21 Option: `--query` or `--dsquery`

What I Do: Queries the contents of a `<dataset>_gribinfo.conf` file

Usage: % `ems_prep` `--dsquery` `<dataset>`

Description:

Using the `--dsquery` option allows you to interrogate the contents of a `<data set>_gribinfo.conf` file, providing information such as default settings, fine naming conventions, and servers where the data set can be found. The argument to `--dsquery` is the moniker used to identify a specific data set, a list of which is provided by the "`--dslist`" option. Lots of good information.

Here is an example of the information for the `nam212` data set:

```
% ems_prep --dsquery nam212
```

Default settings from the `nam212` grib information file: `nam212_gribinfo.conf`

NAM Model 40km resolution 212 grid - Isobaric coordinate (25mb) - 3hourly through hr 84 (7 Mb per file)

```
Default Initial Forecast Hour : 00
Default Final Forecast Hour  : 24
Default BC Frequency         : 03
Available Cycles             : 00 06 12 18
Remote Server Delay          : 03 Hour(s)
Local Filename (GRIB 2)     : YYYYMMDDCC.nam.tCCz.awip3dFF.tmoo.grb2
Vtable Grid Information File : Vtable.NAM
LVtable Grid Information File: Vtable.NAMLSM
```

METHOD	HOST	LOCATION
ftp	NCEP	/pub/data/nam.YYYYMMDD/nam.tCCz.awip3dFF.tmoo.grib2
ftp	STRC	/data/grib/YYYYMMDD/nam/grid212/grib.tCCz/nam.tCCz.awip3dFF.tmoo.grb2
ftp	TGFTP	/data/MT.nam_CY.CC/RD.YYYYMMDD/fh.o0FF_tl.press_gr.awip3d
http	STRC	/data/grib/YYYYMMDD/nam/grid212/grib.tCCz/nam.tCCz.awip3dFF.tmoo.grb2
http	TOC	/data/RD.YYYYMMDD/PT.grid_DF.gr2/fh.o0FF_tl.press_gr.awip3d

7.6.22 Option: `--dslist`

What I Do: Prides summary of available initialization data sets

Usage: % `ems_prep --dslist`

Description:

Passing the "`--dslist`" option provides the user with a brief summary of available data sets for initializing a simulation. If this option is passed then `ems_prep` will print the **INFO** and **CATEGORY** sections of each `<data set>_gribinfo.conf` file along with the data set moniker used as an argument to the "`--dset`" option.

7.6.23 Option: `--[no]dcheck`

What I Do: Does [not] do checking of available disk space prior to running the model

Usage: % `ems_prep [other stuff] --[no]dcheck`

Description:

Note that this option overrides the global environment variable **DSKCHECK** found in the **EMS.eshrc** file.

Passing the `--[no]dcheck` option turns on [off] the checking of available disk space prior to running the model. This option allows the user to monitor disk space usage to on the partition where the EMS resides. Running an NWP model can generate a lot of data and if it is not monitored a disk can fill up with less-than-desirable consequences. Passing `--dcheck` will result in the EMS checking available space before the start of each of the run-time routines. If space is limited (greater than 95% usage) a warning message will be printed to the screen. The routine will also automatically terminate your simulation and provide you with a message (via email too) when your usage reaches 100%. That's the way it's supposed to work at least.

7.7 Just a few `ems_prep` examples

Fortunately, there are many command line options available to `ems_prep` that may be used to override the default settings. If the minimalist approach is not for you, then feel free to read the information provided by 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 very basics with `--dset`

There are many command line options that you can pass to `ems_prep`, but only one mandatory argument, which is the data set(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 nam212
```

Following which `ems_prep` will attempt to use the NAM 212 grid for model initialization and boundary conditions. In the absence of any other options, `ems_prep` will default to the settings in the `nam212_gribinfo.conf` file. The `ems_prep` routine will check the “`grib`” directory for any existing files identified by the default naming convention specified by the `LOCLFIL` parameter and then look for external sources. Because no method of acquisition is specified (`nfs`, `ftp`, or `http`), `ems_prep` will go to each source listed in the `SERVERS` section of the `grib_info` file.

Note: If you have placed your `grib` files in the `<domain>/grib` directory but `ems_prep` fails to find them, the most likely problem is that you are not using the proper filenames. Look at the `ems_prep` error messages and the `LOCLFIL` parameter for more information.

Here's another example:

```
% ems_prep --dset nam212:ftp
```

In the above example, the user is requesting the most recent NAM 40km 212 grid forecast GRIB files for initial and boundary conditions. The `ems_prep` will use the initial forecast hour, final forecast hour, and boundary condition frequency defined by the `INITFH`, `FINLFH`, and `FREQFH` parameters respectively in the `nam212_gribinfo.conf` file. It will initially look in the local “`grib`” directory before searching for the data set on the available FTP servers listed in the `SERVERS` section of the same configuration file. The routine will check each FTP server for the requested files until all of them have been downloaded to the local machine and renamed to the file naming convention defined by `LOCLFIL`. Note that if the FTP server is not anonymous then you must have the user name and password information in your `~/.netrc` file.

Kicking it up a notch with:

```
% ems_prep --dset nam212:nfs::/data/NAM.tCCz.awip3dFF
```

Translation: The above example is similar to the initial example except it accesses the data set via the `copy (cp)` command on a local drive (`nfs`), and defines the location (`/data`) and naming convention for the files (`NAM.tCCz.awip3dFF`). See the `<data set>_gribinfo.conf` file for an explanation of the filename place holders, i.e., `[YY]YY, MM, DD, HH, CC, FF, TT`.

7.7.2 Using the `--cycle` option with emotion

Remember that in the absence of a specified cycle time, `ems_prep` will figure out the most recent run from which data are available. Yes, it's magic.

Here is a simple use of the `--cycle` option:

```
% ems_prep --dset nam212:ftp --cycle 12
```

Here, the `--cycle` option is used to request initialization data from the most recent 12 UTC run on the NAM on the 212 grid. It doesn't matter if the 18 UTC run is available, `ems_prep` doesn't care because you requested the 12 UTC data and that's exactly what you are going to receive gosh darn it.

Some more `--cycle` fun:

```
% ems_prep --dset nam212:ftp --cycle 12:06:30:06
```

Similar to the above example, except that the cycle time (12 UTC), initialization forecast hour (06), final forecast hour (30), and boundary condition frequency (06 hourly) is specified with the `--cycle` option. The forecast length is 24 hours (30 - 06). If the current cycle time of the requested initialization data is 00 UTC, that run will be ignored in favor of the previous 12 UTC run.

This time, something really interesting:

```
% ems_prep --dset namtile:http --cycle CYCLE:06
```

Translation: Use the 12km NAM personal tiles available from SOO STRC data server beginning with the 06 hour forecast from the most recent run of the NAM. The “**CYCLE**” is a placeholder for the current cycle run and must be used if you are overriding the default initial forecast hour (**INITFH**) value in `namtile_gribinfo.conf` with the 6 hour forecast (06).

7.7.3 Witness the power of `--length`

Here is a simple use of the `--length` option:

```
% ems_prep --dset nam212 --length 48
```

Translation: Acquire and process 48 hours of forecast files on the 212 grid from the most recent NAM run. Override the default simulation length defined in the `nam212_gribinfo.conf` file. Easy enough.

Here is a more complicated use of the `--length` option:

```
% ems_prep --dset nam212 --cycle CYCLE:12:36 --length 72
```

Translation: Without the “`--length 72`” flag, `ems_prep` will acquire and process 24 hours of forecast files on the 212 grid from the most recent NAM run (**CYCLE**) beginning with the 12 hour and going through the 36 hour forecast. However, the “`--length 72`” usurps the `--cycle` option and thus `ems_prep` will actually acquire and process 72 hours of forecast files beginning with the 12 hour and going through the 84 hour forecast. Pretty sneaky!

7.7.4 An example using the `--date` option

Here is a simple use of the `--date` option:

```
% ems_prep --dset nam212 --cycle 12 --date 20050230
```

Similar to the above examples except `ems_prep` will acquire and process the NAM 212 grid files from the 12 UTC cycle run on 30 February 2005. If the files are not already located in the “`grib`” directory then `ems_prep` will scour the world or at least those sources in the `nam212_gribinfo.conf` file, looking for your data files. You might use this example for running case studies.

Note that if you fail to include the `--cycle` option when specifying the date, `ems_prep` will default to the last cycle time on that date, which in this case would be from the 18 UTC run.

7.7.5 Working with multiple initialization data sets

The `ems_prep` routine can easily handle different data sets for initial and boundary conditions. For example:

```
% ems_prep --dset nam212%gfs --length 24
```

Translation: Use the most recent NAM 00 hour forecast on the 40km 212 grid for your 00 hour (initialization) and the most recent GFS run on the 0.5 degree grid as the lateral boundary conditions out to 24 hours. The `ems_prep` routine will attempt to acquire the 00 hour forecast from the NAM data set and then the 03 to 24 hour forecast files from the current GFS run, provided the current available cycle times are the same for each data set.

Important: The “`--length`” flag must be included when using a different data set for initial and boundary conditions.

So, what happens if the current cycle times are not the same? For example, what if it is 15 UTC and the 12 UTC run of the NAM is available but the GFS current run is from 06 UTC.

What then, Mr `ems_preppy`?

The `ems_prep` routine is all-wise! If the cycle time of the initialization and boundary condition data sets do not match then `ems_prep` will adjust the accordingly. In the above situation, the NAM 12 UTC 00 hour forecast will be used for your 00 hour forecast and the 09 through 30 hour forecasts from the GFS will be used. Don't bother to do the math, it's much too complicated.

How about this mind blower, again assume its **15 UTC**:

```
% ems_prep --dset nam212%gfs --length 24 --cycle CYCLE:12
```

Translation: Use the most recent NAM 12 hour forecast for your 00 hour (initialization) and the 21 through 42 hour forecast from the 06 UTC GFS run.

That's not magic, that's Voodoo.

Finally, you may also request the initial and boundary condition data sets be obtained from different sources:

```
% ems_prep --dset namptile:http:soostrc%gfsptile:http:emsdata1 --cycle CYCLE:84 --length 48 --sfc ssthr
```

Translation: Use the 12km NAM personal tiles (namptile) available from SOOSTRC 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 (`gfsptile`) from the EMSDATA1 data server. Total run length will be 48 hours (`--length 48`). Also, get the 8.3km high resolution SST data set (`--sfc ssthr`). Note that `ems_prep` will download the appropriate GFS files from the most recent GFS run. Again, if the GFS run is 6 hours older than the NAM the forecast times will be adjusted accordingly.

7.7.6 “What do we want? NARR and NNRP data! When do we want it? Now!”

Ah yes, in an effort to do everything for you, the SOO STRC and WRF EMS have focused their devastatingly powerful resources to bring you a much easier way of running historical case studies using the North American Regional Reanalysis (NARR) and Global Reanalysis I and II (NNRP) data sets. Don't understand all the hullabaloo? Try using these data sets without the “**EMS Advantage**®” and then you'll be thanking your Uncle EMS.

The SOO STRC servers (STRC, EMSDATA1, and EMSDATA2) now have a permanent on-line archive of these data sets just for use with the WRF EMS. The NARR data set currently includes files from 1991 to 2006, which the global NNRP data set runs from 1948 to 2006 (Reanalysis I from 1948-1979 and Reanalysis II from 1980-2006). More dates will be available as disk capacity is increased I hope.

To use these data sets, simply request “`narrptile`” or “`nnrp`” with the `--dset` option:

```
% ems_prep --dset narrptile --date 19931214 --cycle 12 --length 24  
Or  
% ems_prep --dset nnrp --date 19630311 --cycle 12 --length 24
```

Caveat: If you are running a case that occurred more than 50 years from the current date you might 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/ems_utils.pm line ...
```

The error is caused by a “design flaw” in Perl, not in the EMS. To fix this problem you will have to edit the “`Time/Local.pm`” module in your local Perl libraries and **decrease** the “50” value to a suitable value such as 30.

Chapter 8: Meet the WRF EMS `ems_run.pl` routine

Chapter Contents:

- 8.1 A taste of the `ems_run.pl` life
- 8.2 Meet your `ems_run` configuration files
- 8.3 Managing output from the model
- 8.4 Running `ems_run` from the command line
- 8.5 The `ems_run` command line options

8.1 A taste of the `ems_run.pl` life

The `ems_run.pl` routine is responsible for making sure that the specified WRF model configuration is valid, creating the model initial and boundary conditions, and then running the simulation. It also manages and controls all the MPICH2 processes if you are running the model across multiple computers and CPUs.

The `ems_run.pl` routine handles many of the minor details that typically encumber users prior to running a model simulation. There are quite a few banal tasks that typically must be completed prior to running the model and the EMS understands that you have better things to do with your time. Making your life easier so you can slack off, that's the primary duty of `ems_run.pl`.

The `ems_run.pl` routine will run the WRF `real.exe` and the either the NMM or ARW core. The WRF “`real.exe`” program, besides searching for a better name, handles the vertical interpolation of the initialization data output from `ems_prep.pl` to the computational domain followed by the creation of the initial and lateral boundary conditions for your run. Following successful completion, the routine will set up and run the desired WRF core. While the model is running, output files are initially placed in the top level of the domain directory. Once the simulation has completed, all data files are moved to the “`wrfprd`” subdirectory, log files are placed in the logs directory, and any ancillary files deleted.

As with all the EMS run-time routines, the `ems_run.pl` file is located in the `wrfems/strc` directory. The user never runs `ems_run.pl` directly, but rather, executes the `ems_prep` command from within one of the domain directories. For the sake of clarity, `ems_run` will be used throughout this chapter to refer to the link that exists in each domain directory.

Note that you must run the `ems_prep` routine prior to running `ems_run`.

8.2 Meet your `ems_run` configuration files

There are a number of configurable parameters that are used by `ems_run` for generating the initial conditions and running the model, all of which are described in quasi-organized configuration files. When you create a new domain, either with the Domain Wizard or manually, copies of the default configuration files are placed in the `<domain>/conf/ems_run` directory.

Novice users should not feel intimidated by all the options presented in these files. The parameters in these files are well-documented, which should help ease a user's “What/How do I do this?” fueled anxiety. Any non-WRF spawned anxieties should be handled professionally but feel free to include the EMS as part of therapy. The EMS comes pre-configured to run most types of simulations so there are only a few changes that a user may initially consider making.

Here is a brief description of the various `ems_run` configuration files, in a very loose order of importance:

A) `run_wrfout.conf`

The `run_wrfout.conf` configuration file handles the frequency of output for the primary and any nested domains. It also defines the format of the output files and a few additional settings. For the most part, the user only needs to modify, if needed, the output frequency of the simulation. The default file format is netCDF and is designed for use with the `ems_post` routine (Chapter 9). You should only change the format from the default netCDF if you should have a very good reason for doing so.

B) `run_physics.conf`

The `run_physics.conf` configuration file handles physics configuration of the primary and any nested domains. Each configuration option includes a brief description, so even the most WRF-savvy users may learn something by reviewing this file. Note that with the exception of the choice in 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. Even if you attempt to set the value for a nested domain to something other than that of the parent domain, the EMS will automatically reset the nested domain to match that of the parent. So no funny stuff, OK?

The EMS comes preconfigured to run the Kain-Fritsch cumulus scheme for the primary domain and without cumulus parameterization (microphysics only) for the first nested domain. If you have set up any domain with a grid spacing of 6km or less then you may consider turning off cumulus parameterization (0). If you have any domains with a grid spacing of 10km or greater, then you should consider turning on cumulus parameterization for those domains. For domains with grid spacing in between 6 and 10km, read the notes in the files and good luck.

Suggestion: If you are running a nested configuration with the primary (Domain 1) grid spacing of less than 10km then do not use a cumulus scheme with any domain.

C) `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 EMS was installed, it attempted to determine the number of physical CPUs and cores per CPU on the local system. The total number of available processors was used to assign default values in the `run_ncpus.conf` file.

It is recommended that you check the configuration of `REAL_NODECPUS` and `WRFM_NODECPUS` in `run_ncpus.conf` prior to running `ems_run`.

D) `run_timestep.conf`

The `run_timestep.conf` configuration file defines how the large timestep for your simulation is determined. You can select from a number of methods or use a specific timestep value. The default method will calculate a time step, based on NMM and ARW recommendations, from the smallest grid spacing within a primary computational domain. This approach should be sufficient for most applications.

E) `run_levels.conf`

The `run_levels.conf` configuration file defines the number and distribution of native model

levels in the computational domain. The default number of levels is 45 for both the NMM and ARW core simulations, 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.

F) `run_nests.conf`

The `run_nests.conf` configuration file defines the feedback (one- or two-way) and smoothing applied when running a nested simulation. The default value is for 2-way nesting but may need to be changed for some NMM nested simulation runs.

G) `run_dynamics.conf`

The `run_dynamics.conf` configuration file defines the configuration of the dynamics used in the model. For the NMM core the options are limited to hydrostatic vs. non-hydrostatic while there are a few more choices for the ARW core. For the most part, the default settings are sufficient for nearly all applications.

H) `run_dfi.conf`

The `run_dfi.conf` configuration file is available for those users wishing to run the WRF digital filter initialization. DFI allows for the reduction in the model spin-up time during the early stages of integration due to a mass/momentum imbalance in the initial conditions. The DFI option and at this time it is experimental; however, it does show promise in initial testing.

Note that the use of DFI can increase the computational time of your model run significantly so use this option wisely. Also testing has been limited so there are few promises as to whether this option will work as advertised.

I) `run_vinterp.conf`

The `run_vinterp.conf` configuration file defines a few options when doing the vertical interpolation of the initialization data to the computational domain. Most users don't need to make any changes to this file unless you are really, really bored.

J) `run_restart.conf`

The `run_restart.conf` configuration file defines a few options for doing a restart of a model run. This may be of interest to some users doing sensitivity studies or exceptionally long simulations but everyone else should ignore this file.

K) `run_auxhist1.conf`

The `run_auxhist1.conf` configuration file turn on the output of additional surface fields from a WRF NMM and ARW core simulation. By default output to these files is turned off simply because most (but not all) of the fields are also contained in the larger **wrfout** files. The fields contained in the "**sfcout**" files include:

Surface Fields:

- | | |
|---------------------|--------|
| 1. Surface Pressure | Pa |
| 2. Surface Height | meters |
| 3. Land/Water Mask | 1/0 |
| 4. PBL Height | meters |

Shelter Level (2 and 10 meter) Fields:

5.	2 Meter Temperature	K
6.	2 Meter Specific Humidity (ARW only)	kg kg ⁻¹
7.	2 Meter Relative Humidity (NMM only)	%
8.	10 Meter U-Wind	m s ⁻¹
9.	10 Meter V-Wind	m s ⁻¹

Accumulated and Total Precipitation Fields:

10.	Accumulated Total Precipitation	mm
11.	Accumulated Grid Scale Precipitation	mm
12.	Accumulated Convective Precipitation	mm
13.	Accumulated Snowfall (Liquid Equiv.)	mm
14.	Snow Depth on Ground	meters
15.	Snow (Liquid Equiv.) on Ground	mm

Fabulous Fun-Filled Fields* (and alliteration too):

16.	Maximum 10 Meter Wind Speed	m s ⁻¹
17.	Maximum Updraft	m s ⁻¹
18.	Maximum Downdraft	m s ⁻¹
19.	Maximum 1000m Reflectivity	dbZ
20.	Instantaneous Updraft Helicity (ARW only)	m ² s ⁻²
21.	Maximum Updraft Helicity	m ² s ⁻²
22.	Maximum Column Integrated Grauple	kg m ⁻²

* “Maximum” fields are computed between output times

Note that in the official WRF V3.1 release this option is fully active and available for use with **ems_autorun**. See Chapter 9 for details on processing the **sfcout** data files.

8.3 Managing output from the model

The primary data files output from the model are moved to the “**wrfprd**” directory following completion of a run. These data files contain a predefined set of variables on native model levels only. Users wishing to view their data on isobaric or other surface must process the output with additional software, which is why the **ems_post** routine is provided.

The default format of the primary WRF output files is netCDF. The EMS package provides a number of utilities that may be used to interrogate the raw data if desired including `rdwrfnc`, `ncview`, and `ncdump`; however, processing these data with **ems_post** is the best way to go.

8.4 Running `ems_run` from the command line

For the purpose of testing real-time forecasting systems and making case study simulations, **ems_run** routine may be run directly from the command line:

```
% ems_run [--domains 2,...,N] [additional optional flags]
```

Where [additional optional flags] may include the host of available options described in section 8.5.

If you are making a single domain simulation then you do not need to include any option flags when starting **ems_run**. If you want to include any nested domains in your run then you must include the **--domains** option flag; otherwise, the run will consist of only the primary domain (Domain 1). More

information on the `--domains` option flag is available in section 8.5.

8.5 The `ems_run.pl` command line options

There are a number of optional flags and arguments that 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 suggested that you modify the default values in the appropriate files so you only have to pass a minimum number of options. You have been warned.

Probably the most useful option is `--help`, which should be obvious 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 here.

A) **OPTION:** `--domains`

WHAT I DO: Provide control over domains included in simulation

USAGE: % `ems_run --domains domain1[:FCST LENGTH],...,domainN[:FCST LENGTH]`

Where $N \leq \text{Max Domains}$

DESCRIPTION:

Passing the `--domains` option defines a list of (nested) domain(s) to include in your simulation. All the domain(s) must have been initialized with `ems_prep` prior to including them in your run. By default, `ems_run` will only execute the primary domain unless the `--domains` flag is passed.

So if you fail to include the `--domains` flag, the simulation will only consist of the primary (outermost) domain.

The `--domains` flag can also be used to refine the length of any nested simulations, which is done by including the length of time followed by the units; d (days), h (hours), m (minutes), or s (seconds). The domain number and the length are separated by a colon (:). 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 with unique Start and Length values are separated by a comma
4. In the absence of a length value, the simulation will default to the length of the PARENT domain simulation

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. If domain 4 is actually the child of domain 3, which is not listed, then domain 3 *will be included* with a start time and length of its parent domain.

Trust me, there are a lot of checks to make sure the start and stop times of the nested simulations are correct!

B) **OPTION:** **--nodes**

WHAT I DO: Specifies the nodes and processors to use when running the ARW or NMM core

USAGE: % **ems_run --nodes** <WRFM_NODECPUS setting>

DESCRIPTION:

Passing the **--nodes** option overrides the value of WRFM_NODECPUS in the `run_ncpus.conf` file, so it's best that you know what WRFM_NODECPUS does before using this option.

The arguments to **--nodes** is a list of machines and processors, separated by a comma, that define the parallel computing environment when running the WRF model. The arguments take the following format:

```
% ems_run --nodes machine1:np,machine2:np,...,machineN:np  
Or  
% ems_run --nodes np  
Or  
% ems_run --nodes local:np  
Or  
% ems_run --nodes local
```

Where `machine` is the hostname of the system and `np` is the number of processors to use on that node. Using a literal `local` refers to the machine running the WRF EMS.

Passing either **--nodes np** or **--nodes local:np** will result in WRF EMS running the model on the local system ONLY with the number of processors specified by `np`. Passing **--nodes local** will result in the model be run on the number of processors specified by the **OMP_NUM_THREADS** environment variable defined in your **EMS.cshrc** file as **NCPUS * CORES**, which is probably what you want anyway.

Failure to correctly specify the machine and number of CPUs will result in the model being run on the local host with the number of CPUs defined by **OMP_NUM_THREADS**. Here are a couple of examples:

```
% ems_run --nodes node1:8,node2:8,node3:8
```

Translation: Run the WRF model on 3 nodes (`node1,node2`, and `node3`) each with 2 physical CPUs and 4 cores on each CPU (8 total virtual processors on each node).

```
% ems_run --nodes local:2
```

Translation: Run the WRF real program on the local machine only with 1 CPU and 2 cores or 2 CPUs with 1 core each (2 processors total).

C) **OPTION:** `--[no]clean`

WHAT I DO: Specifies the amount of directory cleaning before run

USAGE: % **ems_run** `--[no]clean`

DESCRIPTION:

The `--clean` flag overrides the default `ems_run` penchant for cleaning up your run-time directory prior to starting a new simulation. You probably will never need this option but its there for taking. Not passing `--clean` is equivalent to passing `ems_clean --level 2` prior to running `ems_run`, which will remove any left-over `ems_run` and `ems_post` files from the run-time directory and remove old log files. Passing `--noclean` will retain the existing WRF output files that reside in the “`wrfprd`” directory. That is the only real difference.

D) **OPTION:** `--start` and `--sdate`

WHAT I DO: Specifies the simulation start time

USAGE: % **ems_run** `--start [YY]YYMMDD` and `--sdate [YY]YYMMDD`

DESCRIPTION:

Passing either the `--start` or `--sdate` option allows users to start their simulation at a time after the original default start date/time specified when running `ems_prep`. The date specified by `[YY]YYMMDD` must correspond to a initialization data file residing in the “`wpsprd`” directory otherwise `ems_run` will give you a mean look and then quit.

So let’s say that you ran `ems_prep` and requested a 24 hour simulation initialized at 00 UTC on 30 February 2006 with 3-hourly boundary conditions. After `ems_prep` successfully completed its work you were not happy and rather have a 21 hour simulation starting at 03 UTC. Do you need to run `ems_prep` again? Nope, you can pass the `--start` flag to get the results that you deserve!

```
% ems_run --start 2006023003
```

You can also include the `-length` flag if you so desire:

```
% ems_run --start 2006023003 --length 12h
```

This will result in a 12 hour simulation beginning at 03 UTC 30 February 2006. Go ahead and try it.

Using the `--start` or `--sdate` option with `--domain` may cause problems unless you requested a start time for your nested domain(s) after that of the primary domain when running `ems_prep`. This is because there needs to be an initialization file in the “`wpsprd`” directory for each sub domain that corresponds to the start time for that domain.

E) **OPTION:** **--length**

WHAT I DO: Specifies the simulation length for the primary domain

USAGE: % **ems_run --length** TIME<d|h|m|s>

DESCRIPTION:

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 of time 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 will be ignored.

Note that reducing the length of the primary simulation may have an undesired effect on nested domains. 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? Run-on sentences are tough.

F) **OPTION:** **--rundir**

WHAT I DO: Specifies the run-time domain to use

USAGE: % **ems_run --rundir** <simulation run time domain directory>

DESCRIPTION:

Pass the **--rundir** flag if you want to specify the domain you wish to run. The domain directory must exist or **ems_run** will terminate, and you don't want that to happen now do you?

Note that this option should not be passed by the user, but rather, is used by **ems_autorun** internally. So, just say no to **--rundir**.

G) **OPTION:** **--levels**

WHAT I DO: Specifies the number of vertical levels to use in the simulation

USAGE: % **ems_run --levels** <number of vertical levels to use in the simulation>

DESCRIPTION:

The **--levels** command line option serves to override the `LEVELS` parameter in the `run_levels.conf` file. The only difference is that **--levels** flag only accepts an integer number of levels and not the vertical distribution. All the domains included in the simulation will use the same number of vertical levels and distribution.

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 generate a set of well-spaced levels. As stated above, the default number of levels is 45 for both the NMM and ARW core simulations, 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.

Finally, just remember that increasing the number of levels will proportionally increase the amount of time required to run your simulation.

H) **OPTION:** **--interp**

WHAT I DO: Interpolate nested static fields from parent domain (ARW nesting only)

USAGE: % **ems_run --interp**

DESCRIPTION:

Interpolate nested domain static surface fields (terrain, land-sea mask, etc.) from the parent domain. Generally, if you are running a nest you do not want to do this as you do not get the benefit of higher resolution surface fields in the nested domain(s), but it is sometimes helpful when trying to determine the forcing for a phenomenon of interest (ARW Nested runs only).

I) **OPTION:** **--syncts**

WHAT I DO: Interpolate Turn ON/OFF the synchronization of the primary TS to output frequency of nests

USAGE: % **ems_run --syncts** [0, 1, <anything else>]

DESCRIPTION:

Passing the **--syncts** option overrides the value of SYNCTS in the `run_timestep.conf` configuration file.

Passing **--syncts <anything other than 0 or 1>** tells **ems_run** to calculate the primary domain time-step such that all child (nested) domains will have timestep that coincides with the output times of that domain as defined by the `HISTORY_INTERVAL` value.

Passing **--syncts 1** tells **ems_run** to calculate the primary domain time-step such that it will coincide with the expected output times of domain 1 as defined by the `HISTORY_INTERVAL` value.

Passing **--syncts 0** turns this option OFF.

WHY DO I NEED THIS?

There are times when a sub domain timestep does not correspond exactly with the requested output time, in which case the model will dump the data at the first timestep AFTER the anticipated time. This may not be desirable for some applications, especially when running a simulation out beyond 18 hours and creating GRIB files. If you miss your output time such that the date/time stamp includes seconds > 0, then grib 1 file generation will fail, and you don't want that to happen.

Note that currently, the calculated sync timestep must be within 80% of the non-sync timestep value, which is calculated from the grid spacing of the primary domain.

J) **OPTION:** **--noreal**

WHAT I DO: Do not run the REAL program prior to starting forecast model

USAGE: % **ems_run --noreal**

DESCRIPTION:

Pass the **--noreal** option if you do not want to run the WRF real program prior to executing a simulation. The WRF real program generates the initial and boundary condition files that are used in the run, so you would only pass **--noreal** if you already have **wrfbdy_** and **wrfinput_** files lying around in the run directory.

K) **OPTION:** **--nowrf**

WHAT I DO: Terminate **ems_run** prior to running the model. Just runs WRF real.exe

USAGE: % **ems_run --nowrf**

DESCRIPTION:

Pass the **--nowrf** option if you do not want to run the WRF model after the WRF real.exe program has successfully completed. This is primarily used for testing and debugging purposes.

L) **OPTION:** **--autopost**

WHAT I DO: Turns ON autopost option for requested domains

USAGE: % **ems_run --autopost 1,...N**

DESCRIPTION:

Passing the **--autopost** flag initiates synchronous post processing of the WRF model forecast files, i.e., while the model is still running. This option is typically started through the **ems_autorun** routine and configuration file; however, you can override those values by passing the **--autopost** option to **ems_run**.

The arguments to **--autopost** include the domains for which you want to turn auto post processing ON. Passing **--autopost** without options will return an error but you never know.

Be sure to carefully read and configure the **ems_autopost.conf** file. Lots of good stuff there.

M) **OPTION:** **--rstint**

WHAT I DO: Specifies the output frequency of restart files

USAGE: % **ems_run --rstint <frequency in minutes>**

DESCRIPTION:

The **--rstint** flag overrides the **RESTART_INTERVAL** setting in the **run_restart.conf** file. The argument passed with **--rstint** specifies the length of time, in minutes, from the start of any run, restart or not, to generate restart files.

You may restart your WRF simulation following a crash using the `--restart` option to `ems_run`; however, in order to make a restart, restart files must have been generated by the now defunct simulation. These files, named as “`wrfrst_d<domain>_YYYY-MM-DD_HH:MN:SS`”, should have been placed in the “`rstprd`” directory within your run-time domain.

Passing `--rstint <some minutes greater than the length of the forecast>` will result in no restart files being generated. Setting it to something like 60 (minutes) will result in restart files being generated every hour.

Default is set to 360 or every 6 hours.

Finally, the requested frequency of the restart files to be output must be an integer multiplier of the lateral boundary condition frequency; otherwise, you will get a nasty message.

N) **OPTION:** `--restart`

WHAT I DO: Initiates a restart run at specified time

USAGE: % `ems_run --restart YYYY-MM-DD_HH:00:00`

DESCRIPTION:

When you want to restart the run simply make whatever changes you need and then pass the `--restart <date string>` to `ems_run`. The `<date string>` correspond to the `YYYY-MM-DD_HH:MN:SS` portion of one of the restart files listed in the `rstprd` directory. For example:

```
% ems_run --restart 2009-06-18_11:00:00
```

You can also use the `--length` option if you wish to shorten the length of the run:

```
% ems_run --restart 2009-06-18_11:00:00 --length 12h
```

Wherein the total length of the simulation from the ORIGINAL start time will be 12 hour. If the time is longer than the original length of the simulation then the length will automatically be adjusted to that of the original simulation.

O) **OPTION:** `--dfiopt`

WHAT I DO: Specify which Digital Filter Initialization (DFI) option to user

USAGE: % `ems_run --dfiopt <0, 1, 2, or 3>`

DESCRIPTION:

The `--dfiopt` flag overrides the `DFI_OPT` setting in the `run_dfi.conf` file. The argument passed with `--dfiopt` specifies which DFI option to use. Currently, the options are:

- 0 No DFI will be used (Turn OFF DFI)
- 1 Digital Filter Launch (DFL)
- 2 Diabatic DFI (DDFI)
- 3 Twice DFI (TDFI)

Option 3 (twice DFI) is currently recommended.

P) **OPTION:** **--dfilter**

WHAT I DO: Specify which Digital Filter Initialization (DFI) option to user

USAGE: % `ems_run --dfilter <0, 1, 2, 3, 4, 5, 6, 7, or 8>`

DESCRIPTION:

The **--dfilter** flag overrides the **DFI_NFILTER** setting in the **run_dfi.conf** file. The argument passed with **--dfilter** specifies which DFI filter option to use. Currently, the options are:

- 0 Uniform
- 1 Lanczos
- 2 Hamming
- 3 Blackman
- 4 Kaiser
- 5 Potter
- 6 Dolph window
- 7 Dolph
- 8 Recursive high-order

If it seems as though everybody has a filter named after them, you would be correct. There are actually billions and billions of DFI filters but only 8 are listed here.

By the way, option 7 (Dolph) is currently recommended.

Chapter 9: Meet the WRF EMS `ems_post.pl` routine

Chapter Contents:

- 9.1 A taste of the `ems_post.pl` life**
- 9.2 How do I process thee? Let me count the ways**
- 9.3 Welcome to your `ems_post` configuration files**
- 9.4 The `ems_post` command line options**
 - 9.4.1 General `ems_post` options and flags
 - 9.4.2 Options for the creation of GRIB files
 - 9.4.3 Options for the creation of GEMPAK grid files
 - 9.4.4 Options for the creation of GrADS grid files
 - 9.4.5 Options for the creation of BUFR, GEMPAK, and BUFKIT sounding files
- 9.5 Managing the fields in your WRF EMS GRIB files**
 - 9.5.1 Controlling GRIB file contents for different domains
 - 9.5.2 Deciphering the `wrfpost` control file
- 9.6 A message regarding the NMM core to GRIB processing**

9.1 A taste of the `ems_post.pl` life

The primary purpose of the `ems_post.pl` routine is to process output data files generated from a WRF NMM or ARW core simulation. By default, these files are in netCDF format on native model levels, which may not meet the ever-demanding needs of all users. Running the **`wrf_post.pl`** routine allows users to further process these data into a variety of formats and export the files to other systems.

The **`ems_post.pl`** processing options include, but are not limited to, converting the WRF netCDF to GRIB 1 or 2 format, writing to GEMPAK and GrADS files, and creating BUFR, GEMPAK, and BUFKIT sounding files. Users have the option to exports file to remote systems via secure copy (SCP), file transfer protocol (FTP) secure file transfer protocol (SFTP), or a simple copy command (CP).

Following successful completion of a simulation, all output data files are moved to the “**`wrfprd`**” directory. Files processed with **`ems_post.pl`** are placed in the “**`emsprd`**” directory, log files are placed in the “**`logs`**” directory, and any ancillary files deleted.

As with all the EMS run-time routines, the **`ems_post.pl`** file is located in the `wrfems/strc` directory. The user never runs **`ems_post.pl`** directly, but rather, executes the **`ems_post`** command from within one of the domain directories. For the sake of clarity, **`ems_post`** will be used throughout this chapter to refer to the link that exists in each domain directory.

9.2 How do I process thee? Let me count the ways

What can **`ems_post`** do for you? Well, the **`ems_post`** routine can:

1. Processes WRF NMM or ARW core output files in netCDF or GRIB 1 & 2 format on model native coordinate levels
2. Be run concurrently with the model simulation (autopost)
3. Output fields on as many as 80 pressure levels from the surface to 2mb, or as few as none (Section 9.5)

4. Handle output data files with a 1-minute temporal frequency
5. Handle the processing of multiple domains from a nested simulation independently
6. Produce data files in a variety of formats including GRIB 1, GRIB 2, BUFR, GEMPAK grid files, GEMPAK sounding files, BUFKIT, and GrADS, netCDF
7. Be used to automatically generate GEMPAK and/or GrADS images for display on the web.
8. Process BUFR and BUFKIT files with up to 1-minute frequency
9. Export data files to remote systems via FTP, COPY, SCP, or RCP depending on the user's needs
10. Allow users to fine-tune the frequency of the files being processed and exported in each format.

So, what can your `ems_post` do?

9.3 Welcome to your `ems_post` configuration files

There are a number of configurable parameters that are used by `ems_post` for processing WRF output, all of which are described in semi-organized configuration files. When you create a new domain, either with the Domain Wizard or manually, copies of the default configuration files are placed in the `<domain>/conf/ems_post` directory.

The parameters in these files are reasonably well-documented, so users should be able to figure out what to do; however, there is always some confusion as the descriptions in these files are frequently bungled by the author. Note that most of the configurable parameters may be overridden by command-line options, which are described in section 9.4.

Note that if you are processing the WRF output files as part of `ems_autorun` (Chapter 10), then it is imperative that you go through and edit the individual configuration files prior to running `ems_autorun`. If you are running `ems_post` manually, then you have the command-line options available to you.

Here is a brief description of the various `ems_post` configuration files, in no particular order of importance, except for the first one.

A) `ems_post.conf`

The `ems_post.conf` configuration file governs what type of post processing is to be done, which is why it is named different from the traditional convention. It is in this file that the user requests processing into the various formats such as GEMPAK, GrADS, and BUFR, which will then control whether the other configuration files are read. For example, if you set `GEMPAK = Yes` (or pass the `--gempak` option), the `post_gempak.conf` file will be read. So, before you attempt to run the `ems_autorun` routine make sure you make any needed changes to this file.

B) `post_export.conf`

The `post_export.conf` is the only configuration file, other than `ems_post.conf`, that is always read by the `ems_post` routine. This file controls the exporting of raw WRF output and any additional processed files to other systems. The only command-line option related to this file is `--noexport`, which simply turns off the exporting of all data files.

C) `post_grib.conf`

The `post_grib.conf` configuration file controls the processing of WRF output fields on native model surfaces to isobaric coordinates and then writing these data to GRIB 1 and 2 files. This processing is 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 regridding of files to user-specified grids. Good luck with that.

The fields and levels contained in the GRIB files may be controlled by the user. More information on controlling the GRIB file processing is provided in section 9.5.

D) `post_gempak.conf`

The `post_gempak.conf` configuration file manages the processing of the newly-minted GRIBs into GEMPAK grid files. The **ems_post** can handle the processing of remapped GRIB files as well. Additional options allow for the creation of gif images from the model output.

E) `post_grads.conf`

The `post_grads.conf` configuration file manages the processing of the newly-minted GRIB files into GrADS format. Users can also process any remapped GRIB files. Additional options allow for the creation of gif images from the model output that may be viewed in a web browser, so be sure to fire up GrADS processing!

F) `post_bufr.conf`

The `post_bufr.conf` configuration file controls the creation of BUFR sounding files from WRF netCDF output files. Additional options allow for the creation of GEMPAK sounding and BUFKIT files. I think there are some other options as well but I'll leave that for the next edition of this Guide. After all, it must remain "Nearly Complete".

G) `post_regrid.conf`

The `post_regrid.conf` configuration file allows users to remap GRIB files to an alternate grid domain and navigation. These regridded GRIB files are also available for conversion to GEMPAK and GrADS format.

9.4 The `ems_post` command line options

There are a number of optional flags and arguments that can be passed to **ems_post**. These options serve to override existing settings in the configuration files. For the most part, these options are simple and straight forward, which requires little explanation, primarily because I'm growing weary of writing.

The basic usage of **ems_post** looks something like:

```
% ems_post [--domain #] [additional optional flags]
```

If you are processing WRF output from domain 1 (primary domain) then you do not need to pass the **--domain** option. If you want to process data from any nested domain then you must include the **--domain** flag. In the absence of any option flags **ems_post** will default to the information in the

configuration files for domain 1.

Note that the WRF output files must exist in the “**wrfprd**” directory prior to running `ems_post` or else all heck will break loose and you will get a intimidating error message and a note to take home to your mother.

Finally, as always, the most useful `ems_post` flag is “**--help**”, which should be obvious 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 here.

9.4.1 General `ems_post` options and flags

Option: **--domain**

What I Do: Specify which domain to process

Usage: % `ems_post --domain #` [other options]

Description:

Passing the **--domain** option specifies which domain output to processes. Data files from the requested domain must exist in the “**wrfprd**” directory. In the absence of the **--domain** flag `ems_post` will default to domain 1.

Option: **--auxfile**

What I Do: Requests the processing of WRF auxiliary output files

Usage: % `ems_post --auxfile string` [other options]

Description:

Passing **--auxfile string** tells `ems_post` that you want to processes the WRF auxiliary files output from the model. These files are activated when the user specifies an output frequency greater than 0 (default; OFF) in `conf/ems_run/run_auxhist1.conf` configuration *prior* to running a simulation.

The mandatory argument to **--auxfile** is a character string that is used to determine which auxiliary files to process. Any WRF output files matching the string will be processed in accordance to the parameter settings for that domain as defined in the configuration files unless other command line options are passed.

Option: **--sfcout**

What I Do: Request the processing of WRF auxiliary sfcout data files

Usage: % `ems_post --sfcout` [other options]

Description:

The **--sfcout** option is very similar to **--auxfile** in that passing **--sfcout** tells `ems_post` that you want to processes the WRF auxiliary **sfcout** data files from a simulation. It is different from **--auxfile** in that an optional matching string is not necessary as `ems_post` will default to the

“**sfcout**” files. These files are activated when the user specifies an output frequency greater than 0 (default; OFF) in `conf/ems_run/run_auxhist1.conf` configuration *prior* to running a simulation. Processing of **sfcout** files for domains greater than 1 (primary) requires the additional use of the `--domain` option.

Option: `--noexport`

What I Do: Requests the processing of WRF auxiliary output files

Usage: `% ems_post --noexport <string> [other options]`

Description:

Passing `--noexport <string>` tells `ems_post` that you want to turn off the exporting of files that are otherwise requested in the `post_export.conf` file. The string should be carefully specifies so as to not turn off the exporting of files that you otherwise want to processes.

9.4.2 Options for the creation of GRIB files

Option: `--nopost`

What I Do: Turn off processing of netCDF to GRIB format with `wrfpost`

Usage: `% ems_post --nopost [other options]`

Description:

Passing `--nopost` turns off the processing of WRF netCDF files into GRIB format with the `wrfpost` routine. This option is very similar to `--nogrib` flag and is primarily used when you are generating BUFR files and are too lazy to set **GRIB = Yes** in `ems_post.conf`.

Option: `--nogrib`

What I Do: Turn off processing of netCDF to GRIB format

Usage: `% ems_post --nogrib [other options]`

Description:

Passing `--nogrib` turns off the processing of WRF netCDF files into GRIB format.

Option: `--[no]grib2`

What I Do: Turn on [off] the conversion of GRIB 1 to GRIB 2 format files

Usage: `% ems_post --nogrib2 [other options]`

Description:

Passing `--grib2` turns **on** the conversion of GRIB 1 files output from the `wrfpost` routine to GRIB 2 format.

Passing **--nogrib2** turns **off** the conversion of GRIB 1 files output from the `wrfpost` routine to GRIB 2 format.

Option: **--grbcnvrt**

What I Do: Converts WRF-generated GRIB 1 (2) files for GRIB 2 (1) format

Usage: % **ems_post --grbcnvrt** [other options]

Description:

Passing **--grbcnvrt** will convert the WRF-generated GRIB files, i.e., those output directly from the model, between GRIB 1 & 2.

Option: **--[no]regrid** and **--gridnum**

What I Do: Manage the remapping of GRIB 1 files to a different navigation.

Usage: % **ems_post --noregrid**

Or

 % **ems_post --regrid --gridnum <grid number or navigation>**

Description:

Passing the **--[no]regrid** flag overrides the **REGRID** setting in `ems_post.conf` and **--gridnum <grid number>** overrides the **GRIDNUM** value in `post_regrid.conf`. Also the shipment of regridded files across state lines is controlled in the `post_export.conf` file and is not yet considered a felony.

So why bother? Well, you might want to interpolate your run to one of the operational grids for direct comparison to an operational run. You can then easily calculate difference fields.

Passing the **--noregrid** turns **off** the remapping of GRIB 1 files to alternate navigations as requested in the `ems_post.conf` file.

Passing the **--regrid** turns **on** the remapping of GRIB 1 files to alternate navigations as requested in the `ems_post.conf` file. The navigation will default to that specified in the `post_regrid.conf` file. See the `post_regrid.conf` file for more information.

Passing the **--regrid --gridnum <grid number or navigation>** turns **on** the remapping of your GRIB files and specifies the NCEP grid ID number used to designate one of a myriad of possible GRIB grid navigations. Think 211 (80km), 212 (40km), 104 (90km), 221 (32km), 218 (12km), or any other of the many possible choices, most of which are not listed here.

What you will get if you use the **REGRID** option is your model forecasts interpolated to a predefined **GRID** navigation. Let's say that you ran model over central Iowa and then pass **--gridnum 104**. Your final regridded GRIB file will contain a grid point every ~90km over the entire 104 grid domain (North America) but actual non-missing values would only be located over Iowa at a 90km grid spacing.

Selecting a grid number that contains a lot of grid points, such as the 218 or 221 grid, will greatly increase the amount time required for processing.

9.4.3 Options for the creation of GEMPAK grid files

Note: the WRF EMS includes a full installation of NAWIPS/GEMPAK display software for viewing your model output files; however, by default, this option is turned **off** in the `EMS.cshrc` file simply because many users already have NAWIPS installed on their system. If you wish to use NAWIPS for displaying your model data please see chapter 3.4 for further details.

If all you are doing is generating GEMPAK files from WRF EMS output then you **do not need to activate NAWIPS on your system**. The call to the conversion routines is managed internally by `ems_post`.

Option: `--[no]gempak`

What I Do: Override the generation of GEMPAK files as defined in `ems_post.conf`.

Usage: `% ems_post --[no]gempak [other options]`

Description:

Passing `--[no]gempak` overrides the generation of GEMPAK files as defined in the `ems_post.conf` configuration file. All processed gempak grid files are located in the “`emsprd/gempak`” directory.

Option: `--[no]rgempak`

What I Do: Override the generation of GEMPAK files from remapped GRIB files.

Usage: `% ems_post --[no]rgempak [other options]`

Description:

Passing `--[no]rgempak` overrides the generation of GEMPAK files from the remapped GRIB files as defined in the `post_gempak.conf` configuration file. All processed gempak grid files they are located in the “`emsprd/gempak`” directory.

Option: `--nogemsc`

What I Do: Turns on/off the execution of the GEMPAK data processing script

Usage: `% ems_post --nogemsc`

Description:

Passing `--nogemsc` overrides the running of the GEMPAK data processing script specified in the `post_gempak.conf` file. This option is only relevant if you are creating GEMPAK files and have the `POSTSCR` and/or `RPOSTSCR` parameters set in `post_gempak.conf`; otherwise, just ignore it.

9.4.4 Options for the creation of GrADS grid files

Note: the WRF EMS includes a full installation of GrADS display software for viewing your model output files.

Option: **--[no]grads**

What I Do: Overrides the generation of GrADS files as defined in **ems_post.conf**.

Usage: % **ems_post --[no]grads** [other options]

Description:

Passing **--[no]grads** overrides the generation of GrADS files as requested in the **ems_post.conf** configuration file. If you choose to create GrADS grid files they will be located in the “**emsprd/grads**” directory.

Option: **--[no]rgrads**

What I Do: Override the generation of GrADS files from remapped GRIB files.

Usage: % **ems_post --[no]rgrads** [other options]

Description:

Passing **--[no]rgrads** overrides the generation of GrADS files from remapped GRIB files as defined in the **post_grads.conf** file. Again, all GrADS files will be located in the “**emsprd/grads**” directory.

Option: **--nogradsc**

What I Do: Override the option to execute the GrADS data image processing script

Usage: % **ems_post --nogradsc**

Description:

Passing **--nogradsc** overrides the running of the GrADS data processing script identified in the **post_grads.conf** configuration file. This option is only relevant if you are creating GrADS files and have the POSTSCR and/or RPOSTSCR parameter(s) set in **post_grads.conf**; otherwise, just ignore it.

9.4.5 Options for the creation of BUFR, GEMPAK, and BUFKIT sounding files

Option: **--[no]bufr**

What I Do: Override the generation of BUFR files as defined in **ems_post.conf**.

Usage: % **ems_post --[no]bufr** [other options]

Description:

Passing **--[no]bufr** overrides the generation of BUFR files as defined in the **ems_post.conf** file. It will also turn off and BUFKIT processing that was planned.

Passing **--bufr** turns on the generation of BUFR files. If you choose to create BUFR files they will be located in the “**emsprd/bufr**” directory.

Option: **--[no]gemsnd**

What I Do: Override the generation of GEMPAK sounding files as defined in `post_bufr.conf`.

Usage: % `ems_post` **--[no]gemsnd** [other options]

Description:

Passing **--[no]gemsnd** overrides the generation of GEMPAK sounding files as defined in the `post_bufr.conf` file. Passing **--nogemsnd** will not affect the generation of BUFR files, which will still be created if requested by the user in the `ems_post.conf` configuration file or by passing the **--bufr** flag. Passing **--gemsnd** will turn on the generation of GEMPAK sounding and BUFR files however, as BUFR files are needed to produce data for GEMPAK. All GEMPAK sounding files will be located in the “`emsprd/gemsnd`” directory.

Option: **--[no]bufkit**

What I Do: Override the generation of BUFKIT files as defined in `post_bufr.conf`.

Usage: % `ems_post` **--[no]bufkit** [other options]

Description:

Passing **--[no]bufkit** overrides the generation of BUFKIT files as defined in the `post_bufr.conf` file. Passing **--nobufkit** will not affect the generation of BUFR files, which will still be created if specified by the user in the `ems_post.conf` configuration file or by passing the **--bufr** flag. Passing **--bufkit** will turn on the generation of BUFKIT and BUFR files however, as BUFR files are needed to produce data for BUFKIT. All BUFKIT files will be located in the “`emsprd/bufkit`” directory.

9.5 Managing the fields in your WRF EMS GRIB files

9.5.1 GRIB file contents for different domains

Users of the WRF EMS have the ability to modify the fields and levels contained in the GRIB 1 and 2 files generated when running `ems_post`. This data management is accomplished through the use of a control file that is read by the `wrfpost` routine during execution. It is possible to specify a different control file for each WRF domain, thus allowing users to have different fields and levels in the GRIB files of the parent than those of child domains files.

Users specify which control file to use for a given domain with the **GRBCNTRL** parameter in the `conf/ems_post/post_grib.conf` configuration file. This parameter defines the list of control files, separated by a comma (,) to be used by `wrfpost`. Each entry in the list corresponds to the control file that is used when processing each domain. For example:

```
GRBCNTRL= wrfpost_cntrl_do1.parm, wrfpost_cntrl_do2.parm, ..., wrfpost_cntrl_doN.parm
```

Wherein “`wrfpost_cntrl_do1.parm`” will be used for domain 1 (primary domain), “`wrfpost_cntrl_do2.parm`” will be used for domain 2 (1st nested domain), etc. The actual name you give the control file is not important, as long as that file can be found under the “static” directory. If `ems_post` fails to locate a control file it will automatically use the default `wrfpost_cntrl.parm` file located in `wrfems/data/tables/post`.

You do not have to specify a unique GRIB control file for each domain. In the absence of multiple files, the last file listed in **GRBCNTRL** will be used.

GRBCNTRL = wrfpost_cntrl.parm

In the above example if you were running 3 total domains, one primary and two nests, all three domains would use the same **wrfpost_cntrl.parm** file to define the contents of the GRIB files.

9.5.2 Deciphering the wrfpost control file

The initial look at a wrfpost control file can be very confusing. You will find many semi-cryptic field names followed by an incomprehensible string of 1s and 0s. Each 1/0 represents a level that is turned on/off for output to the GRIB file. The wrfpost routine is expecting to find a string of 80 1s and 0s organized into 16 groups of 5. Having fewer than the 80 levels represented, e.g., if you copied this file from the WRF EMS V2 and didn't add additional values, will cause the post processor to crash.

Multi-level fields

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 but they could also be model or isentropic levels.

The DEFAULT pattern of 1s and 0s, as depicted above, outputs 3D fields every 25mb between the surface and 25mb, similar to EMS V2. Additional pressure levels have been added in V3 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 (mbs) in groups of 5, as represented by the 1s and 0s from left to right is:

Pressure Level (mb)					Corresponding 1/0 set
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

Chapter 10: Meet the WRF EMS `ems_autorun.pl` routine

Chapter Contents:

- 10.1 Living the `ems_autorun.pl` life**
- 10.2 Meet your `ems_autorun` master**
- 10.3 Setting up the EMS for real-time forecasting applications**
- 10.4 Running `ems_autorun` from the command line**
- 10.5 The `ems_autorun` command line options**
- 10.6 Concurrent post processing with EMS autopost**

10.1 Living the `ems_autorun.pl` life

The `ems_autorun.pl` routine is intended for use in automating the process of running a simulation and processing the output files. It is designed to read a user-controlled configuration file and then execute `ems_prep.pl`, `ems_run.pl`, and `ems_post.pl` in succession. The routine is ideally suited for use in real-time forecast applications; however, it may be used to run case studies as well. For real-time forecasting, there are various options to improve the reliability of your forecasts. Additionally, there is an option for processing the output data files concurrent with model run.

As with all the EMS run-time routines, the `ems_autorun.pl` file is located in the `wrfems/strc` directory. The user never runs `ems_autorun.pl` directly however, but rather, runs the routine from within one of the domain directories by using the “`ems_autorun`” symbolic link. For the sake of clarity, “`ems_autorun`” will be used throughout this chapter to refer to the link that exists in each domain directory.

This may be one of the few times in life you have the power; feel free to abuse it.

10.2 Meet your `ems_autorun` master

There are a number of controllable parameters that are used to drive the `ems_autorun` routine, all of which are described ad nauseam in the `ems_autorun.conf` configuration file. When you create a new domain, either with the Domain Wizard or manually, a copy of the default configuration is placed in the `<domain>/conf/ems_autorun` directory. The configurable parameters in this file are exceptionally well-documented, so prior to your initial `ems_autorun` attempt; you should take the time and review the contents of this file. You will be glad you did.

Also, remember that although there is only one configuration file used by `ems_autorun`, you will likely need to edit some of the other run-time configuration files for `ems_run`, `ems_post`, and if desired, `ems_autopost` (section 10.6).

10.3 Setting up the EMS for real-time forecasting applications

As stated above, the `ems_autorun` routine is ideally suited for use in real-time forecasting, allowing users to run the system on a regular schedule without any regular intervention. To facilitate this process an entry was placed in the user’s crontab file when the WRF EMS was installed. Using the “`crontab -l`” command you should see:

```
#8 2 *** /usr1/wrfems/strc/ems_bin/ems_autorun-wrapper.csh --rundir /usr1/wrfems/runs/<your domain> >& /usr1/wrfems/logs/ems_autorun.log 2>&1
```

Important: The above tiny example should serve as the template for all real-time forecasting applications.

Because it is important that the EMS environment variables are correctly set prior to running `ems_autorun`, `ems_autorun-wrapper.csh`, and not `ems_autorun.pl`, is actually run from the cron. The `ems_autorun-wrapper.csh` file is a C shell script that serves as a wrapper around the `ems_autorun.pl` routine. The script simply sets the EMS environment variables prior to running `ems_autorun.pl` and any arguments to `ems_autorun-wrapper.csh` are passed along.

In order to initiate a regular real-time forecasting system, a user must:

- a. Identify a domain for use in real-time forecasting
- b. Configure the various `ems_run` configuration files
- c. Configure the various `ems_post` configuration files
- d. Configure the `ems_autopost.conf` file if using the concurrent post-processing option
- e. Configure the `ems_autorun.conf` file
- f. Edit the crontab file (“`crontab -e`”):
 - Remove the comment (“`#`”) from the crontab entry
 - Specify the time(s) to initiate the forecast process (“`8 2 * * *`”)
 - Specify the computational domain to use (<your domain>)
- g. Follow these wise suggestions:
 - i. *Get the start time for your forecasts correct*

Once you have selected the data set(s) for initializing your real-time runs, be sure to note when these data are typically available on the remote server(s) as defined by the DELAY parameter in the `<data set>_gribinfo.conf` file located in the `wrfems/conf/grib_info` directory. The DELAY setting defines the number of hours after the official “cycle time” of a data set that the files are available on the remote server.

The reason for this suggestion is that the EMS knows what data sets should be available at any given time throughout the day. If you are using the most current GFS operational run as your initialization data, which has a DELAY value of 3 (hours), then the EMS knows that files from the 00 UTC run will not be available until after 15 UTC. If you start your real-time run before 03 UTC, even at 2:50 UTC, the EMS will attempt to download files from the previous GFS run (18 UTC previous day), and that is probably not what you want.

You may also want to make sure that the system time on your computer is correct as this can be the source of some problems, especially when the time zone information is incorrect.

- ii. *Before running from cron, test, test, test*

Prior to running everything from cron, test each step of the forecast process by manually running the run-time routines. This step will serve to check if there are any obvious mistakes or problems in your configuration. It will also keep you from much disappointment when your forecasts are not available as anticipated.

Start by running the `ems_prep` routine, including the “`--dset`”, “`--length`” and “`--domains`” options if necessary, followed by `ems_run` and `ems_post`. Make sure

everything ran as you expected. Pay close attention to amount of time required to run `ems_run` and make sure it fits within your operational window.

Once you are happy with the results of running each run-time routine individually, try executing `ems_autorun` manually from the command line. Again, pay close attention to the total amount of time required to run everything. If you are using the EMS autopost option, this a good opportunity to make sure that routine is working as expected.

Only after everything is operating satisfactorily should you activate your runs in the crontab file.

10.4 Running `ems_autorun` from the command line

For the purpose of testing real-time forecasting systems and making case study simulations, `ems_autorun` routine may be run directly from the command line:

```
% ems_autorun [options]
```

Where [options] may include the host of available options described in section 10.5.

For running case study simulations, it is important to include the `-date` and `-cycle` options as the `ems_autorun` routine will default to the date and cycle time of the most current data set being used to initialize your simulation.

10.5 The `ems_autorun` command line options

There are a number of optional flags and arguments that can be passed to `ems_autorun`. These options 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 run-time routines for processing. It works just like magic; it's much more fun if you don't ask too many questions.

Probably the most useful option is "`-help`", which probably obvious 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 here.

First, here is an index listing of the options described in this chapter:

- 10.5.1** Options: `--autopost` and `--post`
- 10.5.2** Option: `--clean`
- 10.5.3** Option: `--cycle`
- 10.5.4** Option: `--besthr`
- 10.5.5** Option: `--date`
- 10.5.6** Option: `--domains`
- 10.5.7** Options: `--dset`, `--sfc`, and `--lsm`
- 10.5.8** Option: `--length`
- 10.5.9** Option: `--nolock`
- 10.5.10** Option: `--noprep`
- 10.5.11** Option: `--rundir`
- 10.5.12** Option: `--verbose`

10.5.1 Options: `--autopost` and `--post`

What I Do: Manage the processing of output files from the simulation

Usage:

`% ems_autorun --post [All, 1, 2, ..., N]`

And

`% ems_autorun --autopost [All, 1, 2, ..., N]`

Description:

The `--post` and `--autopost` options override the **EMSPPOST** and **AUTOPOST** configuration file parameters that are used to define which domains you want processed and whether they should be processed concurrently or after the model has completed.

Use “`--autopost`” to specify which domains to process while the model is running. *Be sure to read and configure the `ems_autopost.conf` file before attempting this option!!*

Use “`--post`” to specify which domains to process following successful completion of the model.

Note passing ANY arguments to `--post` and `--autopost` blank will turn OFF post processing of the model output files, which means that nothing will be done to the forecast files following successful of the simulation. No BUFR files, no GRIB 1 files, no GEMPAK files, and no forecast files shipped anywhere. Have it your way.

Set `--post` and/or `--autopost` to “All” if you want to process all the domains included in the simulation.

Set `--post` and/or `--autopost` to the domain number (1 ... N) that you wish to process. Multiple domains can be included and are separated by a comma (,).

Note that the primary domain is designated as “1”.

Examples:

`--post 1,3` Post process domains 1 (Primary) and 3. Any other domains included in the simulation will not be processed.

`--autopost 2` Turn the autopost option ON and process domain 2.

`--autopost` Turn autopost OFF

`--autopost 1,2,5 --post 1,3` Turn on the autopost for domains 1, 2, and 5 as well as post process domains 1 and 3 *following* completion of the simulation.

10.5.2 Option: `--clean`

What I Do: Keep my working directory tidy

Usage:

`% ems_autorun --clean <0, 3, 4>`

Description:

The “`--clean`” flag overrides the `CLEAN` parameter in the `ems_autorun.conf` file that defines the amount of clearing/scouring done prior to running a simulation. The various integer values are the same as those used by the `ems_clean` routine so you should consult “`ems_clean --help`” for more information as to what is done at each level.

Most `ems_autorun` users should use a value of 4 or 3. Using `--clean 4` will remove any existing initialization files from the `<domain>/grib` directory, while using `--clean 3` will preserve the files. For running case studies it is recommended that you use `--clean 3` unless you want your initialization data deleted. If you are running a real-time system then set the `CLEAN` parameter to “4” in the configuration file. Passing “`--clean 0`” will turn OFF all scouring.

If the above was not clear enough:

- 0 Turn off all scouring because you like your directories dirty
- 3 Remove all non-essential files but retain initialization files in “grib”
- 4 Remove all non-essential files including initialization files in “grib”

10.5.3 Option: `--cycle`

What I Do: Specify the cycle time, start hour, end hour, and frequency of the initialization data used in your simulation.

Usage: % `ems_autorun` [other stuff] `--cycle CYCLE[:INITFH[:FINLFH[:FREQFH]]]`

Description:

Note: This option should NOT be used for real-time modeling purposes

The `--cycle` option defines the cycle time of the model data set to use for initialization of your simulation. The general usage is:

% `ems_autorun --dset <dataset> --cycle CYCLE`

Not passing the `--cycle` option will cause `ems_autorun` to use the cycle time of the most recent model run from which data are available. To determine the cycle time of the most recent run, `ems_autorun` accounts for the amount of time required to run the operational model and process the forecast files for distribution.

For example, if it takes NCEP three hours to run the model and process grib files then `ems_autorun` will not attempt to obtain data from the 12Z run until after 15Z.

The list of available cycles and the delay time for each data set are specified in the `<dataset>_gribinfo.conf` file by the `CYCLES` and `DELAY` parameters respectively.

Users Note: If you are making real-time model runs then you should **not** use `--cycle`.

The `--cycle` option also accepts 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 `<data set>_gribinfo.conf` file as `INITFH`, `FINLFH`, and `FREQFH` respectively.

The format for the argument list is:

% `ems_autorun--dset <dataset> --cycle CYCLE[:INITFH:FINLFH:FREQFH]`

Notice that the values for **INITFH**, **FINLFH**, and **FREQFH** are separated by colons (:). Here are a few examples. Feel free to make some of your own.

```
% ems_autorun--dset <dataset> --cycle 00:00:24:03
```

Translation: Use the 00Z 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. The script will attempt to download a 00, 03,06,09,12,15,18,21, and 24 hour forecast files. All default values are overridden.

```
% ems_autorun--dset <dataset> --cycle 06:06:30
```

Translation: Use the 06Z cycle time, the 06 hour forecast for the initialization time, the 30 hour forecast for the final boundary condition time (thus a 24 hour forecast), and use the default value in <data set> `_gribinfo.conf` (**FREQFH**) for the BC frequency.

```
% ems_autorun--dset <dataset> --cycle CYCLE:INITFH:36:12
```

Translation: Use the default cycle time (current), the default forecast hour for the initialization time, the 36 forecast for the final boundary condition time, and use 12-hourly BC files.

Finally (phew!), you actually don't need to include the **CYCLE**, **INITFH**, **FINLFH**, or **FREQFH** placeholders. You can just use the colons (:) without anything in between. For example:

```
% ems_autorun--dset <dataset> --cycle :::12
```

Or

```
% ems_autorun--dset <dataset> --cycle :24::06
```

Translations: Use the Use the default cycle time (current), the default forecast hour for the initialization time, the default forecast hour for the final boundary condition time, and use 12-hourly BC files.

In the second example, Use the Use the default cycle time (current), the 24 hour forecast for the initialization time, the default forecast hour for final boundary condition time, and use 6-hourly BC files.

You have the power, abuse it wisely.

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

Important: For the final time, *this option should not be used for real-time modeling purposes.*

10.5.4 Option: **--besthr**

What I Do: Synchronize the surface data sets to the model initialization time

Usage: % **ems_autorun --besthr** <sfc data set>,<sfc data set>,...

Description:

Note: For use with `--sfc` data sets only!

Passing `--besthr` tells the EMS to only use the cycle time of the surface data set that is closest to the initialization cycle. Consequently, it will not use the other available cycles when looking for data, but rather, will look for the same cycle hour over the period of days specified by the `AGED` parameter defined in the `<DATASET>_gribinfo.conf` file.

The argument to `--besthr` is the list of static surface data sets, separated by a comma, to which to apply this restriction and overrides the `BESTHR` field, if any, in the corresponding `<DATASET>_gribinfo.conf` file.

Why would you use this option?

Some data sets, such as MODIS SSTs have a diurnal variation that needs to be taken into account when initializing a simulation. It may not be appropriate to use a data set from 00Z for a simulation at 12Z even if that is the most recent time available. If `--besthr` is passed then the EMS will look for 12Z data from the previous day. Note that the available cycle times of the data set do not have to match the simulation initialization time. The WRF EMS will determine the closest cycle hour to the initialization hour and use that value.

10.5.5 Option: `--date`

What I Do: Specify the date for the model initialization

Usage: % `ems_autorun` [other stuff] `--date` [YY]YYMMDD

Description:

Passing the `"--date"` option defines the initialization date of the simulation for the primary domain (domain 1). This date will also be used for any sub (nested) domain unless it is overridden with the `"--domains"` option.

The argument to `--date` is a 4- or 2-digit year, 2-digit month (01 to 12), and 2-digit day (01 to 36). Not passing the `"--date"` option will cause `ems_autorun` to use the current system date.

Important: This option should **not** be used for real-time modeling purposes.

10.5.6 Option: `--domains`

What I Do: Provide control over domains included in simulation

Usage: % `ems_autorun` `--domains` <domain #:start hr:length>,...,<domain #N:start hr:length>

Where:

<domain #> Domain number 2..N (easy enough). Why not 1..N? That is because domain 1 is the primary domain is always executed and the length is controlled by influences outside the control of this file. That's why!

<start> The number of hours following the start of the domain 1, to begin the sub-domain integration. Not including this value will cause the EMS to begin the nested run at the same time as the PARENT domain.

<length> Length of the sub-domain simulation (hours). If the length of the forecast extends beyond that of the parent the run will be terminated with the parent simulation.

Description:

Passing the "**--domains**" option defines a list of (nested) domain(s) to initialize when running a simulation. Any domain(s) must have been defined and localized previously while running the GUI. If you created any sub domains (multiple layers of nests), then passing the **--domains** option will activate them. You will not be able to run a nested simulation unless you activate the sub domains!

NOTE: Domain 1 is the Mother or Parent of all domains and is always included by default.

If three nested domains were created and wish to activate all three:

```
% ems_autorun --dset dataset --domains 2,3,4 (NO spaces between domains)
Or
% ems_autorun --dset dataset --domains 4
```

Note that by requesting domain 4, domains 2 and 3 will automatically be activated!

So, why specify multiple domains (2, 3, and 4) when only one (4) will accomplish the same task? Because you can also control the start and stop times of the individual domains by including a "**:START**" and "**LENGTH**" in the list, where **START** is the number of hours *after* the start of the simulation (Domain 1) and **LENGTH** is the simulation length of the nested domain.

Here is a hypothetical configuration to demonstrate the power of the **--domains** option. Let's assume that you have created 10 domains (1 Parent of all domains and 9 sub domains with the following configuration:

```

                DOMAIN 1
-----
Domain 2   Domain 3   Domain 6
Domain 4   Domain 9   Domain 7
Domain 5   Domain 10  Domain 8
```

In the above configuration, Domain 1 is the parent of domains 2, 3, and 6. Domain 2 is the parent of domain 4, domain 9 the parent of 10, etc.

So, of you were to pass "**--domains 9**" to `ems_autorun`, then domains (1), 2,3,4,5,6,7,8 would also be included in the initialization.

Where:

<domain #> Domain number 2..N (easy enough). Why not 1..N? That is because domain 1 is the primary domain is always executed and the length is controlled by influences outside the control of this file. That's why!

<start> The number of hours following the start of the domain 1, to begin the sub-domain integration. Not including this value will cause the EMS to begin the nested run at the same time as the PARENT domain.

<length> Length of the sub-domain simulation (hours). If the length of the forecast extends beyond that of the parent the run will be terminated with the parent simulation.

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 hours AFTER the start of the main simulation, you would then include the START HOUR in your argument list to **--domains**:

```
% ems_autorun --dset gfs --domains 2:9,6:12
```

Again, although not explicitly included in the list, domains 3, 4, and 5 are implicitly included and thus will also be initialized.

The above command will initialize domains 2, 4, and 5 to start 9 hours *after* domain 1. Domain 6 will begin 12 hours after domain 1; however, domain 3 will start *at the same time* as domain 1. This is because, unless otherwise specified, included sub domains will be initialized to start at the same time as the parent!

Here is a more complicated example:

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

Ok, lots of stuff going on here. First note that domain 10 is not implicitly or explicitly included in the list so it will not be run. Also:

- **Domain 2** is (explicitly) initialized to start 3 hours after the start of domain 1
- **Domain 4** is (implicitly) initialized to start 3 hours after the start of domain 1
- **Domain 5** is (explicitly) initialized to start 15 hours after the start of domain 1

- **Domain 3** is (explicitly) initialized to start 12 hours after the start of domain 1
- **Domain 9** is (explicitly) initialized to start 6 hours after the start of domain 1; however, since this start hour is BEFORE the parent domain, 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 0 hours after the start of domain 1
- **Domain 7** is (explicitly) initialized to start 6 hours after the start of domain 1
- **Domain 8** is (explicitly) initialized to start 24 hours after the start of domain 1; however, since the total length of the domain 1 simulation is 24 hours, *Domain 8 will be turned off!*

Here is an example: If you created 4 nested domains, identified in the projection.jpg image (Note the NMM Warning) as 02...05, and you want to turn ON domains 2,4,and 5 then specify **--domains 2,4,5**.

See the `<domain>/static/projection.jpg` for available domains.

Remember that if you turn a nested domain ON then you also should include the parent domain (except for domain 1). Thus, if domain 3 is a child of 2, and you want to include 3 you should also include 2. Conversely, if you run over domain 2 you don't have to include domain 3.

Some examples, because I know you want them:

- domain 2:3:12** Start domain 2 simulation 3 hours after the start of domain 1 and run for 12 hours.
- domain 2:3** Start domain 2 simulation 3 hours after the start of domain 1 and run through the end of the parent domain simulation.
- domain 2:3:12,3:6:12** Start domain 2 simulation 3 hours after the start of domain 1 and run for 12 hours. Start domain 36 hours after domain 1 and run for 12 hours (!!). **However:** Since the forecast length of domain of domain 3 would extend beyond that of its parent (domain 2) then run length will be reduced to 9 hours.

10.5.7 Options: **--dset**, **--sfc**, and **--lsm**

What I Do: Manage the initialization data sets

Usage:

```
% ems_autorun --dset dataset[:METHOD:SOURCE:LOCATION]%(DATASET[:METHOD:SOURCE:LOCATION])
```

Or

```
--sfc dataset [:METHOD:SOURCE:LOCATION],[DATASET[:METHOD:SOURCE:LOCATION]],...
```

Or

```
--lsm dataset [:METHOD:SOURCE:LOCATION],[DATASET[:METHOD:SOURCE:LOCATION]],...
```

Description:

In its simplest usage, the **--dset**, **--sfc**, and **--lsm** options specify the data set to use for initial and boundary conditions (**--dset**), static surface fields (**--sfc**), and land-surface fields (**--lsm**) respectively.

Note that "**--dset dataset**" is the only mandatory option and oxymoron in **ems_autorun**; well, there may be other oxymora. Use of the **--sfc** and **--lsm** options are, of course, optional.

The behavior of these **ems_autorun** options is inextricably tied to the parameters and settings found in the respective `<data set>_gribinfo.conf` files, in particular the **SERVERS** section, which defines the source(s) for the data files. Thus, to fully understand the functionality of these options, you should review the **SERVERS** section of a `<data set>_gribinfo.conf` file. Just pick one to read, they all say the same thing.

In its simplest usage:

```
% ems_autorun --dset dataset
```

Or for example,

```
% ems_autorun --dset gfs
```

This tells **ems_autorun** to use the `gfs` data set, as defined in the `gfs_gribinfo.conf` file, as initial and boundary conditions for your model run. All default settings will be used unless you override them through the use of arguments to **--dset** and/or other command-line options.

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:/SL.us008001/MT.gfs_CY.CC/RD.YYYYMMDD/PT.grid_DF.gr2/fh.o0FF_tl.press_gr.op5deg

SERVER-HTTP = STRC:/data/grib/YYYYMMDD/gfs/grib.tCCz/YMMDDCC.gfs.tCCz.pgrb2fFF       =
SERVER-HTTP
TOC:/SL.us008001/MT.gfs_CY.CC/RD.YYYYMMDD/PT.grid_DF.gr2/fh.o0FF_tl.press_gr.op5deg

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

The above entries in the `gfs_gribinfo.conf` file indicate two FTP sources for initialization files, NCEP and TGFTP, two HTTP sources, STRC and TOC, and two NFS sources, DATA1 and a local source. The server IDs, i.e., NCEP, TGFTP, STRC, TOC, and DATA1 correspond to predefined hostnames or IP addresses located in the `wrfems/data/conf/ems_prep/prep_global.conf` file. The NFS entry without a server ID specifies the location of the gfs files on the local system. So why have entries for remote servers when the data can be obtained locally? Well, this is for demonstration purposes only; closed course with professional driver.

So, if you were to run the "`ems_autorun --dset gfs`" command as in the example above, `ems_autorun` would use the information provided in the `SERVERS` section of `gfs_gribinfo.conf` file to acquire the initialization files. The `ems_autorun` routine would first use the FTP command to access data on the servers identified by NCEP and TGFTP. If unsuccessful, `ems_autorun` will attempt to access files via http and then finally use secure copy (SCP) to obtain data from the DATA1 server and copy (cp) from the local system.

Yes, `ems_autorun` will attempt every possible source identified in the `gfs_gribinfo.conf` file. *Working harder so you can slack off would be `ems_prep's` motto if it had one, which it doesn't.*

Optional Arguments

There are arguments that may be passed to the `--dset` option that serve to modify its default behavior:

```
% ems_autorun --dset dataset[:METHOD:SOURCE:LOCATION]
```

Where **METHOD**, **SOURCE**, and **LOCATION** specify the method of acquisition, the source of the files, and the location and naming convention of the files at the remote location respectively.

METHOD

METHOD is used to override the default behavior of using each method, ftp, http, and/or nfs specified in the **SERVERS** section to acquire data. **METHOD** can take the form of:

```
nfs    Only use the SERVER-NFS entries in the <data set>_gribinfo.conf file.
ftp    Only use the SERVER-FTP entries in the <data set>_gribinfo.conf file.
http   Only use the SERVER-HTTP entries in the <data set>_gribinfo.conf file.
```

```
nonfs Don't use the SERVER-NFS entries in the <data set>_gribinfo.conf file.
noftp Don't use the SERVER-FTP entries in the <data set>_gribinfo.conf file.
nohttp Don't use the SERVER-HTTP entries in the <data set>_gribinfo.conf file.
```

```
none  Don't use any of the methods listed in the SERVERS section. All files are assumed to
        reside and correctly named in <domain>/grib directory.
```

Here are a couple of examples:

```
% ems_autorun --dset gfs:ftp
```

Translation: Only use the SERVER-FTP entries in the `gfs_gribinfo.conf` file to acquire data. If `ems_autorun` fails to located data from NCEP and TGFTP it will not use the other methods listed.

```
% ems_autorun --dset gfs:noftp
```

Translation: Do not use the SERVER-FTP entries in the `gfs_gribinfo.conf` file to acquire data. The `ems_autorun` routine will use the other methods listed (NFS and HTTP) to acquire data files.

```
% ems_autorun --dset gfs:none
```

Translation: Do not attempt to acquire the files as they already reside and are correctly named in the `<domain>/grib` directory. Note that the same behavior may be achieved by commenting out or deleting all the SERVER- entries in the `gfs_gribinfo.conf` file OR by passing:

```
% ems_autorun --dset gfs --nomethod
```

SOURCE

SOURCE 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 an may be associated with multiple methods. For example:

```
% ems_autorun --dset gfs:http:strc
```

This tells `ems_autorun` to only acquire 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 as specified in the `gfs_gribinfo.conf` file.

Use of a **METHOD** is optional, when excluding **METHOD**, `ems_autorun` will use all the methods listed that are associated with a given source:

```
% ems_autorun --dset gfs::strc
```

To acquire files locally that do not have a **SOURCE** associated with them in the `<data set>_gribinfo.conf`, such as in the last SERVER-NFS entry above, use "local":

```
% ems_autorun --dset gfs:nfs:local
```

Using METHOD, SOURCE, and LOCATION together, just like one big, happy family, only different

You may also use **SOURCE** to specify a hostname or IP address. This is best done in conjunction with both the **METHOD** and **LOCATION** argument. By using all three arguments you can request that initialization files be acquired from a location not listed in `<data set>_gribinfo.conf`. The format looks similar to the SERVER- entries:

```
% ems_autorun --dset gfs:http:nomad6:/pub/gfs/gfsYYYYMMDD/gfs.tCCz.pgrbfff
```

```
Or
% ems_autorun --dset gfs:http:nomads6.ncdc.noaa.gov:/pub/gfs/gfsYYYYMMDD/gfs.tCCz.pgrbfff
Or
% ems_autorun --dset gfs:http:205.167.25.170:/pub/gfs/gfsYYYYMMDD/gfs.tCCz.pgrbfff
```

Notice that all the examples above are equivalent, provided that there is a NOMAD6 entry as a server ID in the `prep_global.conf` file. In the above example you must specify a **METHOD**; otherwise something will fail.

Using Multiple Data Sets

The **--dset** option can be used to specify separate data sets for initial and boundary conditions in your model runs. For example, if you wish to use the 12km NAM files as the initial and 0.5 degree GFS for your boundary conditions, simply separate the two data sets with a "%" in the dataset argument to **--dset**, i.e.,

```
% ems_autorun --dset nam218%gfs --length 24
```

In which case **ems_autorun** will attempt to acquire a single nam218 file to use as the initial conditions and the gfs will be used for the boundary conditions through 24 hours. Note that the **--length** option must be used when specifying multiple data sets.

All the optional arguments detailed ad nauseam above are available for use with multiple data sets as well. For example; knock yourself out with such favorites as:

```
% ems_autorun --dset nam218:http%gfs::strc --length 36
```

Translation: Only use the SERVER-HTTP entries in the `nam218_gribinfo.conf` file to acquire 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 is the classic:

```
% ems_autorun --dset nam218:http:strc:/pub/nam/nam.tCCz.pgrbfff%gfs:nfs:local:/data/gfs/YYYYMMDD/gfs.tCCz.pgrb2fff --length 180
```

Translation: Override the `nam218_gribinfo.conf` SERVER entries and obtain the nam218 file via http from the source identified by STRC in the `prep_global.conf` file and located in the `/pub/nam` directory with the naming convention of `nam.tCCz.pgrbfff`. Also, Override the `gfs_gribinfo.conf` SERVER entries and copy (cp) the gfs files through 180 hours that are located in the `/data/gfs/YYYYMMDD` directory.

10.5.8 Option: **--length**

What I Do: Specifies the simulation length for the primary domain

Usage: % **ems_autorun** [other stuff] **--length HOURS**

Description:

Passing the **--length** option overrides the **FINLFH** (See: **--cycle** option) value with a value that would result in the requested forecast length (hours). The **--length** option usurps the forecast length defined by the **--cycle** option, so passing:

```
% ems_autorun --dset <dataset> --cycle 00:06:30:03 --length 36
```

Is the same as passing:

```
% ems_autorun --dset <dataset> --cycle 00:06:42:03
```

The `--length` option overrides everything when defining the length of your forecast. The `--length` option is king. All hail "`--length`"!

Important: *The `--length` option must be used when specifying separate data sets for initial and boundary conditions with the `--dset` option.* That's all I have to say about length.

10.5.9 Option: `--nolock`

What I Do: Requests that a lock file not be used with a simulation

Usage: % **ems_autorun** [other stuff] `--nolock`

Description:

Passing the `--nolock` flag turns OFF the creation of the **ems_autorun** run-time lock file.

Normally the **ems_autorun** routine will create a lock file in the `/usr1/wrfems/logs` directory containing information about the simulation process ID (PID), start date/time, the domain directory, and hostname of the system running **ems_autorun**. Each time a **ems_autorun** simulation is initiated, the `/usr1/wrfems/logs` directory is checked for existing lock files. If a lock file is found containing information indicating there may be an existing run in that same directory, the PID is checked to see if the simulation is still running on your system. If the previous processes has not finished, **ems_autorun** will wait (WAIT) for a specified amount of time for the simulation to complete before terminating itself.

When the `--nolock` option is passed **ems_autorun** does NOT look for existing lock files before starting a simulation. It doesn't create any either. It just doesn't care.

See the WAIT parameter in **ems_autorun.conf** for more details.

10.5.10 Option: `--noprep`

What I Do: Requests that **ems_prep** not be run before a simulation

Usage: % **ems_autorun** [other stuff] `--noprep`

Description:

Use the `--noprep` option if you wish to not run the **ems_autorun** routine and go straight into **ems_run** without processing the initialization files. By passing `--noprep` you agree to already have the output from **ems_autorun** located in the "wpsprd" directory and just hanker'n for a simulation to be run.

10.5.11 Option: `--rundir`

What I Do: Specifies the run-time domain to use

Usage: % `ems_autorun` [other stuff] `--rundir` <simulation run time domain directory>

Description:

Pass the "`--rundir`" option if you want to define the location of the domain directory. This option is typically used by the `ems_autorun` routine and is not recommended for use by the user, although this suggestion will probably not stop most of you.

Not passing `--rundir` will set the default as the current working directory, which is probably what you want when running from the command line. So again, it is probably in your best interest to leave this option alone. The domain directory must exist or `ems_autorun` will terminate, and you don't want that to happen now do you?

Typically, you would only include this option in a crontab entry for real-time simulations, but hey, who am I to stop you from using it for other ill-conceived activities.

10.5.12 Option: `--verbose`

What I Do: Changes the verbosity of the `ems_autorun` routine

Usage: % `ems_autorun` [other stuff] `--verbose` [`0 1 2 3`]

Description:

Use the `--verbose` option if you wish to override the default behavior as defined in the `EMS.cshrc` and `ems_autorun.conf` files. Accepted values are:

- 0 (Almost) all messages turned OFF
- 1 Only print out Error messages
- 2 Only print out Warning and Error message
- 3+ Print out all messages

The `VERBOSE` value in `EMS.cshrc` can also be overridden by the `VERBOSE` parameter contained in each domain's `ems_autorun.conf` file or by passing the `--verbose` <level> flag to any of the WRF EMS run-time scripts.

10.6 Concurrent post processing with EMS autopost

The WRF EMS has the capability to process output from a model simulation while the model is running. This functionality is quite useful for real-time forecasting purposes when timeliness of the forecast is critical. The option is initiated by the `ems_autorun` routine when AUTOPOST is activated in the `ems_autorun.conf` configuration file or by passing the “**--autopost <domain list>**” flag to `ems_autorun`.

The auto post-processing option is not without its caveats however. ***Running the `ems_autopost` routine concurrent with a simulation on the same system will severely degrade the performance of the run.*** If you are thinking about using the auto post option, and I know you are, it is strongly recommend that you configure a second machine for this task. It is easy to do and the `ems_autopost` code will do all the dirty work, the amount of which has been greatly increased for EMS V3 to make sure everything works as expected.

All you need to do is complete the following steps:

1. Find a second 1 or 2 CPU Linux machine with at least 1Gb memory. You will need at least 1.5Gb of memory for processing BUFR files.
2. Make sure `sshd` (SSH daemon) is running on the post processing machine
3. Set up a user account on the post processing with the same information as the machine running EMS.
4. Create an NFS mount of the `$EMS` directory on simulation workstation on the post machine note that `$EMS` must be the same location and path on both systems
5. Configure the machines so that the user can **SSH from the simulation to the post processing workstation and from the post back to the simulation workstation** without a password. If you can execute:

```
%      simulation machine-> ssh date <post machine>
And
%      post machine-> ssh date <simulation machine>
```

Without a password then you should be OK.

6. Edit the `ems_autopost.conf` configuration file with the hostname of the post processing system and number of available CPUs.

The purpose for the 2-way SSH configuration is that each machine must communicate with the other when their respective tasks have been completed.

Appendix A: Adding or Modifying Initialization Data Sets in the WRF EMS

Appendix Contents:

A.1 Welcome to the `gribinfo.conf` files!

A.2 Anatomy of a `gribinfo.conf` file

- A.2.1 CATEGORY
- A.2.2 INFO
- A.2.3 VCOORD
- A.2.4 INITFH
- A.2.5 FINLFH
- A.2.6 FREQFH
- A.2.7 CYCLES
- A.2.8 DELAY
- A.2.9 TILES
- A.2.10 SERVER-METHOD
- A.2.11 LOCFIL
- A.2.12 VTABLE and LVTABLE
- A.2.13 METGRID
- A.2.14 AGED
- A.2.15 TIMEDEP

A.1 Welcome to the `gribinfo.conf` files!

Each time that you run `ems_prep` or `ems_autorun`, the user requests data set(s) for use in WRF model initialization and boundary conditions:

```
% ems_prep --dset nam212 [other options]
```

In the above example, the NAM 212 grid is being used for model initialization and boundary conditions; however, there are many other data sets from which to choose.

Information describing the NAM 212 grid files and all other data sets used for STRC WRF EMS initialization are contained in `gribinfo.conf` files, which are located in the `wrfems/conf/grib_info` directory. The `<data set>_gribinfo.conf` files define the default attributes for each data set available for model initialization. When accessing these files, `<data set>` 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 first 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 want to modify the information contained in one of these configuration files 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 complete 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 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.

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.

Examples:

INITFH = 00

To use the 00 hour forecast from the data set as your 00 hour forecast.

INITFH = 06

To use the 06 hour forecast from the data set as your 00 hour forecast.

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.

Note that **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 (,). For example:

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. See "**ems_guide --emsprep cycle**" for the gory details.

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**.

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 TILES

What I Do: Define the default tile number to use when requesting NCEP tiles

Description:

The **TILES** parameter contains a list of NCEP model grib tiles separated by a comma (.). If the data set described by this file does not use NCEP tiles then **TILES** will be ignored. If the data set does consist if individual tiles that must be quilted together then be sure to include the **TT** placeholder for tile number in the file naming convention for both the remote and local filenames.

Example: **TILES = 31,32,22,23**

Note that gif images depicting the 32km 221 and 12km 218 tile locations can be found in the wrfems/docs directory.

Important: It is up to the user to make sure that the list of tiles, when quilted together, creates a rectangle and that the quilted domain is sufficiently large to cover the model computational domain. Bad things happen when the model domain resides outside the areal coverage of the initialization data set. *Not like William Shatner sings "Lucy in the Sky with Diamonds" bad, but you still do not want to go there.*

A.2.10 SERVER-METHOD

What I Do: Lists local, http, and ftp the sources of the data set

Description:

Appendix A – Adding or Modifying Initialization Data Sets in the WRF EMS

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**:/directory location of data on server/filename convention

Important: The **SERVER ID** must have a corresponding IP/hostname defined in the **SERVER** section of the wrfems/data/conf/**ems_prep**/prep_global.conf configuration file.

Note that the following place holders 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
TT Tile number(s) for GRIB tiles.

The **METHOD** indicates the methods 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.pgrb2foFF.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/YMMDDCC.gfs.tCCz.pgrb2fFF

Appendix A – Adding or Modifying Initialization Data Sets in the WRF EMS

Where in `prep_global.conf`: KIELBASA = roz@kielbasa (Note the all capital letters for kielbasa)

And

```
SERVER-NFS = roz@kielbasa:/data/YYYYMMDD/gfs/YMMDDCC.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 you to specify a server when passing the `--dset <data set>:nfs:<server>` flag to **ems_prep**. So if you had:

```
SERVER-NFS =  
SERVER_A:/data/YYYYMMDD/grib/grib.tCCz/nam.tCCz.awip3dFF.tmoo.gz =  
SERVER-NFS =  
SERVER_B:/data/YYYYMMDD/grib/grib.tCCz/nam.tCCz.awip3dFF.tmoo.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 (either 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 if 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.tmoo.gz  
Or  
SERVER-NFS = /data/YYYYMMDD/grib/grib.tCCz/nam.tCCz.awip3dFF.tmoo.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>
```

Attempt to get <data set> via <method> from <server>

```
% ems_prep --dset <data set>:<method>
```

Attempt to get <data set> from all the <method> servers listed in the `gribinfo.conf` file.

```
% ems_prep --dset <data set>
```

Attempt to get <data set> via all the methods and servers listed in the <data set>_gribinfo.conf file.

A.2.11 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** place holders listed in the **SERVER-METHOD** section.

GRIB 2 <-> 1 CONVERSION

The WRF EMS 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.2.12 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 wrfems/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.2.13 METGRID

What I Do: Specifies an alternate METGRID.TBL table to use by **ems_prep**

Description:

The **METGRID** parameter is used to specify an alternate **METGRID.TBL** file when running the metgrid routine as part of **ems_prep**. If **METGRID** is not specified, then the default tables will be used for both the NMM and ARW cores. There are some data sets, such as the NNRP, require modifications to the default tables (one for the ARW and NMM core), and thus, a new **METGRID.TBL** file should be created based upon an existing table.

The key work "CORE" in the **METGRID** filename below will be replaced by the appropriate core. For example, when setting **METGRID** below to

METGRID = METGRID.TBL.NNRP.CORE

The **ems_prep** routine will set the name of the file to either **METGRID.TBL.NNRP.ARW** or **METGRID.TBL.NNRP.NMM** depending on the WRF model being run. Using the **METGRID** setting assumes that you have created appropriately named files and placed them in the wrfems/data/conf/tables/wps directory with the other metgrid tables.

A.2.14 AGED

What I Do: Define the number of days before a data set is too old for use in a simulation

Description:

The **AGED** parameter is the number of days prior to the model initialization date before the surface files are considered "old" and should not be used in your run. The **ems_prep** routine will attempt to get the most current data set available but will use surface data sets up to **AGED** days old if a more recent file is not available. If you requested that alternate static surface fields be used by passing the **--sfc <data sets>** option and nothing is available, then the fields from the primary initialization files will be used.

Note: This setting is *ONLY* used with **--sfc** option.

A.2.15 TIMEDEP

What I Do: Specify the data set requires time dependent LSM fields

Description:

The **TIMEDEP** parameter is set to Yes if the files are to be used as time dependent LSM fields. Some data sets, such as the NNRP2D, require **TIMEDEP = Yes**, but for most data sets this option is not necessary. Note that currently there is no way to override this value from the command line.

Appendix B: Running the WRF EMS Benchmark Cases

Appendix Contents:

- B.1 A bit of an introduction is in order**
- B.2 Benchmark case background information**
- B.3 How to run the benchmark case**
- B.4 What do I do with the benchmark results?**

B.1 A bit of an introduction is in order

The EMS package includes preconfigured ARW and NMM core domains for the purpose of testing the installation and evaluating the performance of the cores on your computer system. In addition, data are provided that allow users to benchmark their system and compare the results to other systems. Running these benchmarks is straight forward and should be a priority for a new user or anyone else who would like to taste the sweet nectar of success.

Each benchmark simulation comes preconfigured with default physics and dynamics settings for each core. If you are running these benchmarks for the purpose of comparing the performance of your system to others, then it is recommended that you not modify these settings; otherwise the comparison is useless. If you wish to compare the results and/or performance from the various physics and dynamics options then feel free to modify the configuration files in the respective conf/ems_run directory.

The default configuration for each benchmark can be viewed by using the “runinfo” command from a benchmark directory:

```
% runinfo [--domain 2]
```

Finally, prior to running a 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.

B.2 Benchmark case background information

The benchmark case is a 24-hour simulation of an extreme rainfall event over Minnesota, Iowa, and Wisconsin in the northern US that occurred 18-19 August 2007.

The areal coverage of the primary domain for both the NMM and ARW core benchmarks domains extends over the central US and consists of ~11250 horizontal grid points (NMM:75x150, ARW:106x106) with 15km grid spacing and 45 vertical levels. There is a second level nested domain that also consists of ~11250 horizontal grid points for each core with a 5km grid spacing (3:1 parent:child ratio) and 45 vertical levels. The nested domain is centered over Iowa and extends into all six neighboring states. The areal coverage for each domain is depicted in the projection.jpg image located in each static directory.

B.3 How to run the benchmark case

Running the benchmark case for each WRF core with the WRF EMS is straight forward:

Step I. From the benchmark/<core> directory, run the `ems_prep`

```
% ems_prep --benchmark
```

Or if you wish to include the nested domain:

```
% ems_prep --benchmark --domain 2
```

Step II. Run `ems_run` to begin the simulation

```
% ems_run
```

Or if you wish to include the nested domain:

```
% ems_run --domain 2
```

Following completion of the simulation the model output will be located in the `wrfprd` directory.

Step III. (Optional) Convert the output files and view the results

The 3-hourly netCDF forecast files will be located in the `wrfprd` directory along with 1-hourly files from the nested domain (if selected). You can convert the files to grib 1 format (and gempak) by running the `ems_post` routine.

```
% ems_post --grib (For grib 1 files only)
```

Or if you want to process the nested domain:

```
% ems_post --grib --domain 2
```

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 many other post processing options available to you. Please see Chapter 9 on the “`ems_post`” routine for the gory details.

B.4 What do I do with the benchmark results?

Upon completion of the run there will be a file in the `static` directory called `wrfems_benchmark.info`. Feel free to send this file to the SOO STRC for inclusion in the WRF EMS benchmarks site:

<http://strc.comet.ucar.edu/wrfems/benchmarks.htm>

Send your benchmark results to: Robert.Rozumalski@noaa.gov

Appendix C – A Description of the EMS wrfpost output fields

Appendix C: A Description of the EMS wrfpost output fields

Appendix Contents:

- C.1 A bit of an introduction is in order
- C.2 A list of fields available from the EMS wrfpost routine

C.1 A bit of an introduction is in order

As described in Chapter 9, the **wrfpost_ctrl_*.parm** files allow users to control the fields output by the wrfpost routine in GRIB 1 format. The wrfpost routine has been extensively modified from the NCEP wrfpost package so you cannot use the binaries created from that release with the EMS.

These control files are located beneath each domain directory in the “**static**” subdirectory. Note that you should not change the Process (Model), Center, or Subcenter IDs in the control files as the values are replaced with those specified by the user in **post_grib.conf**.

Below is a list of the fields available from the **ems_post**. Those in **bold face** are turned **ON** by default; however, some fields are only available when running a specific WRF core or physical parameterization scheme, and thus may not be included in the final GRIB file.

C.2 A list of fields available from the EMS wrfpost routine

Control File ID	Description	GRIB ID	Level
ABS VORT ON MDL SFCS	Absolute vorticity on model surface	041	109
ABS VORT ON P SFCS	Absolute vorticity on pressure surface	041	100
0-2000FT LLWS	0 to 2000 ft low level wind sheer		
ACC SFC EVAPORATION	Accumulated surface evaporation	057	001
ACM CONVCTIVE PRECIP	Accumulated convective precipitation	063	001
ACM SNOWFALL	Accumulated snowfall	065	001
ACM SNOW TOTAL MELT	Accumulated total snow melt	099	001
ACM TOTAL PRECIP	Accumulated total precipitation	061	001
AIR DRY SOIL MOIST	Air dry soil moisture	231	001
ASYMPT MSTR LEN SCL	Asymtropic length scale on model surface	227	109
AVE GROUND HEAT FX	Ground heat flux - time-averaged	155	001
AVE INCMG SFC LW RAD	Incoming sfc longwave radiation - time-averaged	205	001
AVE INCMG SFC SW RAD	Incoming sfc shrtwave radiation - time-ave	204	001
AVE OUTGO SFC LW RAD	Outgoing sfc longwave radiation - time-ave	212	001
AVE OUTGO SFC SW RAD	Outgoing sfc shrtwave radiation - time-ave	211	001
AVE OUTGO TOA LW RAD	Outgoing model top LW radiation –average	212	008
AVE OUTGO TOA SW RAD	Outgoing model top SW radiation -average	211	008
AVE SFC LATHEAT FX	Surface latent heat flux - time-averaged	121	001
AVE SFC MOMENTUM FX	Surface momentum flux - time-averaged	172	001
AVE SFC SENHEAT FX	Surface sensible heat flux - time-averaged	122	001
AVG CNVCT CLD FRAC	Time-averaged convective cloud fraction	072	200
AVG STRAT CLD FRAC	Time-averaged stratopheric cloud fraction	213	200
AVG TOTAL CLD FRAC	Time-averaged total cloud fraction	071	200
BOTTOM SOIL TEMP	Soil temperature at the bottom of soil layer	085	111
BRIGHTNESS TEMP	Clout Top Brightness Temperature		

Appendix C – A Description of the EMS wrfpost output fields

Control File ID	Description	GRIB ID	Level
CANOPY COND HUMID	Canopy conductance - humidity componen	248	001
CANOPY COND SOILM	Canopy conductance - soil component	249	001
CANOPY COND SOLAR	Canopy conductance - solar component	246	001
CANOPY COND TEMP	Canopy conductance - temperature compo	247	001
CANOPY CONDUCTANCE	Canopy conductance	181	001
CANOPY WATER EVAP	Canopy water evaporation	200	001
CEILING	Ceiling height		
CLEAR AIR TURBULENCE	Clean Air Turbulence		
CLD FRAC ON MDL SFCS	Cloud fraction on model surface	071	109
CLD ICE ON MDL SFCS	Cloud ice on model surface	058	109
CLD WTR ON MDL SFCS	Cloud water on model surface	153	109
CLOUD BOT PRESSURE	Cloud bottom pressure	001	002
CLOUD BOTTOM HEIGHT	Cloud bottom height	007	002
CLOUD ICE ON P SFCS	Cloud ice on pressure surface	058	100
CLOUD TOP HEIGHT	Cloud top height	007	003
CLOUD TOP PRESSURE	Cloud top pressure	001	003
CLOUD TOP TEMPS	Cloud top temperature	011	003
CLOUD WATR ON P SFCS	Cloud water on pressure surface	153	100
CNVCT AVBL POT ENRGY	CAPE	157	001
CNVCT INHIBITION	CIN	156	001
COMPOSITE RADAR REFL	Composite radar reflectivity	212	200
CONDENSATE ON P SFCS	Total condensate on pressure surface	135	100
CONV CLOUD BOT PRESS	Covective cloud bottom pressure	001	242
CONV CLOUD FRACTION	Convective cloud fraction	072	200
CONV CLOUD TOP PRESS	Covective cloud top pressure	001	243
CONV PRECIP RATE	Convective precipitation rate	214	001
CU CLOUD EFFICIENCY	Convective cloud efficiency	134	200
DEEP CU CLD BOT PRES	Deep covective cloud bottom pressure	001	251
DEEP CU CLD TOP PRES	Deep covective cloud top pressure	001	252
DIFFUSION H RATE S S	Heat diffusivity on sigma surface	182	107
DIRECT SOIL EVAP	Direct soil evaporation	199	001
DWPT IN BNDRY LYR	Dew point temperature in boundary layer (30 mb	017	116
DWPT TEMP ON MDL SFC	Dew point temperature on model surface	017	109
DWPT TEMP ON P SFCS	Dew point temperature on pressure surfac	017	100
DOMINANT PRECIP TYPE	Dominant precip type	141	100
FLIGHT RESTRICTION	Flight Restriction		
FRICTION VELOCITY	Friction velocity	253	001
GREEN VEG COVER	Vegetation cover	087	001
GRID CLOUD BOT PRESS	Grid scale cloud bottom pressure	001	206
GRID CLOUD TOP PRESS	Grid scale cloud top pressure	001	207
HEIGHT AT TROPOPAUSE	Height at tropopause	007	007
HEIGHT OF FRZ LVL	Freezing level height (above mean sea level	007	004
HEIGHT OF PRESS SFCS	Height on pressure surface	007	100
HEIGHT ON MDL SFCS	Height on model surface	007	109
HIGH CLOUD FRACTION	High level cloud fraction	075	234
HIGHEST FREEZE LVL	Highest freezing level height	007	204
IN-FLIGHT ICING	In Flight Icing		
INSTANT PRECIP RATE	Precipitation rate - instantaneous	059	001
INSTANT PRECIP TYPE	Precipitation type (4 types) - instantaneous	140	001
INST GROUND HEAT FLX	Ground heat flux - instantaneous	155	001

Appendix C – A Description of the EMS wrfpost output fields

Control File ID	Description	GRIB ID	Level
INSTN INC SFC LW RAD	Incoming sfc LW radiation - instantaneous	205	001
INSTN INC SFC SW RAD	Incoming sfc SW radiation - instantaneous	204	001
INSTN OUT SFC LW RAD	Outgoing sfc LW radiation - instantaneous	212	001
INSTN OUT SFC SW RAD	Outgoing sfc SW radiation - instantaneous	211	001
INST SFC LATHEAT FX	SFC latent heat flux - instantaneous	121	001
INST SFC SENHEAT FX	SFC sensible heat flux - instantaneous	122	001
LAND SEA MASK	Land sea mask (land=1, sea=0)	081	001
LATITUDE	Latitude	176	001
LCL AGL HEIGHT	Above-ground height of LCL	007	005
LCL PRESSURE	Pressure of LCL	001	005
LIFTED INDEX--BEST	Lifted index--best	132	116
LIFTED INDEX--BNDLYR	Lifted index--from boundary layer	024	116
LIFTED INDEX--SURFCE	Lifted index--surfce based	131	101
LIQUID SOIL MOISTURE	Liquid soil moisture	160	112
LONGITUDE	LONGITUDE	177	001
LOW CLOUD FRACTION	Low level cloud fraction	073	214
LW RAD TEMP TNDY	Temp tendency from longwave radiative fl	251	109
MASTER LENGTH SCALE	Master length scale on model surface	226	109
MAXIMUM SNOW ALBEDO	Maximum snow albedo	159	001
MESINGER MEAN SLP	Mesinger (Membrane) sea level pressure	130	102
MID CLOUD FRACTION	Mid level cloud fraction	074	224
MIN STOMATAL RESIST	Minimum stomatal resistance	203	001
MST CNVG ON MDL SFCS	Moisture convergence on model surface	135	109
MST CNVG ON P SFCS	Moisture convergence on pressure surface	135	100
MST CNV IN BNDRY LYR	Moisture convergence in boundary layer (30 mb n	135	116
NO OF ROOT LAYERS	Number of root layers	171	001
OMEGA IN BNDRY LYR	Omega in boundary layer (30 mb mean)	039	116
OMEGA ON MDL SFCS	Omega on model surface	039	109
OMEGA ON PRESS SFCS	Omega on pressure surface	039	100
PBL HEIGHT	PBL height	221	001
PERCENT SNOW COVER	Snow cover in percentage	238	001
PLANT TRANSPIRATION	Plant transpiration	210	001
POTENTIAL EVAP	Potential evaporation	145	001
POTENTL TEMP AT TROP	Potential temperature at tropopause	013	007
POT TEMP AT 10 M	10 M potential temperature	013	105
POT TEMP ON MDL SFCS	Potential temperature on model surface	013	109
POT TEMP ON P SFCS	Potential temperature on pressure surface	013	100
POT TMP IN BNDRY LYR	Potential temperature in boundary layers (30 mb	013	116
PRECIPITABLE WATER	Column integrated precipitable water	054	200
PRESS AT TROPOPAUSE	Press at tropopause	001	007
PRESS IN BNDRY LYR	Pressure in boundary layer (30 mb mean)	001	116
PRESS ON MDL SFCS	Pressure on model surface	001	109
P WATER IN BNDRY LYR	Precipitable water in boundary layer (30 mb mear	054	116
RADAR REFL AGL	Radar reflectivity at certain above ground heights	211	
RADAR REFL MDL SFCS	Radar reflectivity on model surace	211	109
RADAR REFL ON P SFCS	Radar reflectivity on pressure surace	211	100
RADFLX CNVG TMP TNDY	Temperature tendency from radiative flux	216	109
RAIN ON MDL SFCS	Rain on model surface	170	109
RAIN ON P SFCS	Rain on pressure surface	170	100
RCHDSN NO ON MDL SFC	Richardson number on model surface	254	109

Appendix C – A Description of the EMS wrfpost output fields

Control File ID	Description	GRIB ID	Level
REL HUMID AT FRZ LVL	Freezing level RH	052	004
REL HUMID ON P SFCS	Relative humidity on pressure surface	052	100
REL HUM IN BNDRY LYR	RH in boundary layer (30 mb mean)	052	116
REL HUM ON MDL SFCS	Relative humidity on model surface	052	109
ROUGHNESS LENGTH	Roughness length	083	001
SEA ICE MASK	Sea ice mask	091	001
SEA SFC TEMPERATURE	Sea surface temperature	080	001
SFC DRAG COEFFICIENT	Surface drag coefficient	252	001
SFC EXCHANGE COEF	Heat exchange coeff at surface	208	001
SFC MIDDAY ALBEDO	Surface midday albedo	084	001
SFC (SKIN) TEMPRATUR	Skin temperature	011	001
SFC U WIND STRESS	Surface u wind stress	124	001
SFC V WIND STRESS	Surface v wind stress	125	001
SFC WIND GUST	Surface wind gust	180	001
SHAL CU CLD BOT PRES	Shallow convective cloud bottom pressure	001	248
SHAL CU CLD TOP PRES	Shallow convective cloud top pressure	001	249
SHEAR AT TROPOPAUSE	Wind shear at tropopause	136	007
SHELTER DEWPOINT	2 M dew point temperature	017	105
SHELTER PRESSURE	2 M pressure	001	105
SHELTER REL HUMID	2 M RH	052	105
SHELTER SPEC HUMID	2 M specific humidity	051	105
SHELTER TEMPERATURE	2 M temperature	011	105
SHUELL MEAN SLP	Shuell sea level pressure	002	102
SLOPE TYPE	Slope type	222	001
SNOW DEPTH	Snow depth	066	001
SNOW FREE ALBEDO	Snow free albedo	170	001
SNOW ON MDL SFCS	Snow on model surface	171	109
SNOW ON P SFCS	Snow water on pressure surface	171	100
SNOW SUBLIMATION	Snow sublimation	198	001
SNOW WATER EQVALNT	Snow water equivalent	065	001
SOIL MOIST POROSITY	Soil moist porosity	240	001
SOIL MOIST REFERENCE	Soil moist reference	230	001
SOIL MOISTURE AVAIL	Soil moisture availability	207	112
SOIL MOISTURE	Soil moisture in between each of soil layers	144	112
SOIL MOIST WILT PT	Soil moist wilting point	219	001
SOIL TEMPERATURE	Soil temperature in between each of soil layers	085	112
SOIL TYPE	Soil type	224	001
SPC HUM IN BNDRY LYR	Specific humidity in boundary layer (30 mb mean)	051	116
SPEC HUM AT 10 M	10 M specific humidity	051	105
SPEC HUM ON MDL SFCS	Specific humidity on model surface	051	109
SPEC HUM ON P SFCS	Specific humidity on pressure surface	051	100
STORM REL HELICITY	Helicity	190	106
STRMFUNC ON MDL SFCS	Geostrophic streamfunction on model surface	035	109
STRMFUNC ON P SFCS	Geostrophic streamfunction on pressure surface	035	100
SURFACE DEWPOINT	Skin dew point temperature	017	001
SURFACE HEIGHT	Terrain height	007	001
SURFACE POT TEMP	Skin potential temperature	013	001
SURFACE PRESSURE	Surface pressure	001	001
SURFACE REL HUMID	Skin Relative humidity	052	001
SURFACE SPEC HUMID	Skin specific humidity	051	001

Appendix C – A Description of the EMS wrfpost output fields

Control File ID	Description	GRIB ID	Level
SW RAD TEMP TNDY	Temp tendency from shortwave radiative fl	250	109
TEMP AT FD HEIGHTS	Temperature at flight levels	011	103
TEMP AT TROPOPAUSE	Temperature at tropopause	011	007
TEMP IN BNDRY LYR	Temperature in boundary layer (30 mb me	011	116
TEMP ON MDL SFCS	Temperature on model surface	011	109
TEMP ON PRESS SFCS	Temperature on pressure surface	011	100
TOTAL CLD FRACTION	Total cloud fraction	071	200
TOTAL COL CONDENSATE	Column integrated total condensate	140	200
TOTAL COLUMN CLD ICE	Column integrated cloud ice	137	200
TOTAL COLUMN CLD WTR	Column integrated cloud water	136	200
TOTAL COLUMN RAIN	Column integrated rain	138	200
TOTAL COLUMN SNOW	Column integrated snow	139	200
TRBLNT KE ON MDL SFC	Turbulent kinetic energy on model surface	158	109
TRBLNT KE ON P SFCS	Turbulent kinetic energy on pressure surfa	158	100
U COMP STORM MOTION	U component storm motion	196	106
U WIND AT ANEMOM HT	10 M u component wind	033	105
U WIND AT FD HEIGHTS	U wind at flight levels	033	103
U WIND AT TROPOPAUSE	U wind at tropopause	033	007
U WIND IN BNDRY LYR	U wind in boundary layer (30 mb mean)	033	116
U WIND ON MDL SFCS	U component wind on model surface	033	109
U WIND ON PRESS SFCS	U component wind on pressure surface	033	100
V COMP STORM MOTION	V component storm motion	197	106
VEGETATION TYPE	Vegetation type	225	001
VISIBILITY	Visibility	020	001
V WIND AT ANEMOM HT	10 M v component wind	034	105
V WIND AT FD HEIGHTS	V wind at flight levels	034	103
V WIND AT TROPOPAUSE	V wind at tropopause	034	007
V WIND IN BNDRY LYR	V wind in boundary layer (30 mb mean)	034	116
V WIND ON MDL SFCS	V component wind on model surface	034	109
V WIND ON PRESS SFCS	V component wind on pressure surface	034	100

Appendix D: Getting your WRF EMS output displayed in AWIPS

Editor’s Note: All the guidance detailed in this section was assembled from multiple documents graciously provided by the brilliant contributors listed below. Any instruction that may be confusing or inaccurate is the fault of the editor alone.

Contributors in no particular order of greatness or flexibility:

Vasil Koleci	ITO	ALY
Paul Wolyn	SOO	PUB
Paul Shannon	ITO	AJK
Tom Fisher	ITO	LOX

Appendix Contents:

D.1 Ingesting GRIB 1 files in AWIPS

D.2 Ingesting GRIB 2 files in AWIPS

D.1 Ingesting GRIB 1 files in AWIPS

- a. Begin your journey as user **fxa** on **dx3**
- b. Create a directory under **/data/fxa/Grid/LOCAL/netCDF** for each domain what you will be sending to AWIPS. If your run has (multiple) nests, then you must create a directory for each nested domains well. The directory name may be almost anything you wish, for example:

```
%> mkdir /data/fxa/Grid/LOCAL/netCDF/mydomain
```

And

```
%> mkdir /data/fxa/Grid/LOCAL/netCDF/mynest
```

But you don’t need to use those cool directory names. For the sake of this guidance the directory shall henceforth be known as “**wrfems**”:

```
%> mkdir /data/fxa/Grid/LOCAL/netCDF/wrfems
```

Note that if you ever want to change the model dimensions or grid navigation, it is recommended that you remove the entire directory then recreate it. There have been issues with AWIPS not updating the template in this directory. Therefore, geographical changes to any model will not be processed properly (Dimension mismatch errors in the grib decoder or geography not lined up with model data).

- c. A “.cdl” file must be created/modified with the correct dimensions and forecast times for each domain and placed in the **/data/fxa/customFiles** directory on **dx1**. You may use the “.cdl” template files for the NMM and ARW cores floating around your Region or alternatively, you can use the “**wrfems.cdl**” file included with the WRF EMS in the **wrfems/util/awips** directory, just because EMS is your buddy.

Appendix D – Getting your WRF EMS output displayed in AWIPS

In the **wrfems.cdl** file you may have to change the following:

- dimensions:** Change the X and Y values to your NX and NY.
- data:** Change or take note of the variables as necessary
- model:** Provide model name to use for ingesting this data into GFE and for putting into the localGridSourceTable.txt file (coming up)
- cdlDate:** Just to keep track of changes made to the file
- n_valtimes:** The number of timesteps (model output times) – See **valtimeMINUSreftime**

The **wrfems.cdl** file has a 1-hourly display frequency specified in the file. If you want to go with a 3- or even 5.5-hourly output then the **valtimeMINUSreftime** section in the **wrfems.cdl** file will need to be changed. The times will have to be in the following format (in seconds):

```
valtimeMINUSreftime = 0, 3600, 7200, 10800, 14400, 18000, 21600, 25200, 28800, 32400, 36000, 39600, 43200, 46800, 50400, 54000, 57600, 61200, 64800, 68400, 72000, 75600, 79200, 82800, 86400, 90000, 93600, 97200, 100800, 104400, 108000, 111600, 115200, 118800, 122400, 126000, 129600, 133200, 136800, 140400, 144000, 147600, 151200, 154800, 158400, 162000, 165600, 169200, 172800;
```

Note that the above variable reflects an hourly forecast to be displayed in AWIPS out to 48 hours (3600 seconds in one hour times 48 hours). A 24-hour model would stop at 86400.

If you don't know the grid dimensions then see the next step. Again, be sure to copy any "**<file name>.cdl**" file into the **/data/fxa/customFiles** directory on **dx1** once you are done.

- d. Get the dimensions and grid navigation for the GRIB files you will be sending to AWIPS. You may need to run an initial simulation and convert the output to either GRIB 1 or 2 format. Then using the "**gribnav**" utility that came with your WRF EMS:

```
% gribnav <GRIB 1 or 2 file>
```

Appendix D – Getting your WRF EMS output displayed in AWIPS

The output of which will look something like (These values are correct with EMS V3.1):

% gribnav 0708171800_arw_d02.grb2f000000

File Name : 0708171800_arw_d02.grb2f000000
File Format : GRIB 2
Model ID : 116
Center ID : 7
Sub Center ID : 0

Grid Projection : **Lambert Conformal** - Assumed for the example

Grid Dimensions NX x NY : 201 x 201
Grid Spacing DX, DY : 4.00km, 4.00km
Standard Latitude : 43.00
Standard Longitude : -92.00
Latitude, Longitude (1,1) : 39.25, -96.71
True Lat1, Lat2 : 43.00, 43.00
Pole : North Pole
Scan : WE:SN

Corner Lat-Lon points of the domain:

46.425, -97.292	46.432, -86.866
*	*
	* 42.909, -92.124
*	*
39.248, -96.708	39.254, -87.432

The information that is most important for you is the following (your values will be different):

(1) True Lat1 : 43.00 - should be same as standard latitude
(2) Standard Longitude : -92.0
(3) Lower Left Lat (sw) : 39.248
(4) Lower Left Lon (sw) : -96.708
(5) Upper Right Lat (ne) : 46.432
(6) Upper Right Lon (ne) : -86.866
(7) NX Grid Dimension : 201
(8) NY Grid Dimension : 201

Editors Note: The values identified in 1 through 8 above are only used for a Lambert Conformal (LC) projection. You may display other projections in AWIPS but the guidance supplied here assumes LC projection for either the NMM or ARW core.

- e. Once you have mined the above information from your GRIB file, add the following line for each EMS domain grid to **/data/fxa/customFiles/makeDataSupps.patch**; note the (numbers)

Appendix D – Getting your WRF EMS output displayed in AWIPS

relate to the table above

maksuparg 3 (1) (2) (1) wrfems.sup l (3) (4) (5) (6)

Note #1: that the character after **wrfems.sup** is a lower-case “L”

Note #2: The leading “3” is for “Lambert Conformal” projection

Substituting the example values:

maksuparg 3 43.0 -92.0 43.0 wrfems.sup l 39.248 -96.708 46.432 -86.866

- f. Now, make sure that you are in the **/data/fxa/customFiles** directory and add the following line to the *end* of the **localGridSourceTable.txt** or **XXX-localGridSourceTable.txt** file for each grid you are sending (on a separate line):

```
|1|Grid/LOCAL/netCDF/wrfems |wrfems |wrfems |NX|NY|wrfems |2,3,4,5 |wrfems |wrfems |255-3 |wrfems | |DT
```

Where **DT** is the forecast frequency in hours, E.g., 1, 2, 3, etc., and **NX** and **NY** and the grid dimensions for that model grid obtained earlier. So the above example would look like:

```
|1|Grid/LOCAL/netCDF/wrfems |wrfems |wrfems |201|201|wrfems |2,3,4,5 |wrfems |wrfems |255-3 |wrfems | |1
```

- g. Add the following lines to **activeGridSources.txt** or **XXX-activeGridSources.txt**:

wrfems – Yes, that is all

- h. The MDLID in the GRIB files are sent to AWIPS has to be defined for each domain. This step is very important because the MDLID, along with the originating center ID in the GRIB file, uniquely defines the domain. This is how the GRIB decoder knows a GRIB file is from a specific model and domain.

Fortunately for you the **MDLID** (Model ID), **OCNTR** (Originating Center), and **SCNTR** (Sub Center) are also provided by provided “**gribnav**” utility. From the output above:

```
Model ID      : 116    MDLID
Center ID     : 7      OCNTR
Sub Center ID : 0      SCNTR
```

The above Model ID indicates that the data are from an ARW core (116) run (112 for the NMM core), that the Originating Center is NCEP Operations (7; 9 for WFO Center). These values originate from the **conf/ems_post/post_grib.conf** file read when processing your EMS output with **ems_post**. They may be changed; however, *and this is a big however*, many of the WRF EMS routines require a Center ID of 7 (NCEP Operations) as it identifies the grid table used to define the names of all the really cool fields available to you, such as updraft helicity and 1km intra-period maximum reflectivity (Ya, I know, calm down). So, by changing these values, you run the risk of the maximum reflectivity field being identified as “squid fertility index”, which has limited value in operational weather forecasting. But I digress.

Appendix D – Getting your WRF EMS output displayed in AWIPS

You will need to make sure the Model ID is not already in use as part of the AWIPS baseline. To determine whether a Model ID is being used, look at, *but do not edit*, the `/awips/fxa/data/gribModelsNCEP_ed1.txt` file.

Once you have confirmed that the Model ID is not being used, edit the `gribModelsNCEP_ed1.txt` or `XXX-gribModelsNCEP_ed1.txt` files in the `/data/fxa/customFiles` directory and add the following line:

MDLID|<name> |<description>

Where **MDLID** is the Model ID number (**116**), **<name>** is the name we have need using to identify the EMS domain (**wrfems**), and **<description>** is just that. So, following the previous examples the line may look something like:

116|wrfems|The EMS is the greatest

Note that the Model ID uniquely defines the domain in AWIPS, so if you have multiple domains, e.g., a primary domain and its child (nest), you will need multiple Model IDs. Each Model ID will need to be defined on a separate line, similar to that specified above.

Fortunately, the EMS allows for multiple Model IDs in `conf/ems_post/post_grib.conf`.

- i. Check for the following line:

`NCEP_UNKNOWN_ed1|gribGridsNCEP_ed1.txt|gribModelsNCEP_ed1.txt|gribParametersNCEP_ed1.txt`

In these files:

`/data/fxa/customFiles/gribParametersNCEP_ed1.txt`

Or

`/data/fxa/customFiles/XXX-gribParametersNCEP_ed1.txt` (where XXX = site ID)

If the specified line is missing from the above files you will need to add it to the

`/awips/fxa/data/gribTableInfo.txt` (on dx3)

file on **dx3**, and then copy `gribTableInfo.txt` to the `/awips/fxa/data` directory on every other AWIPS machine. Adding the above line should correct the error in the GribDecoder logfile on dx3:

PROBLEM: NEED TO ADD NCEP_UNKNOWN_ed1 to NCEP_UNKNOWN_ed1| gribGridsNCEP_ed1.txt | gribModelsNCEP_ed1.txt

But the error isn't quite this small.

- j. With the WRF EMS, there are many fields available that are not from the standard list of NCEP fields, such as “updraft helicity”, “maximum intra-period 1km reflectivity”, and “squid fertility index”. If you want these data to be processed by the GRIB decoder, you will have to create a

Appendix D – Getting your WRF EMS output displayed in AWIPS

unique `gribParametersXXXX_ed1.txt` file that includes the correct GRIB field identification information for these fields so that they can be processed in AWIPS. In addition, you must modify the appropriate `.cdl` template file. It is recommended that you read the AWIPS documentation for more information. Maybe, someday, this information will be available from this document, but not today.

- k. As user **fxa**, from the `/awips/fxa/data/localization/scripts` directory on **dx3**, run the following:

```
%> ./mainScript.csh f -dirs -dataSups -grids
```

Note: The use of “-dirs” will allow for the creation of a new directory and template file. The “f” option will force the system to create all the necessary files in `/awips/fxa/data/localizationDataSets/XXX`.

If you get an “**ncgen: syntax error line 33**” error running **mainScript.csh**, you will have to remove the **wrf_nmm.sup** file and then try again.

If no errors were encountered on the `dx3` localization and it is correct, then execute the localizations on the other AWIPS machines as directed below. As a convenience, the **wrfems/util/awips/localizegrids.csh** script is provided that will allow you to localize each of the AWIPS machines rather than doing the task manually. For those “Please sir, may I have another?!” users, you may do it yourself:

```
%> ssh dx1 /awips/fxa/data/localization/scripts/mainScript.csh f -dataSups -grids
%> ssh dx2 /awips/fxa/data/localization/scripts/mainScript.csh f -dataSups -grids
%> ssh dx4 /awips/fxa/data/localization/scripts/mainScript.csh f -dataSups -grids

%> ssh px1 /awips/fxa/data/localization/scripts/mainScript.csh f -dataSups -grids
%> ssh px2 /awips/fxa/data/localization/scripts/mainScript.csh f -dataSups -grids

%> ssh lx1 /awips/fxa/data/localization/scripts/mainScript.csh f -dataSups -grids
%> ssh lx2 /awips/fxa/data/localization/scripts/mainScript.csh f -dataSups -grids
%> ssh lx3 /awips/fxa/data/localization/scripts/mainScript.csh f -dataSups -grids
%> ssh lx4 /awips/fxa/data/localization/scripts/mainScript.csh f -dataSups -grids
%> ssh lx5 /awips/fxa/data/localization/scripts/mainScript.csh f -dataSups -grids
```

- l. Kill the **GribDecoder** process on **dx3**:

```
%> killProc GribDecoder
```

The **GribDecoder** will restart immediately.

- m. On **dx1** as user **fxa** get the grid keys for the WRF grids on your AWIPS.

```
%> grep -i wrf /awips/fxa/data/localizationDataSets/XXX/gridNetcdfKeys.txt
```

Where `XXX` is replaced by your site ID.

You might see something like:

Appendix D – Getting your WRF EMS output displayed in AWIPS

1070000061 | | | | | Grid/LOCAL/netCDF/wrfems | | | raw wrfems grids

Here, **1070000061** is the grid key number and **Grid/LOCAL/netCDF/wrfems** is the storage location for the WRF EMS netCDF data files.

Set up the purging on **dx1**. Edit the **/data/fxa/customFiles/localPurgeInfo.txt** file and add each key and purge requirements. For example:

1070000061 | | r| 1- 3 // ARW WRF 12km

Where: **1070000061** is the model key from above and

| | Is the directory of the data to be deleted resides. Should be blank because the px servers should already know the location from **gridNetcdfKeys.txt**.

r Tells the purger to delete recursively. Optional.

1- Tells the purger to key anything less than 1 day.

3 Keeps only 3 versions of the WRF.

Next, from the **/awips/fxa/data/localization/scripts** directory on **dx1**, run a **-purge** localization as user **fxa**:

```
%> ./mainScript.csh -purge
```

Restart the purge process and notification server:

```
%> stopPurgeProcess
```

```
%> startPurgeProcess
```

Finally, from the **/awips/fxa/data/localization/scripts** directory on **dx2**, run a **-purge** localization as user **fxa**:

```
%> ./mainScript.csh -purge
```

- n. Enable the AWIPS ingest process. *You will need to be **user ldad** on **px2** for the next several steps to accomplish this task.*

You must be located in the **/data/fxa/LDAD/data** directory.

Examine the LDADinfo.txt file and look for a line that handles the “Grb1F” pattern, which should look something like:

GrbF | | | | |preProcessGRIB1.pl

The actual string pattern used will have to match that of the GRIB 1 files being sent to AWIPS from your WRF EMS, so you may need to change changed the above “GrbF” to something like “grib”, “grb”, GrbF”, or “turtle”.

Appendix D – Getting your WRF EMS output displayed in AWIPS

If you don't already have a **preProcessGRIB1.pl** file you will have to get one. The WRF EMS includes a **preProcessGRIB1.pl** file in the **wrfems/util/awips** directory or you may get the file from a Regional source.

Make sure the permissions are correctly set on the file:

```
%> chmod 775 preProcessGRIB1.pl
```

Then copy the **preProcessGRIB1.pl** file to the **/awips/fxa/ldad/bin** directory on **px1**:

```
%> scp -p preProcessGRIB1.pl px1:/awips/fxa/ldad/bin
```

Finally, restart the LDAD process on **px2** (as user **ldad**):

```
%> /awips/fxa/ldad/bin/stopLDAD.sh  
%> /awips/fxa/ldad/bin/startLDAD.csh
```

- o. Configure WRF EMS to send GRIB forecast files to AWIPS as directed in the **ems_post** configuration files. Make sure that the GRIB files have unique names for each forecast hour and each grid. If you do not differentiate between domain numbers in the GRIB filename, then the grib file for one domain may overwrite the grib file for another domain before it can be ingested into AWIPS.

In the **conf/ems_post/post_grib.conf** file, you should use the domain number (WD) option when defining the GRIBNAME. A suggested option is:

```
GRIBNAME = YMMDDHHMN_CORE_dWD_FH.GrbF
```

This definition includes the WD option and it has a file suffix of GrbF.

Congratulations, your task is completed.

D.2 Ingesting GRIB 2 files in AWIPS

Note: *In order to ingest WRF EMS GRIB 2 files you first must create them as they do not just magically fall from the sky, but rather, they fall magically from your WRF EMS provided you have **GRIB2 = Yes** in your `post_grib.conf` file.*

- a. Begin your journey as user **fxa** on **dx3**
- b. Create a directory under `/data/fxa/Grid/LOCAL/netCDF` for each domain what you will be sending to AWIPS. If your run has (multiple) nests, then you must create a directory for each nested domains well. The directory name may be almost anything you wish, for example:

```
%> mkdir /data/fxa/Grid/LOCAL/netCDF/mydomain
```

And

```
%> mkdir /data/fxa/Grid/LOCAL/netCDF/mynest
```

But you don't need to use those cool directory names. For the sake of this guidance the directory shall henceforth be known as "**wrfems**":

```
%> mkdir /data/fxa/Grid/LOCAL/netCDF/wrfems
```

Note that if you ever want to change the model dimensions or grid navigation, it is recommended that you remove the entire directory then recreate it. There have been issues with AWIPS not updating the template in this directory. Therefore, geographical changes to any model will not be processed properly (Dimension mismatch errors in the grib decoder or geography not lined up with model data).

- c. A ".cdl" file must be created/modified with the correct dimensions and forecast times for each domain and placed in the `/data/fxa/customFiles` directory on **dx1**. You may use the ".cdl" template files for the NMM and ARW cores floating around your Region or alternatively, you can use the "**wrfems.cdl**" file included with the WRF EMS in the `wrfems/util/awips` directory, just because EMS is your buddy.

In the **wrfems.cdl** file you may have to change the following:

dimensions: Change the X and Y values to your NX and NY.
data: Change or take note of the variables as necessary
model: Provide model name to use for ingesting this data into GFE and for putting into the `localGridSourceTable.txt` file (coming up)
cdlDate: Just to keep track of changes made to the file
n_valtimes: The number of timesteps (model output times) – See **valtimeMINUSreftime**

The **wrfems.cdl** file has a 1-hourly display frequency specified in the file. If you want to go with a 3- or even 5.5-hourly output then the **valtimeMINUSreftime** section in the **wrfems.cdl** file will need to be changed. The times will have to be in the following format (in seconds):

```
valtimeMINUSreftime = 0, 3600, 7200, 10800, 14400, 18000, 21600, 25200, 28800,  
32400, 36000, 39600, 43200, 46800, 50400, 54000, 57600, 61200, 64800, 68400,  
72000, 75600, 79200, 82800, 86400, 90000, 93600, 97200, 100800, 104400,
```

Appendix D – Getting your WRF EMS output displayed in AWIPS

108000, 111600, 115200, 118800, 122400, 126000, 129600, 133200, 136800, 140400, 144000, 147600, 151200, 154800, 158400, 162000, 165600, 169200, 172800;

Note that the above variable reflects an hourly forecast to be displayed in AWIPS out to 48 hours (3600 seconds in one hour times 48 hours). A 24-hour model would stop at 86400.

If you don't know the grid dimensions then see the next step. Again, be sure to copy any “<file name>.cdl” file into the /data/fxa/customFiles directory on dx1 once you are done.

- d. Get the dimensions and grid navigation for the GRIB files you will be sending to AWIPS. You may need to run an initial simulation and convert the output to either GRIB 1 or 2 format. Then using the “gribnav” utility that came with your WRF EMS:

```
% gribnav <GRIB 1 or 2 file>
```

The output of which will look something like (These values are correct with EMS V3.1):

```
% gribnav 0708171800_arw_do2.grb2f000000
```

```
File Name           : 0708171800_arw_do2.grb2f000000
File Format          : GRIB 2
Model ID            : 116
Center ID           : 7
Sub Center ID       : 0
```

```
Grid Projection      : Lambert Conformal - Assumed for the example
```

```
Grid Dimensions NX x NY : 201 x 201
Grid Spacing DX, DY : 4.00km, 4.00km
Standard Latitude      : 43.00
Standard Longitude     : -92.00
Latitude, Longitude (1,1) : 39.25, -96.71
True Lat1, Lat2       : 43.00, 43.00
Pole                   : North Pole
Scan                   : WE:SN
```

Corner Lat-Lon points of the domain:

```
46.425, -97.292      46.432, -86.866
*
*
* 42.909, -92.124
*
*
39.248, -96.708      39.254, -87.432
```

Appendix D – Getting your WRF EMS output displayed in AWIPS

The information that is most important for you is the following (your values will be different):

- (1) **True Lat1** : **43.00** - *should be same as standard latitude*
- (2) **Standard Longitude** : **-92.0**
- (3) **Lower Left Lat (sw)** : **39.248**
- (4) **Lower Left Lon (sw)** : **-96.708**
- (5) **Upper Right Lat (ne)** : **46.432**
- (6) **Upper Right Lon (ne)** : **-86.866**
- (7) **NX Grid Dimension** : **201**
- (8) **NY Grid Dimension** : **201**
- (9) **DX Grid Spacing** : **4.00**
- (10) **DY Grid Spacing** : **4.00**
- (11) **Projection** : **3 (Lambert Conformal)**

Note that the Projection number (3) comes from the tables below:

1	Stereographic	7	Satellite View
2	Orthographic	8	Cylindrical Equidistant
3	Lambert Conformal	9	Mercator
4	Azimuthal Equal Area	10	Mollweide
5	Gnomonic	16	Azimuth Range
6	Azimuthal Equidistant	17	Radar Az_Ran

- e. The MDLID in the GRIB files sent to AWIPS has to be defined for each domain. This step is very important because the MDLID, along with the originating center ID in the GRIB file, uniquely defines the domain. This is how the GRIB decoder knows a GRIB file is from a specific model and domain.

Fortunately for you the **MDLID** (Model ID), **OCNTR** (Originating Center), and **SCNTR** (Sub Center) are also provided by provided “**gribnav**” utility. From the output above:

```

Model ID      : 116      MDLID
Center ID     : 7        OCNTR
Sub Center ID : 0        SCNTR
  
```

The above Model ID indicates that the data are from an ARW core (116) run (112 for the NMM core), that the Originating Center is NCEP Operations (7; 9 for WFO Center). These values originate from the **conf/ems_post/post_grib.conf** file read when processing your EMS output with **ems_post**. They may be changed; however, *and this is a big however*, many of the WRF EMS routines require a Center ID of 7 (NCEP Operations) as it identifies the grid table used to define the names of all the really cool fields available to you, such as updraft helicity and 1km intra-period maximum reflectivity (Ya, I know, calm down). So, by changing these values, you run the risk of the maximum reflectivity field being identified as “squid fertility index”, which has limited value in operational weather forecasting. But I digress.

The default behavior of the WRF EMS is to assign “**20 + domain number**” as the Sub Center ID

Appendix D – Getting your WRF EMS output displayed in AWIPS

(SCNTR). Using a value of “0” or greater than 20 avoids conflicts with predefined Sub Center values. If the SCNTR value is something other than 0, then make sure the Sub Center is defined in `/awips/fxa/data/grib2Centers-SubCenters.txt`, where a line like the one below should be added to the **TABLE OF SUB-CENTERS for CENTER 7(NCEP)** section.

```
21 |XXX |NWS XXX <- Where XXX is your STID
```

- f. Once you have mined the above information from your GRIB file, run the **maksuparg** command from the `/data/fxa/customFiles` directory. The command syntax look something like:

```
%> maksuparg (11) (1) (2) (1) wrfems.sup l (3) (4) (5) (6)
```

Where the (numbers) correspond to the values above.

Note #1: that the character after **wrfems.sup** is a lower-case “L”

Note #2: The leading “3” is for “Lambert Conformal” projection

Substituting the example values:

```
%> maksuparg 3 43.0 -92.0 43.0 wrfems.sup l 39.248 -96.708 46.432 -86.866
```

- g. Add your model data grid information to the `/awips/fxa/data/grib2Grids.txt` file using the values from the table created above. **You will likely need to copy the file to /home/XXX to prevent the file from being overwritten during later builds.** Note that **<Grid>** is a unique grid name that will be used in a later step to tell the Grib2Decoder which grid is associated with this grib data. It can be any name you want as long as it is unique.

```
#Name |GRIB id|Nx |Ny | Lat1 | Lon1 |Dx |Dy |Description  
#-----|-----|-----|-----|-----|-----|-----|-----|-----
```

```
<Grid> | |<7>|<8>| <3> | <4> | <9>| <10> | <Descriptive Name>
```

Add the model data line to the TableGridNCEP section if you used the NCEP center (OCNTR = 07) in your model configuration file. Add it to the TableGridNWSO section if you used the WFO center (OCNTR = 09).

Leaving the second column (GRIB id) blank appears to work.

With the information collected from the above GRIB 2 file:

```
WRFEMS | | 201 | 201 | 39.248| 263.292| 4.00 | 4.00 | WRF ARW
```

Note: The longitude must be defined in values of 0-360. So, in the above example,

```
263.292 = 360 - 96.708 (from gribnav)
```

- h. Edit the `/awips/fxa/data/grib2Models.txt` file and add a line similar to the following one, substituting your **<MDLID>** and **<modelName>** values.

Appendix D – Getting your WRF EMS output displayed in AWIPS

```
#Grid      | Name      | Description
#-----|-----|-----
<MDLID> | <modelName> | <Descriptive name for your local WRF Model>
```

If your OCNTR value is 07 (originating center of NCEP) add the line to the section **TableModelNCEP**. Note: If you use a **MDLID** of 112 (NMM) or 116 (ARW) these lines are already in the **TableModelNCEP** section of **grib2Models.txt**.

If your **OCNTR** is 09 (originating center of NWS field office) with an **SCNTR** corresponding to your office, add the line to the **TableModelNWSO** section (create the section if necessary).

For the above example:

116 | wrfems | Local WRF ARW model at 4KM resolution

- i. Again, if your originating center (OCNTR) is 07 and your Sub Center is some thing other than 0, then you will have to make sure the Sub Center is defined in **/awips/fxa/data/grib2Centers-SubCenters.txt**, where a line like the one below should be added to the **TABLE OF SUB-CENTERS for CENTER 7(NCEP)** section.

21 |XXX |NWS XXX <- Where XXX is your STID

Note: Copy /awips/fxa/data/grib2Centers-SubCenters.txt to /home/XXX to prevent the files from getting overwritten in upgrades.

- j. Now, make sure that you are in the **/data/fxa/customFiles** directory and add the following line to the *end* of **XXX-localGridSourceTable.txt** file for each grid you are sending (on a separate line):

```
alias|1|<directory>|<cdlfile>|<geofile>|<Nx>|<Ny>|<title>|<scales>|<name>|<topofile>|<grid>|<modelName>|parameters|<dt in hours>
```

The alias and parameters fields can be left blank.

The other fields are defined as:

- directory** : Where the netCDF files for this model are found, relative to /data/fxa.
- cdlfile** : Matches the prefix of your WRF .cdl file in /data/fxa/customFiles.
- geofile** : Matches the prefix of the .sup file created by maksuparg.
- NX and NY**: Values <7> and <8> from the table.
- title** : Name you want to be displayed in the volume browser source menu
- scales** : Scales this model will be displayed on (Generally 0-5 with 1 being hemispheric and 5 being the smallest scale in D2D)
- name** : Name assigned to the "model" variable in the WRF .cdl file in /data/fxa/customFiles.
- topofil** : The .topo file is created during a -grids localization. The name comes from the prefix of the .sup file created with maksuparg.
- grid** : "Name" (first) parameter from the WRF line in the grib2Grids.txt

Appendix D – Getting your WRF EMS output displayed in AWIPS

file.
processes : "Name" (second) parameter from the WRF line in the grib2Models.txt file.
DT : The output frequency of the WRF model in hours

So the above example would look like:

```
|1|Grid/LOCAL/netCDF/wrfems |wrfems |wrfems |201|201|wrfems |2,3,4,5 |wrfems |wrfems |wrfems |WRFEMS | |1
```

- k. Add the following lines to **activeGridSources.txt** or **XXX-activeGridSources.txt**:

wrfems – Yes, that is all

- l. With the WRF EMS, there are many fields available that are not from the standard list of NCEP fields, such as “updraft helicity”, “maximum intra-period 1km reflectivity”, and “squid fertility index”. If you want these data to be processed by the GRIB decoder, you will have to create a unique grib2ParametersXXXX_ed1.txt file that includes the correct GRIB field identification information for these fields so that they can be processed in AWIPS. In addition, you must modify the appropriate .cdl template file. It is recommended that you read the AWIPS documentation for more information. Maybe, someday, this information will be available from this document, but not today.
- m. As user **fxa**, from the **/awips/fxa/data/localization/scripts** directory on **dx3**, run the following:

```
%> ./mainScript.csh f -dirs -dataSups -grids
```

Note: The use of “-dirs” will allow for the creation of a new directory and template file. The “f” option will force the system to create all the necessary files in /awips/fxa/data/localizationDataSets/XXX.

If no errors were encountered on the dx3 localization and it is correct, then execute the localizations on the other AWIPS machines as directed below. As a convenience, the **wrfems/util/awips/localizegrids.csh** script is provided that will allow you to localize each of the AWIPS machines rather than doing the task manually. For those “Please sir, may I have another” users, you may do it yourself:

```
%> ssh dx1 /awips/fxa/data/localization/scripts/mainScript.csh f -dataSups -grids -dirs
%> ssh dx2 /awips/fxa/data/localization/scripts/mainScript.csh f -dataSups -grids -dirs
%> ssh dx4 /awips/fxa/data/localization/scripts/mainScript.csh f -dataSups -grids -dirs

%> ssh px1 /awips/fxa/data/localization/scripts/mainScript.csh f -dataSups -grids -dirs
%> ssh px2 /awips/fxa/data/localization/scripts/mainScript.csh f -dataSups -grids -dirs
```

Slightly different command:

```
%> ssh lx1 /awips/fxa/data/localization/scripts/mainScript.csh f -dataSups -grids
%> ssh lx2 /awips/fxa/data/localization/scripts/mainScript.csh f -dataSups -grids
%> ssh lx3 /awips/fxa/data/localization/scripts/mainScript.csh f -dataSups -grids
%> ssh lx4 /awips/fxa/data/localization/scripts/mainScript.csh f -dataSups -grids
%> ssh lx5 /awips/fxa/data/localization/scripts/mainScript.csh f -dataSups -grids
```

Appendix D – Getting your WRF EMS output displayed in AWIPS

- n. Kill the **GribDecoder** process on **dx3**:

```
%> killProc Grib2Decoder
```

The **Grib2Decoder** will restart immediately.

- o. On **dx1** as user **fxa** get the grid keys for the WRF grids on your AWIPS.

```
%> grep -i <model name> /awips/fxa/data/localizationDataSets/XXX/gridNetcdfKeys.txt
```

Where XXX is replaced by your site ID.

You might see something like:

```
1070000061 | | | | | Grid/LOCAL/netCDF/wrfems | | | raw wrfems grids
```

Here, **1070000061** is the grid key number and **Grid/LOCAL/netCDF/wrfems** is the storage location for the WRF EMS netCDF data files.

Set up the purging on **dx1**. Edit the **/data/fxa/customFiles/localPurgeInfo.txt** file and add each key and purge requirements. For example:

```
1070000061 | | r| 1- | 3 // ARW WRF 12km
```

Where: **1070000061** is the model key from above and

| | Is the directory of the data to be deleted resides. Should be blank because the px servers should already know the location from **gridNetcdfKeys.txt**.

r Tells the purger to delete recursively. Optional.

1- Tells the purger to key anything less than 1 day.

3 Keeps only 3 versions of the WRF.

Next, from the **/awips/fxa/data/localization/scripts** directory on **dx1**, run a **-purge** localization as user **fxa**:

```
%> ./mainScript.csh -purge
```

Restart the purge process and notification server:

```
%> stopPurgeProcess
```

```
%> startPurgeProcess
```

Finally, from the **/awips/fxa/data/localization/scripts** directory on **dx2**, run a **-purge** localization as user **fxa**:

```
%> ./mainScript.csh -purge
```

- p. Enable the AWIPS ingest process. You will need to be **user ldad** on **px2** for the next several steps to accomplish this task.

You must be located in the **/data/fxa/LDAD/data** directory.

Examine the LDADinfo.txt file and look for a line that handles the “Grb2F” pattern, which should look something like:

```
Grb2F | | | | |preProcessGRIB2.pl
```

The actual string pattern used will have to match that of the GRIB 2 files being sent to AWIPS from your WRF EMS, so you may need to change changed the above “Grb2F” to something like “grib2”, “grb2”, “Grb2F”, or “turtle2”.

If you don’t already have a **preProcessGRIB2.pl** file you will have to get one. The WRF EMS includes a **preProcessGRIB2.pl** file in the **wrfems/util/awips** directory or you may get the file from a Regional source.

Make sure the permissions are correctly set on the file:

```
%> chmod 775 preProcessGRIB2.pl
```

Then copy the **preProcessGRIB2.pl** file to the **/awips/fxa/ldad/bin** directory on **px1**:

```
%> scp -p preProcessGRIB2.pl px1:/awips/fxa/ldad/bin
```

Finally, restart the LDAD process on **px2** (as user **ldad**):

```
%> /awips/fxa/ldad/bin/stopLDAD.sh
```

```
%> /awips/fxa/ldad/bin/startLDAD.csh
```

- q. Configure WRF EMS to send GRIB forecast files to AWIPS as directed in the **ems_post** configuration files. Make sure that the GRIB 2 files have unique names for each forecast hour and each grid. If you do not differentiate between domain numbers in the GRIB filename, then the grib file for one domain may overwrite the grib file for another domain before it can be ingested into AWIPS.

In the **conf/ems_post/post_grib.conf** file, you should use the domain number (WD) option when defining the GRIBNAME. A suggested option is:

```
GRIBNAME = YMMDDHHMN_CORE_dWD_FH.Grb2F
```

This definition includes the WD option and it has a file suffix of Grb2F.

Congratulations, your task is completed.

Exercises: Information regarding the WRF EMS workshop exercises

Chapter Contents:

- E.1** Some information for the caretaker of the WRF EMS
- E.2** System Requirements
- E.3** Data Sets Provided for the Exercises
- E.4** A brief summary of the experiments
- E.5** Summary of the 17-18 August 2007 Event
- E.6** Exercise #1 - Installing and Benchmarking the WRF EMS
- E.7** Exercise #2 - Introduction to the WRF EMS
- E.8** Exercise #3 - Sensitivity of a Simulation to Model Configuration
- E.9** Exercise #4 - Running a Nested Simulation
- E.10** Exercise #5 - Real-time NWP with the WRF EMS

E.1 Some information for the caretaker of the WRF EMS

In the following sections of this nearly complete guide you will find a series of exercises that may be used to learn the process of installing, configuring, and running the WRF EMS. These materials have been used, in various forms, for a numerous international workshops on NWP and may easily be adapted to meet specific local learning objectives. All the data needed to conduct these exercises are included with the WRF EMS package including html pages of the results and both PDF and MS Word files of these instructions, all of which can be found below the `wrfems/util/workshop` directory.

E.2 System Requirements

The amount of time required to run the experiments greatly depends upon the amount of processing power available on the system used. The single domain experiments presented in this guide, experiments 2 and 3, require a minimum of 3 Gb of physical memory to run. If the system contains less than the required amount of memory it is suggested that the number of horizontal grid points be reduced and the grid spacing increased such that the areal coverage of the computation domain remains the same as that presented.

E.3 Data Sets Provided for the Exercises

The WRF EMS package includes a number of data sets in GRIB 2 format for use with the exercises. These data are located in the `wrfems/util/workshop/wrfems/data` directory and have been subset to reduce the size of the files and eliminate any fields not used for model initialization. The areal coverage of the data sets extends just beyond computational domain defined in each experiment, so increasing the intended size of the domain may result in a failure during model initialization.

You may substitute another data set(s) but be sure to modify the directions presented as necessary.

Also, the “labgfs_gribinfo.conf”, “labnam218_gribinfo.conf”, and “labnam212_gribinfo.conf” files may need to be modified as necessary.

Also, PowerPoint and PDF files that summarize the event used for the exercises are provided in the **wrfems/util/workshop/wrfems** directory.

E.4 A brief summary of the experiments

Exercise #1 - Installing and Benchmarking the WRF EMS

This material leads the student through the process of installing the WRF EMS system on a single workstation and then testing the installation by running one of the benchmark cases. The critical learning objective in this exercise is the process of installing and testing the EMS on a workstation; and not the interrogation of the simulation results.

Exercise #2 - Introduction to the WRF EMS

In this exercise, the student will configure and run a short ARW or NMM core simulation for the purpose of becoming familiar with the WRF EMS package. Step-by-step directions will guide the student through the process of setting up and running the model, and there will be a few questions along the way that will require some investigation of the system. The critical learning objective in this exercise is the process of running the model and not the interrogation of the forecast; however, students will have an opportunity to examine and share results with others if time allows.

Exercise #3 - Sensitivity of a Simulation to Model Configuration

For this exercise, the student will set up and run the WRF EMS to test the sensitivity of a forecast to different controllable parameters. A series of unique experiments is provided (Appendix A) that should be run and the results compared to a control simulation (Appendix B). Due to the time required to run, the simulations are started during an initial period and then processing and data analysis occur at a later time. The critical learning objective in this exercise is the sensitivity of model simulation to controllable parameters.

Exercise #4 - Running a Nested Simulation

**Note: As configured, nested simulations more than 6Gb of physical memory*

This material leads the student through the process of setting up and running a nested simulation with either the WRF ARW or NMM core. As with the previous EMS exercises, step-by-step directions will provide guidance through the process. There will also be a few questions along the way that will require the student to investigate various parts of the system. The critical learning objective in this exercise is the process of running a nested simulation and post-processing the forecast data. If time allows, students will have an opportunity to examine the results but that task is not critical.

Exercise #5 - Real-time NWP with the WRF EMS

For this exercise, the students will set up and run the WRF EMS for the purpose of making a real-time forecast. They will be using the `ems_autorun` routine initiated from a cron that will execute each of the steps from downloading the initialization files, processing the data, running the model, and post processing the forecasts. Students have control over the configuration of the computational domain and forecast run. The primary objective is to have a 24 hour forecast completed and available within 4 hours from the start of the process. The runs will be initiated during a first period and analysis completed during a later period. Each group will then present their forecasts to the class in a brief 10-minute presentation.

E.5 Summary of the 17-18 August 2007 Event

The data provided for the WRF EMS are for a Heavy rain and flash flood event that occurred over a 2 day period across the US states of Iowa, Wisconsin, and Minnesota from 17 to 19 August 2007. During this period a quasi-stationary warm front was located to the south of the impacted region with nearly continuous redevelopment of thunderstorms to the west, which moved eastward over the same area. Rainfall amounts over this period in excess of 300mm were commonplace with an event total of 462mm recorded at La Crescent, Minnesota.

PowerPoint and PDF files that summarize the event used for the exercises are provided in the `wrfems/util/workshop/wrfems` directory

Workshop Exercise #1 - Installing and Benchmarking the WRF EMS

For this exercise, you will install the WRF EMS on your workstation and then run a benchmark case for either the NMM or ARW core. Step-by-step directions will guide you through the process. The critical learning objective in this exercise is the process of installing and testing the EMS on your workstation; and not the interrogation of the simulation results. Feel free to ask for assistance should you encounter any problems.

There are a few system requirements for installing and running the WRF EMS, which are listed below:

- **A relatively current Linux distribution, but not too current** – The WRF EMS has been tested on Red Hat Enterprise, Fedora, CentOS, SuSe, and Ubuntu distributions, although the use of Ubuntu has required some changes to the (dash/bash) interpreter in the past. Other distributions will probably work; however, it's simply too difficult to keep up with all the releases and updates. There is typically a lag before the EMS developer can install a newly released OS and run the tests. So just stick with what works.
- **8 Gb of available disk space** – This requirement pertains to the installation of the EMS only. Of course, running an NWP model can use up a significant amount of disc space so this requirement should be considered as an absolute minimum.
- **The T|C shell must be installed on the system** – Linux purists may get all upset about this requirement but that's just the way it is for now; however, it will likely change in the future. Note that in many current Linux releases, the default is not to install **tcsh**, so you may have to install it separately. The `ems_install.pl` routine will check whether it is installed and provide an error message.
- **The EMS user must be using a T|C shell** – If you are installing the EMS under a known user then they must be running **csh** or **tcsh**; otherwise horrible things may happen such as the EMS failing to run. And you don't want that to happen. There are ways around this requirement but we're not going there now.
- **Root permission and/or write permission on a large disc partition.** The `ems_install.pl` routine was designed to be run a root user. You will have the opportunity to assign ownership to an existing or new user during the processes. That said, the EMS can be installed by an existing user provided the user has write permission on the drive were the EMS will reside.

During the installation process install routine will attempt to do the following:

- Provide you with a greeting, just because that's important and you're worth the effort
- Prompt you for the installation directory (Default: `/usr1`)
- Check that the directory exists and whether you have write permission
- Determine whether an existing EMS installation resides at that location:
 - Get the version of existing installation
 - Ask whether you want to rename the existing installation to `wrfems.<release>`
- Prompt you for the name of the user to assign ownership of the package
- Create a new account and home directory if the user does not exist
- Check that the user's login shell is either `tcsh` or `csh`
- Prompt you for a password if a new user was created
- Finally install the EMS from the specified source
- Do the post-install configuration

- Congratulate you on yet another wise decision

Note that all sorts of useful information will be printed to the screen while the installation is running so don't leave the room.

The installation process will also attempt to:

- Determine the appropriate run-time binaries based on the CPU type on your system. See section 2.5 for more on target binaries.
- Attempt to determine the number of physical CPU and cores per CPU on your system. Should a problem occur, you will be prompted to give up this information and you will be powerless to resist.
- Install an entry (disabled) in the user's crontab file to automate the process of updating the EMS. You will be able to configure the `ems_install.pl` utility to automatically download and install updates from the SOO/STRC EMS servers and notify user(s) of any changes. More on the updating capabilities utility later.
- Install an entry (again, disabled) in the user's crontab file to assist in the automation of real-time forecasting. All you need to do is make the appropriate changes to this entry and you're off!

See, isn't that simple? The WRF EMS practically installs itself.

Depending on the speed of your DVD reader and hard disks, the installation should take ~5 to 15 minutes. Most of this time is spent unpacking very large surface data sets for topography and land use.

Step I Installing the WRF EMS - Success is only a few keystrokes away!

Installation of the WRF EMS requires the use of `ems_install.pl`, which was developed to tackle all the challenges that you might encounter if you were to attempt a manual installation of an NWP system. It is highly recommend that you use the most current version of this routine as your life will become less complicated and more rewarding if you do. *And that's a statement only a few modeling systems can make!*

It is recommended that the installation be conducted by user root; however, it may be installed by another user provided that you have full write permission on the partition where the package will reside.

The guidance provided in this exercise assumes that the installation is being conducted as root user, as indicated by the “#” prompt. If the installation is not being done by root, then the existing user must be running a C|T shell and have the proper permissions.

Installation option #1 - Installing the EMS from DVD

If you are lucky enough to possess one of the fabulous DVDs from the WRF EMS collector series then consider yourself fortunate, because life doesn't get any better. Besides being a great coaster, the DVD can be used to install the EMS. This capability does come with a few caveats however so don't get too excited. Ok, go ahead and jump up and down a little.

In order to install the EMS from DVD you will need to actually mount the DVD, which may require root privilege on your system. For the sake of this guidance it will be assumed that you have mounted the EMS DVD under “**/media/cdrom**”, although the exact location may differ on your machine. If you’re actually reading these instructions then it’s assumed that you can figure out what to do.

Step a. Load WRF EMS DVD

Step b. Change directories to DVD drive, e.g., “**cd /media/cdrom**”

At this point you should see a copy of the install routine on the DVD. Use this version for the installation unless told otherwise by a rainbow or small domesticated farm animal with fur, not feathers. You can’t trust the feathered ones. They lie.

Step c. Run the `ems_install.pl` routine like you want something good to happen:

```
# ./ems_install.pl
```

With any luck something good *will* happen and the installation process will begin. On some systems however, you may see an error such as:

```
# ./ems_install.pl
bash: ./ems_install.pl: /usr/bin/perl: bad interpreter: Permission denied
```

Or

```
% ./ems_install.pl
ems_install.pl: Permission denied.
```

The above errors likely indicate that your DVD is mounted “**noexec**”, meaning you can’t run an executable file from the DVD.

All is not lost, besides the time required interpret the error message that is, because there is a work-around available:

Step c1. Copy `ems_install.pl` from the DVD to another location such as “**/tmp**”.

Step c2. Run `ems_install.pl` from that location with the following flags:

```
# ./ems_install.pl --dvd /media/cdrom (or wherever you mounted the DVD)
```

That should solve your EMS installation-related problems for now.

Installation option #2 - Installing from the internet

Unless you are installing the EMS directly from DVD, the default behavior of the `ems_install.pl` routine is to attempt to download the necessary files from the SOO STRC data sever. There is no need to include any special arguments or flags as everything, including the IP address of the SOO STRC server, are hardcoded into `ems_install.pl`; however, this may have to be changed in the future. Just follow the basic guidance provided in Chapter 2.3.1.

Installation option #3 - A local network installation

If you don’t have direct access to the SOO STRC server you can still install the EMS. This method requires that the EMS tar files be manually downloaded from the server and placed in a directory where they are accessible. Simply follow these few steps:

Step a. Create a temporary directory where the files are to be downloaded. This can be

called anything provided that you have at least 3Gb of space on that partition. For this example the directory will be named “**/usr1/emsfiles**”.

```
# mkdir /usr1/emsfiles
```

Step b. Open up the SOO STRC site in your browser for full releases:

<http://soostrc.comet.ucar.edu/wrfems/releases>

and determine which release you want to install, which should be the most current one if you know what’s good for you. The releases are identified by such silly names as “**3.1.1.4.99.beta**” or “**3.1.1.5.0**”, because the developer was born without an imagination.

Step c. Create a second sub directory below “**/usr1/emsfiles**” with the name of the release to be downloaded. For example:

```
# mkdir /usr1/emsfiles/3.1.1.5.0
```

Step d. Again open up the SOO STRC site to the desired release:

<http://soostrc.comet.ucar.edu/wrfems/releases/3.1.1.5.0>

And download all the tar files to your “**/usr1/emsfiles/3.1.1.5.0**” directory. There may be quite a few and some are rather large, so while you are waiting you can take up waterfall kayaking or some other worthwhile activity.

Step e. Once all the files are downloaded and your injuries have healed, run the `ems_install.pl` routine as follows (Replacing the example with the actual location of the downloaded files on your system):

```
# ./ems_install.pl --install --locdir /usr1/emsfiles/3.1.1.5.0
```

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

Step II Checking for a successful installation

Following a successful and care-free installation, you should *log out and return as the WRF EMS user*. Make sure your EMS environment is set correctly by attempting the following commands:

```
% cd $EMS
```

And you should be located at the top level of the WRF EMS.

Then try

```
% ls $EMS_STRC
```

There you will see the contents of the `wrfems/strc` directory. If both of the above tests are successful then try running the “`sysinfo`” command that is provided with your EMS (and it IS yours now.):

```
% sysinfo
```

You should see a summary of your system configuration that includes the Linux distribution and additional information such as your blood pressure (just kidding). Please make sure that the following values are correct:

Physical CPUs : 2 The number of CPUs or mice that you would see if you opened up the computer case
Cores per CPU : 4 The number of cores on each physical CPU, the stuff you can't see
Total Processors : 8 This value should be just Physical CPUs * Cores per CPU

If either the number of Physical CPUs or the Cores per CPU is incorrect, you will have to change these values in the `wrfems/EMS.cshrc` file. If everything appears to be correct then your installation is complete and you are ready to become a modeler.

Step III Running a WRF EMS benchmark simulation

The EMS package includes preconfigured ARW and NMM core domains for the purpose of testing the installation and evaluating the performance of the cores on your computer system. In addition, data are provided that allow users to benchmark their system and compare the results to other systems. Running these benchmarks is straight forward and should be a priority for a new user or anyone else who would like to taste the sweet nectar of success.

Each benchmark simulation comes preconfigured with default physics and dynamics settings for each core. If you are running these benchmarks for the purpose of comparing the performance of your system to others, then it is recommended that you not modify these settings; otherwise the comparison is useless.

Benchmark Case Background

The benchmark case is a 24-hour simulation of an extreme rainfall event over the states of Minnesota, Iowa, and Wisconsin in the north central US that occurred 18-19 August 2007. During this event precipitation amounts in excess of 250mm were reported over this region.

The areal coverage of the primary domain for both the NMM and ARW core benchmarks domains extends over the central US and consists of ~11250 horizontal grid points (NMM:75x150, ARW:106x106) with 15km grid spacing and 45 vertical levels. There is a second level nested domain that also consists of ~11250 horizontal grid points for each core with a 5km grid spacing (3:1 parent:child ratio) and 45 vertical levels. The nested domain is centered over Iowa and extends into all six neighboring states. The areal coverage for each domain is depicted in the `projection.jpg` image located in each static directory.

Do: Decide which core (ARW or NMM) you will run

The ARW and NMM benchmark cases are located in the `wrfems/util/benchmark/arw` and `wrfems/util/benchmark/nmm` directories respectively.

Do: Change to the appropriate directory used for running the benchmark you selected:

```
Or % cd <path>/wrfems/util/benchmark/arw
% cd <path>/wrfems/util/benchmark/nmm
```

Again, you should not need to make any changes to the model configuration for this exercise. However, if you wish, the default configuration for each benchmark can be viewed by using the “runinfo” command from a benchmark directory:

```
All Do: % runinfo [--domain 2]
```

Where the “**--domain 2**” is optional when running the nested benchmark.

Finally, prior to running a 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.

Running the benchmark case for each WRF core with the WRF EMS is straight forward:

- a. From the benchmark/<core> directory, run the **ems_prep**

```
% ems_prep --benchmark
```

Or if you wish to include the nested domain (*only if time and computer resources allow*):

```
% ems_prep --benchmark --domain 2
```

- b. Run **ems_run** to begin the simulation

```
% ems_run
```

Or if you wish to include the nested domain (*only if time and computer resources allow*):

```
% ems_run --domain 2
```

The amount of time required for the benchmark case to complete will depend upon the WRF core run as well as the number and speed of the processors on the system. Upon completion of the run there will be a file in the static directory called **benchmark.results** that contains your results. You may report that information to the rest of the class.

Workshop Exercise #2 - Introduction to the WRF EMS

For this exercise, you will configure and run a short model simulation for the purpose of familiarizing yourself with the WRF EMS package. Step-by-step directions will guide you through the process of setting up and running the model, and there will be a few questions along the way that will require you to investigate various parts of the system. The critical learning objective in this exercise is the process of running the model and not the interrogation of the forecast; however, you will have an opportunity to examine and share results with others if time allows.

Step I Create a computational domain

In the previous exercise you installed the EMS on a workstation and ran a benchmark simulation. You will now use the EMS to create a computational domain for a new simulation.

Do: % `cd <path>/wrfems/runs`

You should see nothing. That is because you need to create a domain using the Domain Wizard (DW) utility. This utility is provided with the EMS to assist you in creating and localizing a computational domain. This task can also be done manually but that's rather cumbersome and is less visually exciting because there will be no pretty pictures in the instructions. SO, for those reasons you will be using the DW; however, if you really want more information then Chapter 5 is your best friend.

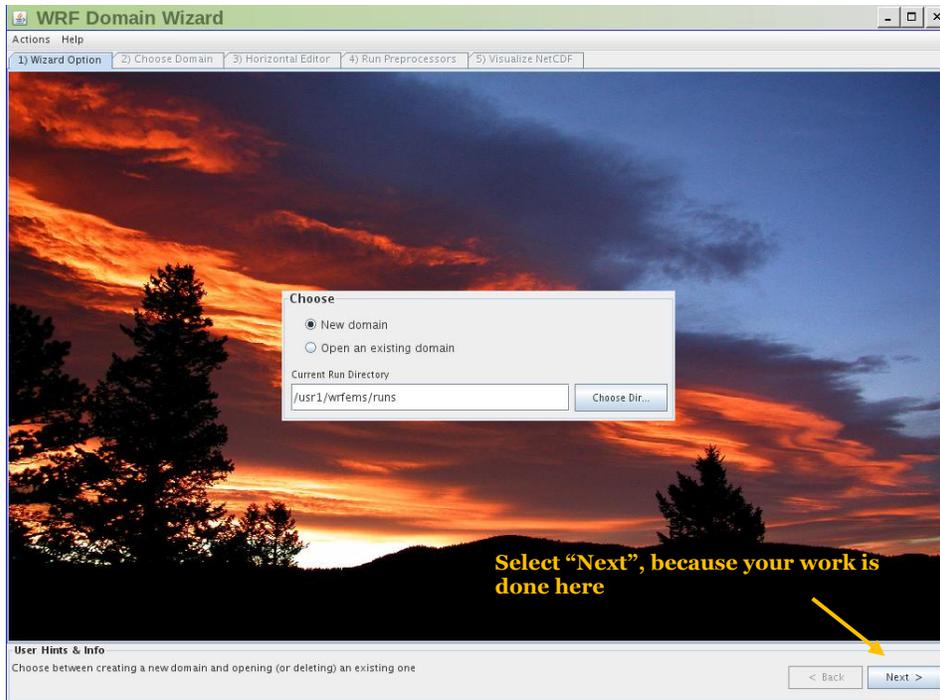
The specifications for the ARW and NMM domains are provided in appendix A. If you are running the ARW (NMM) core then use the ARW (NMM) domain configuration. For both simulations, the areal coverage and grid spacing of the model domains are the same. The only difference in the computational domains is the horizontal grid point dimensions specified for the ARW and NMM cores. This apparent conflict is due to how the NMM core defines the distance between grid points.

To start the domain Wizard:

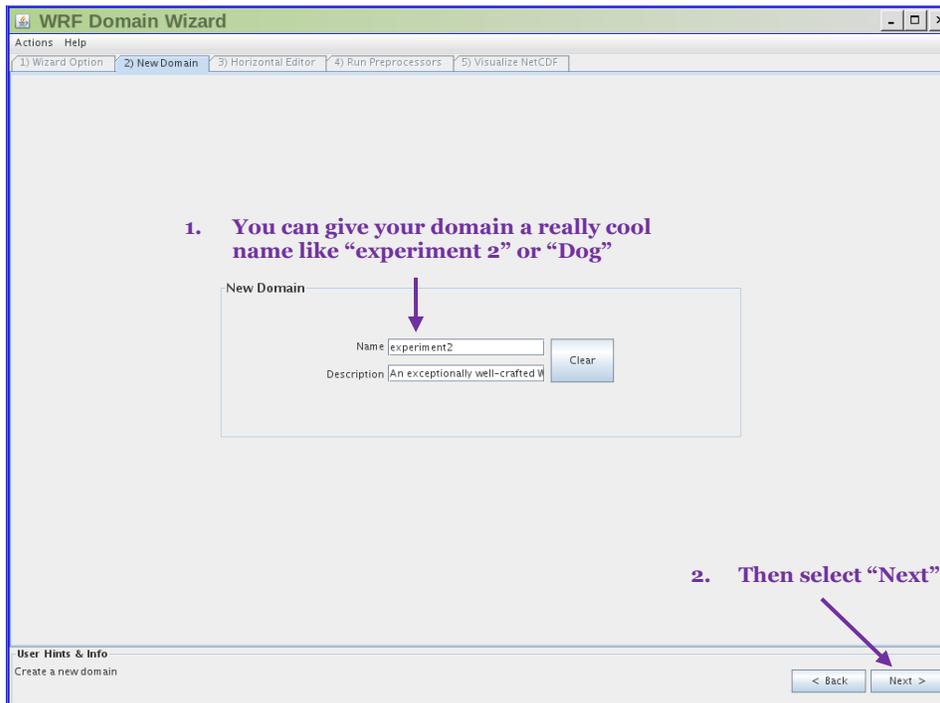
Do: % `dwiz`

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

WRF EMS Workshop Exercises – Introduction to the WRF EMS

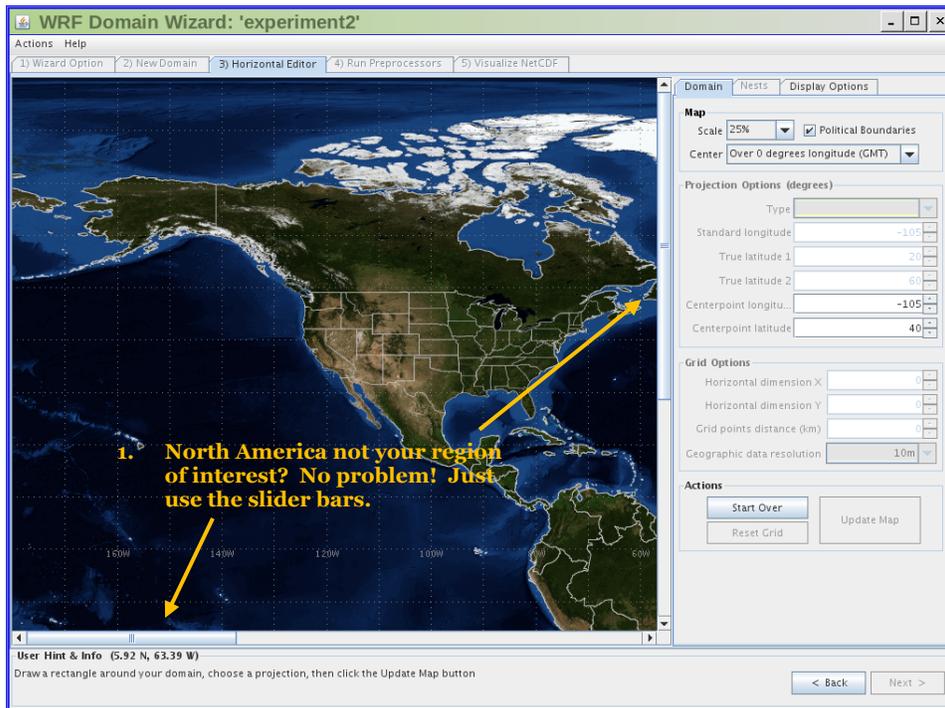


Since you are creating a new domain simply select "Next" to move on where you will see:

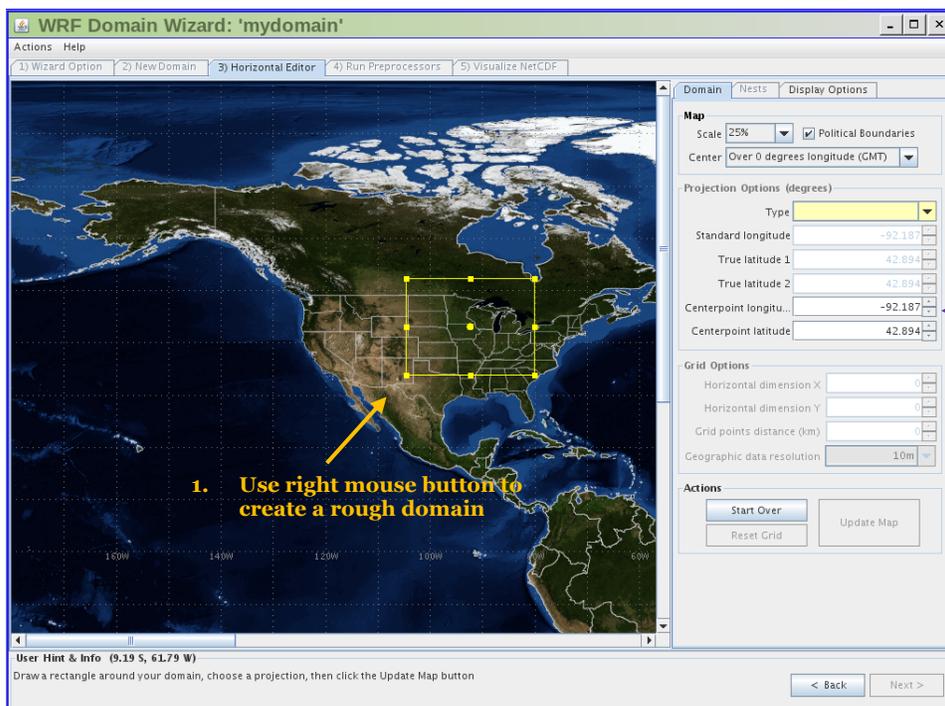


WRF EMS Workshop Exercises – Introduction to the WRF EMS

The name that you give to your domain may be whatever you like provided you can remember it. After selecting “Nest” you will see the “Horizontal Editor” window:



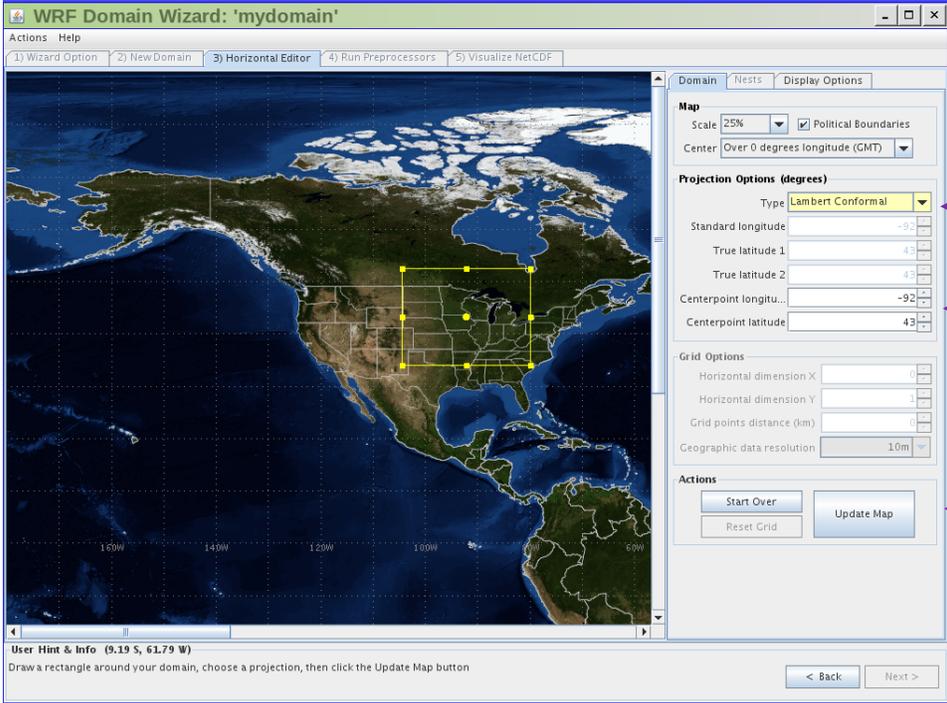
You can now use the **left mouse button** to create a rough domain, which will be fine-tuned shortly.



WRF EMS Workshop Exercises – Introduction to the WRF EMS

After creating a rough domain set the desired center point. In this experiment the center point is specified in Appendix A. After the center point is defined you can select the grid projection type from the “Type” pull-down menu. Note: Selecting “Rot Lat-Lon” automatically sets the WRF core that you are running to **NMM**; anything else, such as “Lambert Conformal” gets you the **ARW** core.

Do: **Decide whether you want to run the ARW or NMM core. Go ahead and just pick one or the other.**



The screenshot shows the 'WRF Domain Wizard: mydomain' window. The main map displays a satellite-style view of North America with a yellow rectangular domain box centered over the United States. The right-hand panel contains configuration options:

- Map:** Scale: 25%, Political Boundaries: checked, Center: Over 0 degrees longitude (GMT).
- Projection Options (degrees):** Type: Lambert Conformal (highlighted), Standard longitude: -92, True latitude 1: 43, True latitude 2: 43, Centerpoint longitude: -92, Centerpoint latitude: 43.
- Grid Options:** Horizontal dimension X: 1, Horizontal dimension Y: 1, Grid points distance (km): 1, Geographic data resolution: 10m.
- Actions:** Start Over, Update Map, Reset Grid.

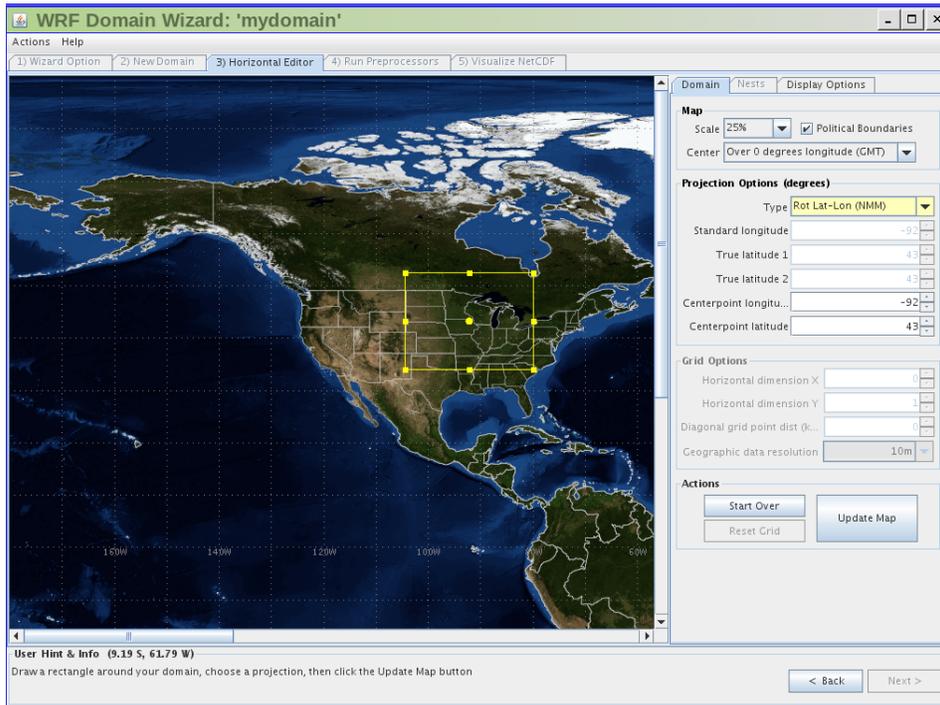
Annotations on the right side of the image:

1. Modified center point as per Appendix A (points to the Centerpoint longitude and latitude fields)
2. Choose Lambert Conformal for ARW core (points to the Projection Type dropdown)
3. Select “Update Map”, just because you can (points to the Update Map button)

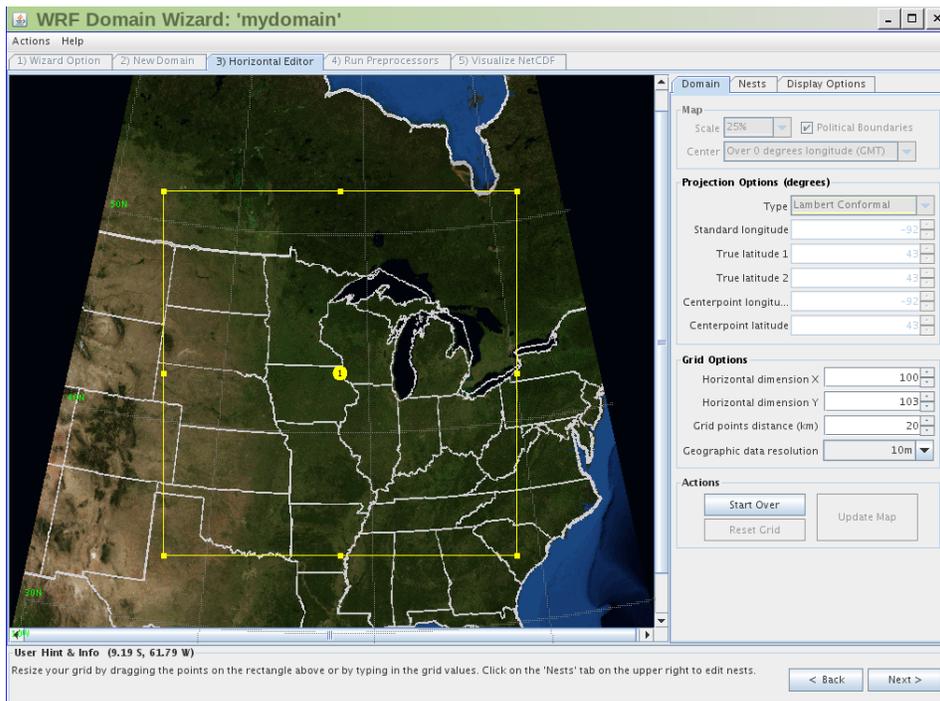
User Hint & Info (9.19 S, 61.79 W)
Draw a rectangle around your domain, choose a projection, then click the Update Map button

WRF EMS Workshop Exercises – Introduction to the WRF EMS

NMM core users should select “Rot Lat-Lon” under projection type:



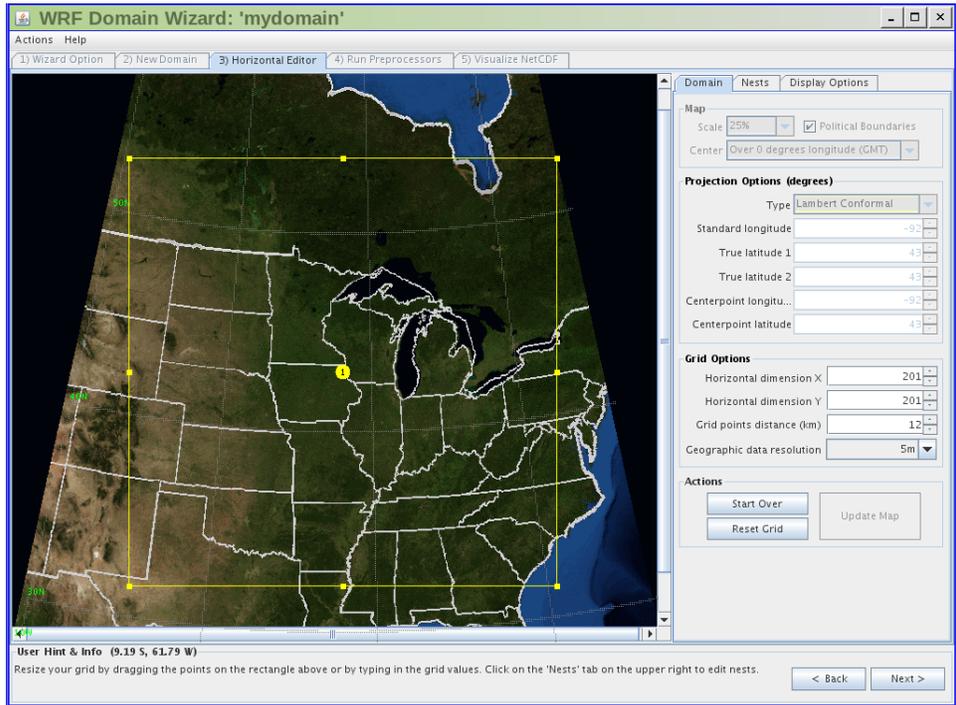
After choosing the core and selecting the “Update Map”, you will see a close-up of your domain. Notice that you can no longer control the center point location but you can modify the grid spacing and grid dimensions:



WRF EMS Workshop Exercises – Introduction to the WRF EMS

At this point you need to pay attention to whether you selected the NMM or ARW core of the WRF because the grid dimensions are different (Appendix A). Modify the horizontal dimensions and grid spacing as indicated in Appendix A. You'll be glad you did.

While making changes for the grid dimensions and spacing you will notice that the domain box on the left changes size. If you've done everything correctly the final ARW core domain should look something like:



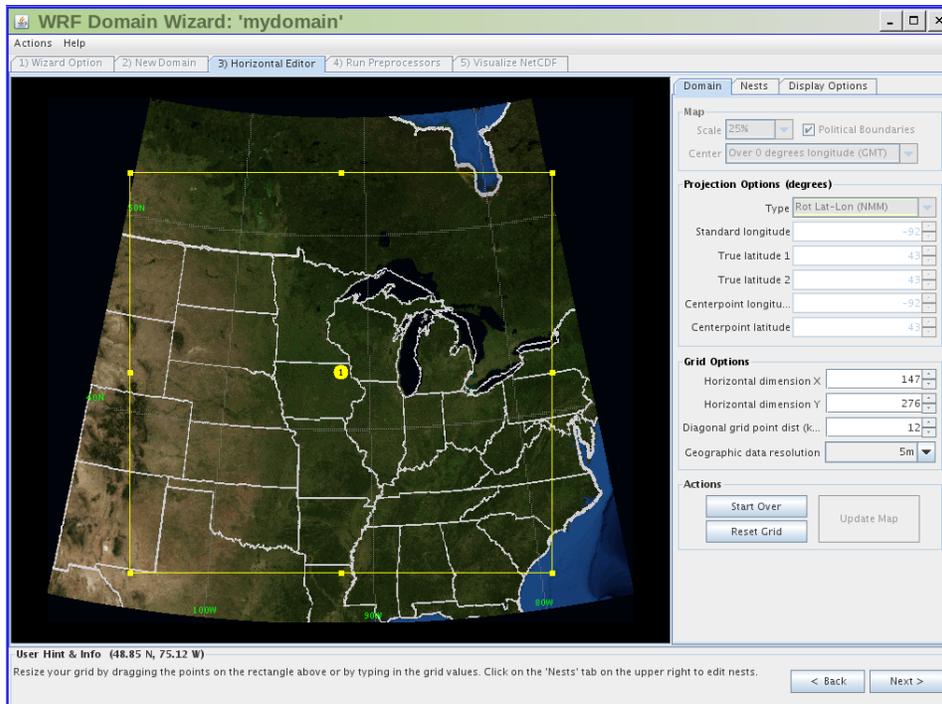
The screenshot shows the 'WRF Domain Wizard: mydomain' application. The main window displays a map of the United States with a yellow rectangular domain box overlaid. The right-hand panel contains several configuration sections:

- Map:** Scale: 25%, Political Boundaries, Center: Over 0 degrees longitude (GMT)
- Projection Options (degrees):** Type: Lambert Conformal, Standard longitude: -92, True latitude 1: 43, True latitude 2: 43, Centerpoint longitu...: -92, Centerpoint latitude: 43
- Grid Options:** Horizontal dimension X: 201, Horizontal dimension Y: 201, Grid points distance (km): 12, Geographic data resolution: 5m
- Actions:** Start Over, Update Map, Reset Grid

Two purple arrows point to the 'Grid Options' section, with the text '← Modified for the ARW core'. Another purple arrow points to the 'Next >' button at the bottom right, with the text '← Select "Next" when done, which you are'.

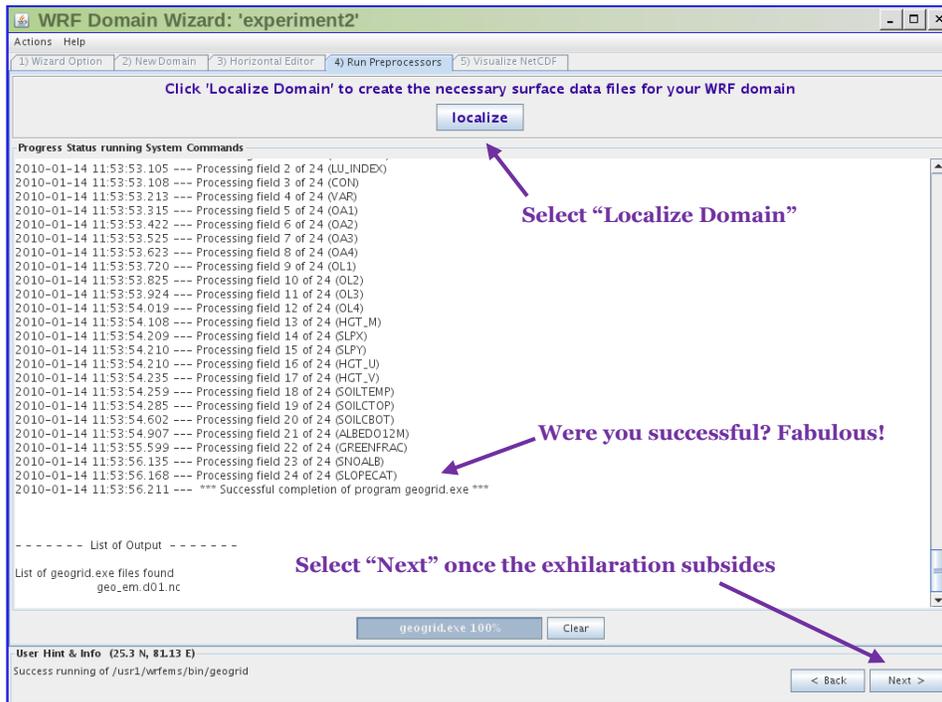
And the NMM core domain will look like:

WRF EMS Workshop Exercises – Introduction to the WRF EMS



Regardless of the core you selected, clicking “Next” will bring you to the Localization Window. When localizing a domain, you are extracting the information about your computational domain from the global static terrestrial data sets, which includes information on the land-water mask, terrain elevation, lands use, climatological albedo, etc.

Go ahead and select the “Localize” button at the top.

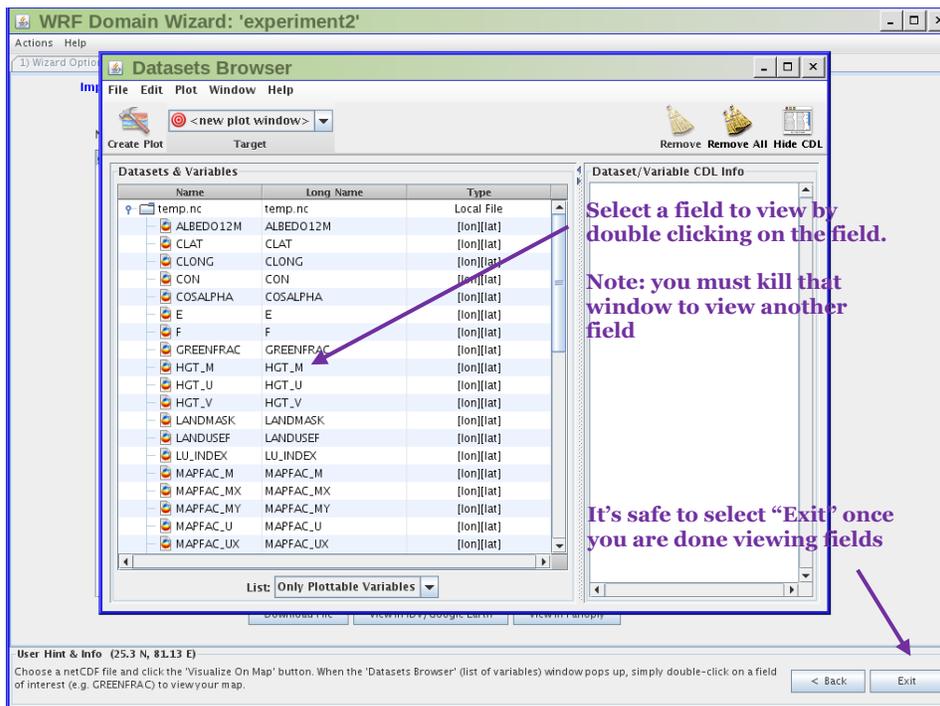


WRF EMS Workshop Exercises – Introduction to the WRF EMS

After clicking “Next” you will see the “Visualize netCDF” window. At this point you can choose to exit out of the Domain Wizard by clicking on “Exit” or going all the way by viewing the terrestrial data set you just created. We only go all the way in this exercise.

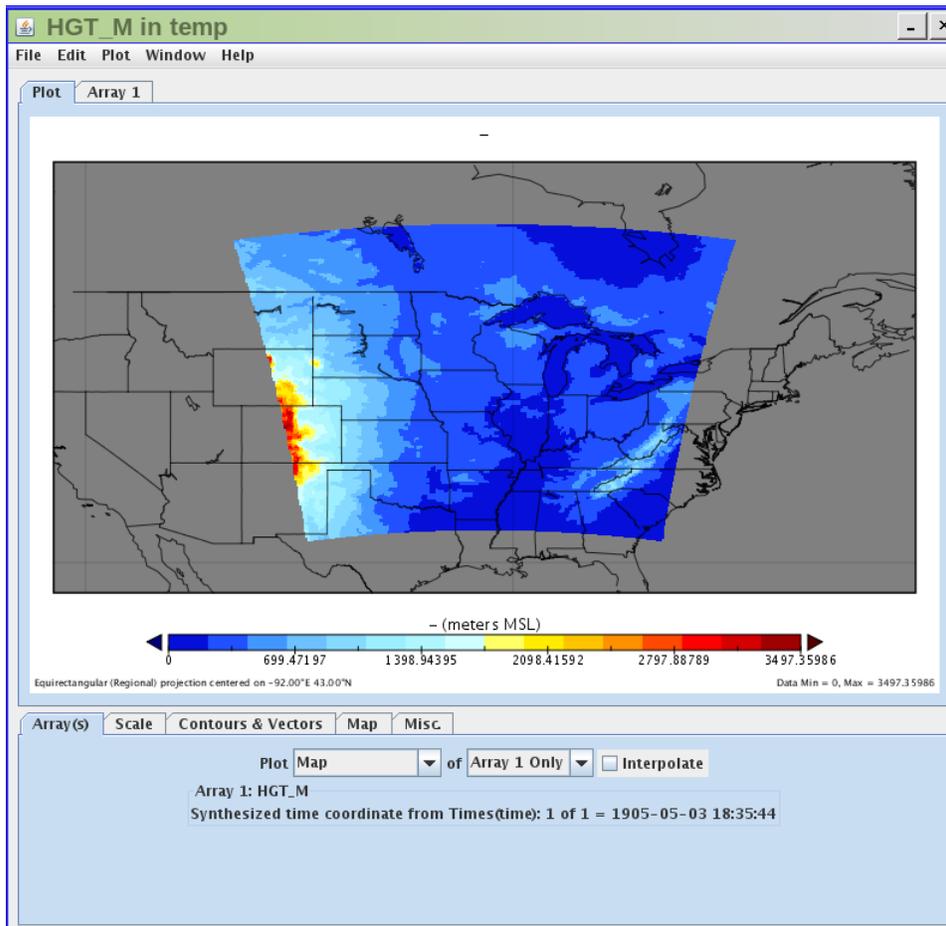


So, what is it? “Cool” or “Quitter”? Fantastic, I knew you were “cool”:



WRF EMS Workshop Exercises – Introduction to the WRF EMS

Double-click on a field to see an image of the data. You must kill that window before viewing another field. For example, the “HGT_M” (surface height at mass points) field should look like:



Simply select “Exit” when you are done viewing fields. It’s OK now.

Step II Running the WRF model

Do: Change directories to your newly-created domain

```
% cd <path>/wrfems/runs/<your domain>
```

Step IIa Process the initialization files

In your newly minted model domain directory you will find the four WRF EMS "run-time" routines, **ems_autorun**, **ems_prep**, **ems_run**, and **ems_post**, which are actually links to files in the “**wrfems/strc**” directory. These are the only routines you need to run the model successfully. If you are looking for something to do, you may review the documentation with each of these utilities by running “**ems_<routine> --help**” or “**ems_guide**”; however, there is no need to do this step for the exercise.

The first step in running the WRF EMS, besides following the instructions, is to run the **ems_prep** routine with a data set that is used to initialize your model simulation. The primary purpose of **ems_prep** is to identify, acquire, and process the external data sets for use as initial and boundary condition information. The **ems_prep** routine is the most complex of the run-time scripts as it must sort through a myriad of user options to determine which data to download, where to obtain the files, how to process the files, and then complete the horizontal interpolation to the user's computational domain. The final output files from **ems_prep** are in netCDF format and serve as the input to the WRF “real.exe” program, which is run by the **ems_run** routine.

All the necessary initialization GRIB files for this experiment already reside on your system, and thus, you will not have to access these data via ftp or http as you would when running in real-time. The WRF EMS is always thinking ahead so you don't have to.

Wait! - Before you do:

```
%ems_prep --dset labgfs:nfs --date 20070817 --cycle 18:00:36:03
```

Answer the following questions:

Question #1: Write the meaning of each flag, listed in *bold face* below, used in the previous **ems_prep** command. Hint: Look in the **labgfs_gribinfo.conf** file for clues. The **gribinfo.conf** files are located in the **wrfems/conf/grib_info** directory.

Option	Description
--dset labgfs:nfs	
--dset labgfs:nfs	
--date 20070817	
--cycle 18:00:36:03	

Question #2: How could the **--cycle 18:00:36:03** flag be rewritten? Hint#1: you can include another flag. Hint#2: Look at the **CYCLE** description in the appropriate **gribinfo.conf** file.

Question #3: Where are the data files located on your system?

Now Do:

```
% ems_prep --dset labgfs:nfs --date 20070817 --cycle 18:00:36:03
```

WRF EMS Workshop Exercises – Introduction to the WRF EMS

Running **ems_prep** should take a few minutes depending on the speed of your system. Following successful completion, you will find the processed files located in the “**wpsprd**” directory. These files are in netCDF format and the contents may be scanned with the “**rdwrfnc**” utility:

Do: % cd wpsprd

Do: % rdwrfnc <filename>

Step IIb Configure your simulation

You will actually not need to configure your run for this exercise as you will be using the default values defined in the configuration files; however, now is a good time for you to become familiar with the various subdirectories and files beneath the **conf** directory. To assist you in this task, please identify the following settings as specified in the configuration files.

Question #4: **Please identify the following settings as specified in the conf/ems_run configuration files.**

Model Physics	Scheme
Cumulus Scheme	
Microphysics Scheme	
PBL Scheme	
Land Surface Scheme	
Short Wave Radiation	
Long Wave Radiation	
Model Dynamics	Scheme
Time integration Scheme	
Model Output Information	Value
Forecast Output Frequency	
Precip Accumulation Frequency	
Default Forecast File Format	

An alternative to manually mining the information requested above is to use the “**runinfo**” utility. This utility allows users to review the configuration of a simulation in a neat tabular format.

Do (from the top level of the domain directory): % runinfo

Step IV Run the Simulation

The next step in making a simulation is to execute the **ems_run** routine. The purpose of **ems_run** is to;

- 1) Read the files output from **ems_prep**;
- 2) Run the WRF “real.exe” program to create the initial and boundary condition files for the primary and any nested domains; and
- 3) Execute either the NMM or ARW core of the model.

Before you start the model, take another look at your configuration and make sure everything is acceptable.

Do: % **runinfo**

Question #5: How many processors are being used to run the simulation?

After reviewing the configuration, start the model:

Do: % **ems_run**

The model will take some time to finish depending on the core (ARW or NMM) you chose to run and the processing performance of your system. While the model is running, you may follow along with its progress with the command specified in the window.

Question #6: How long did it take to complete the simulation?

You will not be processing your forecast files for this exercise; however, you may still view previously created images for this case by pointing a browser to:

% **firefox <path>/wrfems/util/workshop/wrfems/lab02/html/index.htm**

End of Exercise #2

Appendix A: NMM and ARW Exercise #2 Model Configurations

Initialization Dataset:

Date:	17 August 2007
Cycle run:	1800 UTC GFS Model forecast
Data set	0.5 degree Global GFS Grib 2 format
BC frequency:	3 hourly

Model Domain:

	ARW	NMM
Grid spacing:	12km	12km
Grid points (NX x NY):	201 x 201	147 x 276
Model levels:	45	45
Grid Center:	92W 43N	92W 43N
Map Projection Type	Lambert Conformal	Rotated Lat-Lon

Model Forecast Details:

Forecast length:	36 hours	36 hours
Dynamics	Non-Hydro	Non-Hydro

Workshop Exercise #3 - Sensitivity of a Simulation to Model Configuration

For this exercise, you will set up and run the WRF EMS to test the sensitivity of a forecast to different controllable parameters. A series of unique experiments is provided (Appendix A) that should be run and the results compared to a control simulation (Appendix B). Due to the time required to run, the simulations are started during an initial period and then processing and data analysis occur at a later time. The critical learning objective in this exercise is the sensitivity of model simulation to controllable parameters.

It is recommended that a single sensitivity experiment be assigned to an individual or group that will focus on any differences between their experiment and the control simulation. Each group should then present their observations to the class in a brief 5 to 10 minute presentation.

Period 1: Set up the model and run the experiment

Step I Create a computational domain

Just like in the previous exercise, you need to create your computational domain using the Domain Wizard (DW) GUI. The areal coverage and grid spacing for each model domain is the same, which is also the same as that used for the control simulation. The domain configuration information you should use is provided in Appendix A while a screen capture of the DW horizontal domain configuration window is provided in Appendix B.

Do: % `dwiz`

The name that you provide for your domain in the DW should be “Exp#” where “#” is the number of the experiment that you have been assigned.

Note that only the groups conducting experiments 5 and 6 will need to change the number of vertical levels in the model domain, just because they are special (the groups, not the experiments). All other groups should use the default number of levels (45).

When your domain is configured, go ahead and run the localization, which should take a minute or so depending upon the speed of your machine. When the localization has completed, exit out of the DW with a “next” and “exit” (it’s OK, you are still cool) and then change directories to your domain. You have done this step before.

Do: % `cd $EMS_RUN/<your domain>`

Step II Process the initialization files

All of the experiments will be using the 00 hour forecast from the 0.5 degree Global Forecast System (GFS) data set (--dset gfs) for the model initial and boundary conditions. All the initialization files have been placed on your local system defined in the gfs_gribinfo.conf file; thus, you will be using “nfs” as the file acquisition method, just as with the previous exercise.

Do: Run `ems_prep` to process the grib files:

Experiments 1, 2, 3, 4, 5, 6, 7, 8, and 13:

Do: `% ems_prep --dset labgfs --date 20070817 --cycle 18 --length 36`

The above command is instructing `ems_prep` to acquire and process the first 36 hours of the GFS 18 UTC run from 17 August 2007 for model initialization.

Experiment 9:

Do: `% ems_prep --dset labnam218%labgfs --date 20070817 --cycle 18 --length 36`

The above command is instructing `ems_prep` to use the NAM 218 data set for the initial (00 hour) conditions and the GFS data set for the lateral boundary conditions.

Experiment 10:

Do: `% ems_prep --dset labnam212%labgfs --date 20070817 --cycle 18 --length 36`

The above command is instructing `ems_prep` to use the NAM 212 data set for the initial (00 hour) conditions and the GFS data set for the lateral boundary conditions.

Experiment 11:

Do: `% ems_prep --dset labnam218 --date 20070817 --cycle 18 --length 36`

The above command is instructing `ems_prep` to use the NAM 218 data set for both the initial boundary conditions.

Experiment 12:

Do: `% ems_prep --dset labnam212 --date 20070817 --cycle 18 --length 36`

The above command is instructing `ems_prep` to use the NAM 212 data set for both the initial boundary conditions.

When `ems_prep` has completed you can move on to Step III.

Step III Configure the simulation

The configuration of each sensitivity experiment and the change from the control simulation are defined in Appendix C. Experiments 1 through 6 involve making changes to the default model physics, the settings for which are located in the `conf/ems_run/run_physics.conf` file. This is the only file that you need to edit. Experiments 7 & 8 require making changes to the `conf/ems_run/run_levels.conf` file. All other model configurations have been previously incorporated into your WRF EMS for the purpose of the instructor's sanity.

WRF EMS Workshop Exercises – Sensitivity of a Simulation to Model Configuration

Experiments 1, 2, 3, 4, 5, & 6:

Do: edit `conf/ems_run/run_physics.conf`

Experiments 7 & 8:

Do: edit `conf/ems_run/run_levels.conf`

Experiments 9, 10, 11, 12, & 13:

Do: Nothing as you made your change when running `ems_prep` or `DW`.

Step IV Run the simulation

The next step in making a simulation is to run the `ems_run` routine. The final output from `ems_run` are WRF forecast files in netCDF format, which will be processed further by `ems_post`.

But Wait! Before you start your experiment now is a good time to check the configuration just to make sure everything is correct.

All Do (Check the model configuration): `% runinfo`

All Do: `% ems_run`

All Do Not: *Do anything else with this exercise. You are done for now!*

Just use the space below for notes

Appendix B: Exercise #3 - ARW Control Run Domain & Model Configurations

Initialization Dataset:

Date:	17 August 2007
Cycle run:	1800 UTC GFS Model forecast
Data set	0.5 degree Global GFS Grib 2 format
BC frequency:	3 hourly

Model Domain:

	ARW
Grid spacing:	12km
Grid points(IM x JM):	201 x 201
Model levels:	45
Grid Center:	92W 43N
Map Projection	Lambert Conformal

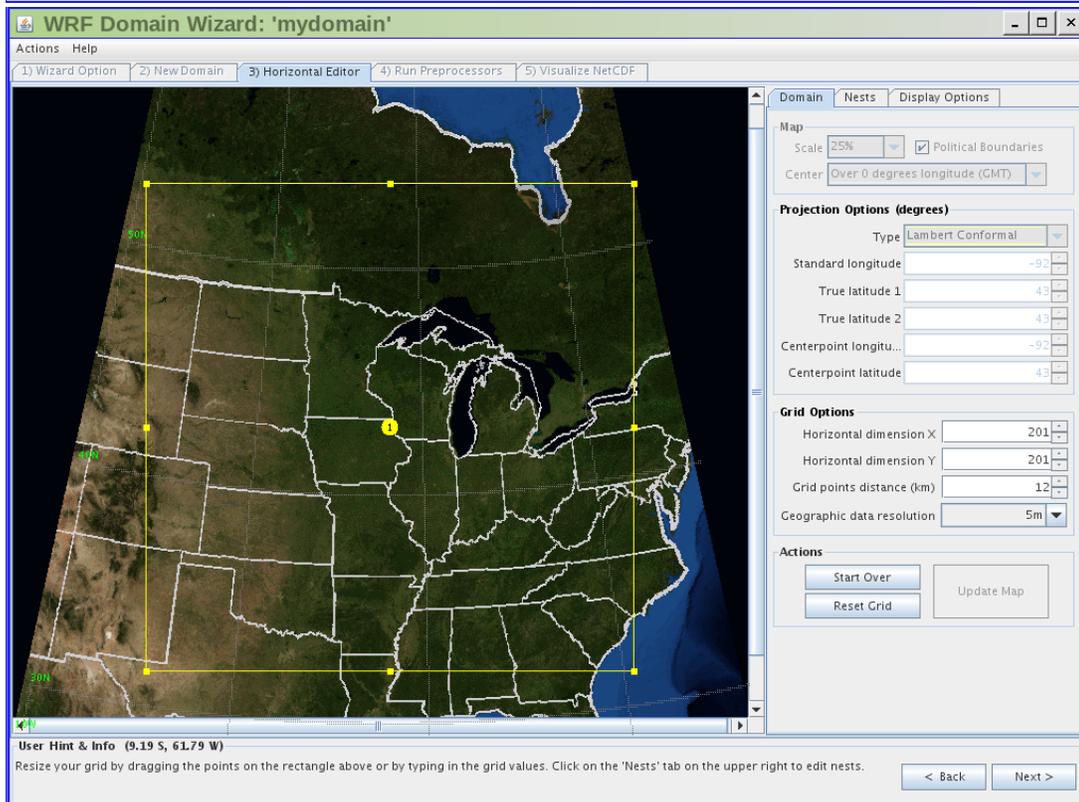
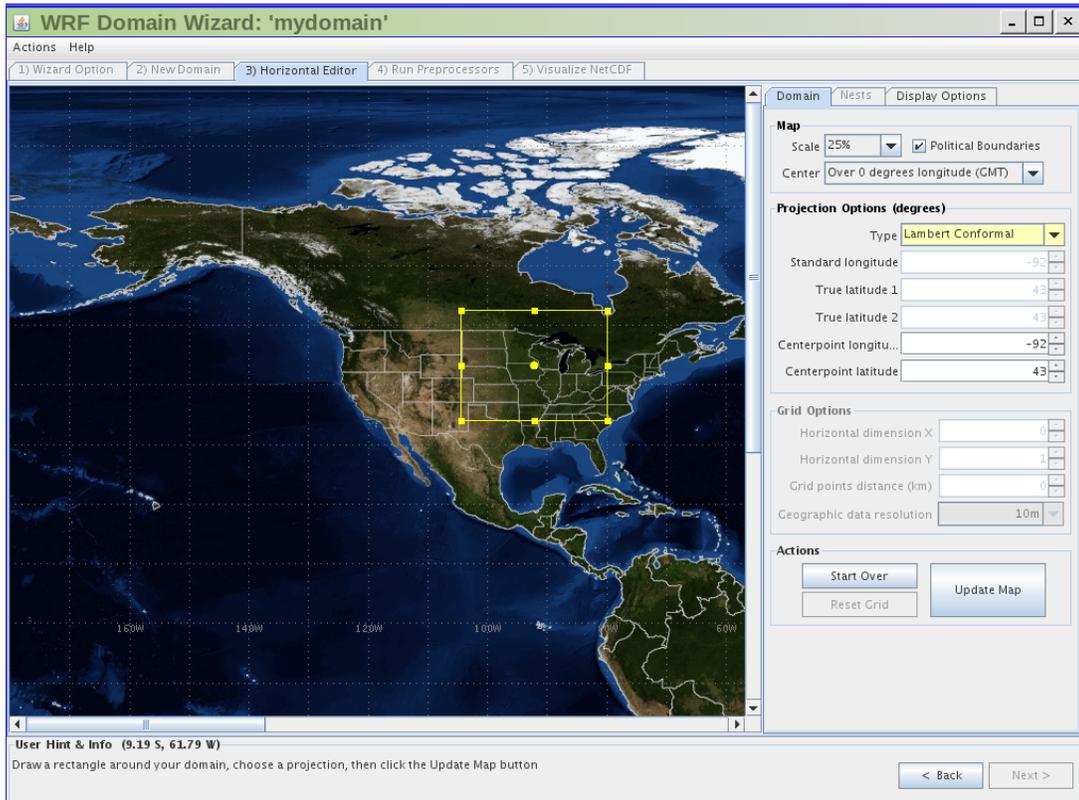
Model Forecast Details:

Forecast length:	36 hours
Model timestep:	60 seconds
Cumulus scheme	Kain-Fritsch
Microphysics	Lin
Dynamics	Non-Hydro

Model Output Information:

Output frequency:	Hourly
Precip Accum Freq:	Hourly & Storm Total
Output levels:	From 1000 to 50mb every 25mb

Appendix C: What your ARW core horizontal editor configuration windows should look like



Appendix C: Model Configurations for ARW Sensitivity Experiments

Name	Core	CU	Dynamics	MP	Radiation	Other	Compare to run(s)
Control	ARW	KF	Non-Hydro	Lin	RRTM	45 levels	
Exp 1	ARW	None	Non-Hydro	Lin	RRTM		Control
Exp 2	ARW	BMJ	Non-Hydro	Lin	RRTM		Control
Exp 3	ARW	Grell	Non-Hydro	Lin	RRTM		Control
Exp 4	ARW	KF	Non-Hydro	WSM5	RRTM		Control
Exp 5	ARW	KF	Non-Hydro	Thompson	RRTM		Control
Exp 6	ARW	KF	Non-Hydro	Lin	RRTMG		Control
Exp 7	ARW	KF	Non-Hydro	Lin	RRTM	23 levels	Control
Exp 8 (1)	ARW	KF	Non-Hydro	Lin	RRTM	91 levels	Control & #7
Exp 9	ARW	KF	Non-Hydro	Lin	RRTM	NAM 218 IC with GFS BCs	Control, #10, #11, & #12
Exp 10	ARW	KF	Non-Hydro	Lin	RRTM	NAM 212 IC with GFS BCs	Control, #09, #11, & #12
Exp 11	ARW	KF	Non-Hydro	Lin	RRTM	NAM 218 IC & BCs	Control, #09, #10, & #12
Exp 12	ARW	KF	Non-Hydro	Lin	RRTM	NAM 212 IC & BCs	Control, #09, #10, & #11
Exp 13	ARW	KF	Non-Hydro	Lin	RRTM	ARW Domain changed: 151x151 (12km)	Control

Note: Grey Shaded entries are the value that will need to be changed for each experiment

- (1) Experiment with 91 vertical levels requires 6Gb of physical memory on the system

Workshop Exercise #3 - Sensitivity of a Simulation to Model Configuration

Period 2 – Forecast processing and evaluation

The penultimate step in this exercise will be to process your forecast files into GRIB and then GrADS formats. The **ems_post** routine is used to post-process WRF EMS forecast files from the WRF NMM or ARW cores. By default, the WRF EMS forecast files are in netCDF format on native model levels. The **ems_post** routine will allow users to create GRIB formatted files containing numerous additional fields interpolated to isobaric coordinates, which can be processed further and the resultant files exported to other systems.

For this exercise, you will process the files into GrADS format. Each group will have the same set of fields available from their experiment to compare to any other simulation that they choose; however, most of you should focus on comparing your results to the suggested experiment (Appendix A). If there is another experiment that provides an interesting contrast to your simulation then feel free to discuss it as well.

Step IV Process the output data files

The **ems_post** configuration files are located in the “**conf/ems_post**” directory. Fortunately, the files have already been modified so that you don’t have to do anything except execute the following command:

All Do: % ems_post --grads

After **ems_post** has completed you will find GRIB 1 and GrADS files in the “**emsprd/grib**” and “**emsprd/grads**” directories respectively. If you are not familiar with GrADS display package there is no need to fear as images from your experiment have already been created and placed on the local system, just because the EMS is always looking out for you. These are the images that you should primarily use for the experiment inter-comparison.

While you are waiting for **ems_post** to process the data files you can view images of the experiment results by pointing your browser to:

% firefox <path>/wrfems/util/workshop/wrfems/lab03/html/index.htm

The fields available for use in your evaluations are:

- 1) Hourly 10 meter wind (m/s) and 500 to 1000 (hpa) thickness
- 2) Hourly 10 meter wind gust (m/s)
- 3) Inter-hour maximum 10 meter wind (m/s)
- 4) Hourly streamlines from 10 meter winds
- 5) Hourly accumulated precipitation (mm)
- 6) Hourly storm total accumulated precipitation (mm)
- 7) Hourly storm total accumulated grid scale precipitation (mm)
- 8) Hourly storm total accumulated cumulus precipitation (mm)
- 9) Hourly instantaneous simulated reflectivity (DbZ)

WRF EMS Workshop Exercises – Sensitivity of a Simulation to Model Configuration

- 10) Intra-hourly maximum 1 km simulated reflectivity (DbZ)
- 11) Intra-hourly maximum updraft helicity (m^2/s^2)
- 12) Hourly precipitation rate (mm/hour)
- 13) Hourly 850, 700, 500, & 300 hPa Heights (dm) and Winds (dm)

Step V Interrogate the forecast results and present findings

Each group will be allotted 5 to 10 minutes to present the results of their experiment to the rest of the workshop. You should be prepared to discuss how the configuration of your experiment differed from that of the control simulation, and whether this difference resulted in changes to the forecast. Provide an explanation for the differences if possible.

Suggestions for inter-experiment comparisons:

- a) The total amount of precipitation between experiments
- b) Location of precipitation maxima between experiments
- c) The timing of precipitation between experiments
- d) The ratios of grid scale to convective precipitation
- e) The locations and intensity of mesoscale features in the wind
- f) Impact of increased/decreased vertical resolution in the simulation
- g) Anything else that might be of interest

End of Exercise #3

WRF EMS Workshop Exercises – Sensitivity of a Simulation to Model Configuration

Appendix A: Model Configurations for ARW Sensitivity Experiments

Name	Core	CU	Dynamics	MP	Radiation	Other	Compare to run(s)
Control	ARW	KF	Non-Hydro	Lin	RRTM	45 levels	
Exp 1	ARW	None	Non-Hydro	Lin	RRTM		Control
Exp 2	ARW	BMJ	Non-Hydro	Lin	RRTM		Control
Exp 3	ARW	Grell	Non-Hydro	Lin	RRTM		Control
Exp 4	ARW	KF	Non-Hydro	WSM5	RRTM		Control
Exp 5	ARW	KF	Non-Hydro	Thompson	RRTM		Control
Exp 6	ARW	KF	Non-Hydro	Lin	RRTMG		Control
Exp 7	ARW	KF	Non-Hydro	Lin	RRTM	23 levels	Control
Exp 8 (1)	ARW	KF	Non-Hydro	Lin	RRTM	91 levels	Control & #7
Exp 9	ARW	KF	Non-Hydro	Lin	RRTM	NAM 218 IC with GFS BCs	Control, #10, #11, & #12
Exp 10	ARW	KF	Non-Hydro	Lin	RRTM	NAM 212 IC with GFS BCs	Control, #09, #11, & #12
Exp 11	ARW	KF	Non-Hydro	Lin	RRTM	NAM 218 IC & BCs	Control, #09, #10, & #12
Exp 12	ARW	KF	Non-Hydro	Lin	RRTM	NAM 212 IC & BCs	Control, #09, #10, & #11
Exp 13	ARW	KF	Non-Hydro	Lin	RRTM	ARW Domain changed: 151x151 (12km)	Control

Note: Grey Shaded entries are the value that will need to be changed for each experiment

Workshop Exercise #4 – Running a Nested Simulation

For this exercise, you will configure and run a short, nested simulation with the WRF ARW core. As with the previous EMS exercises, step-by-step directions will guide you through the process of setting up and running the nested simulation. There will also be a few questions along the way that will require you to investigate various parts of the system. The critical learning objective in this exercise is the process of running a nested simulation and post-processing the forecast data. If time allows, you will have an opportunity to examine the results but that task is not critical.

Background Information

The WRF supports synchronous 1- and 2-way nesting, which is neatly integrated into the WRF EMS. For the uninitiated, 2-way nesting allows for information from a higher resolution child domain to be fed back to the parent domain. The parent 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 provides the boundary conditions to the nested domain, there is no exchange of information from the nest back to the parent. With the EMS, all nesting is concurrent with the parent domain, meaning that the nest runs at the same time as the outer domain. The benefit of concurrent nesting is that the boundary conditions for the nested domain are provided at every parent domain time step. The limitation of this method is that it requires more, sometimes much more, physical memory on your workstation in order to run multiple domains simultaneously.

The steps in running a nested simulation are similar to running a non-nested simulation:

1. Configure a nested domain with the Domain Wizard
2. Run **ems_prep** with “**--domains**” option
3. Edit the **ems_run** configuration files for a nested simulation
4. Run **ems_run** with the “**--domains**” flag to include any nested domains
5. Run **ems_post** with the “**--domains**” flag to process the forecast files for a specific domain.

Step I Create a computational domain

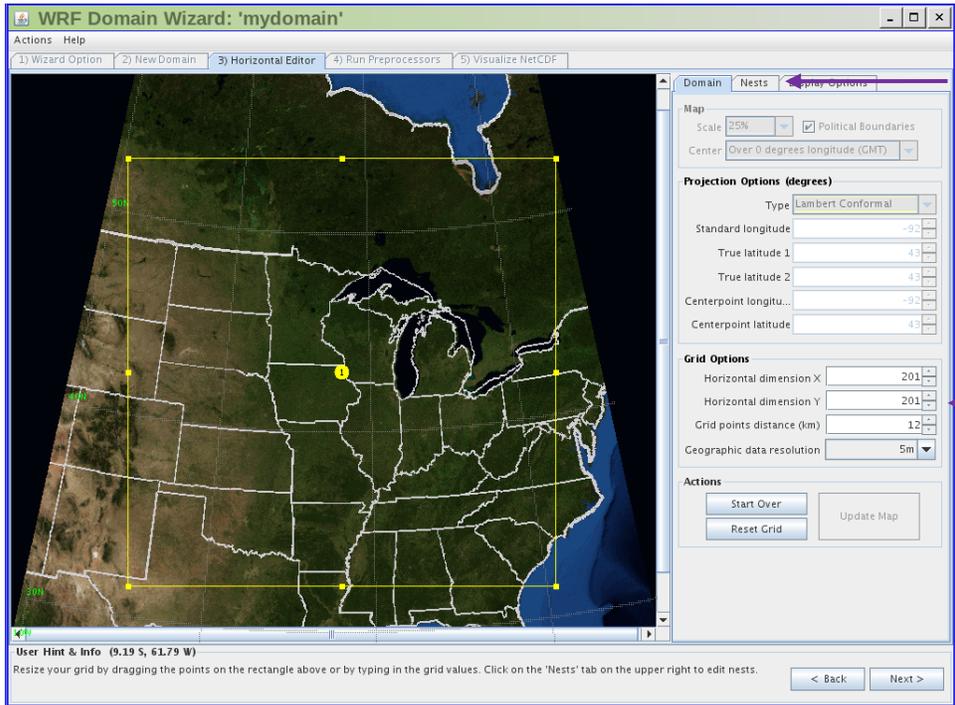
Just like in the previous exercises, you need to create your computational domain using the Domain Wizard (DW) GUI. The areal coverage and grid spacing of the primary or outer domain is the same as that used for Experiment #2 but you will be adding a sub (nested) domain. The domain configuration information for both domains is provided in Appendix A. *Pay careful attention to the configuration of the nested domain.*

Do: % **dwiz**

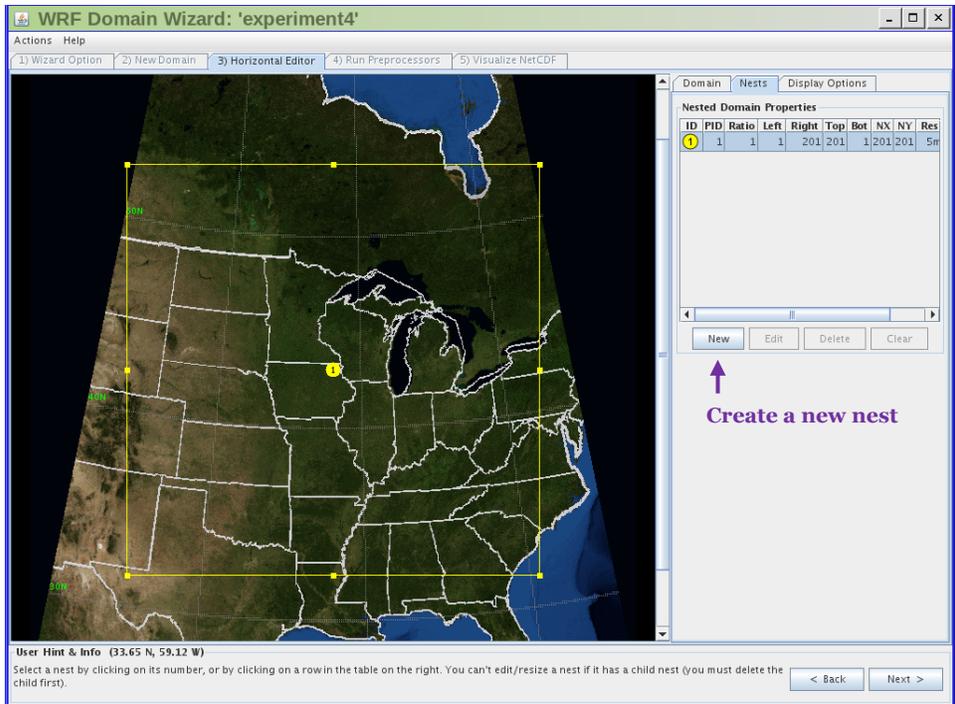
You can follow the same steps outlined in Experiment #2 in creating your computational domain. The primary domain for this experiment is the same as that used for Experiment #2. The only difference is that for this experiment you will create a nested domain for the NMM or ARW, which is where this guidance begins.

WRF EMS Workshop Exercises – Running a Nested Simulation

If you are creating a nested domain for the ARW core:

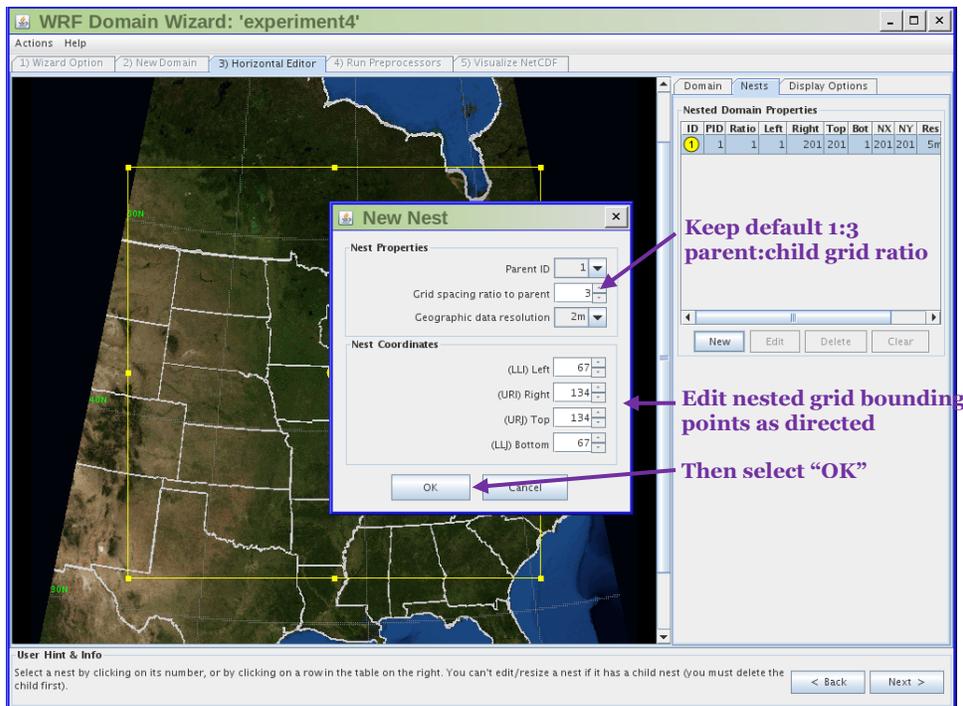


Selecting the “Nests” tab will get you the “Nested Domain Properties” menu:

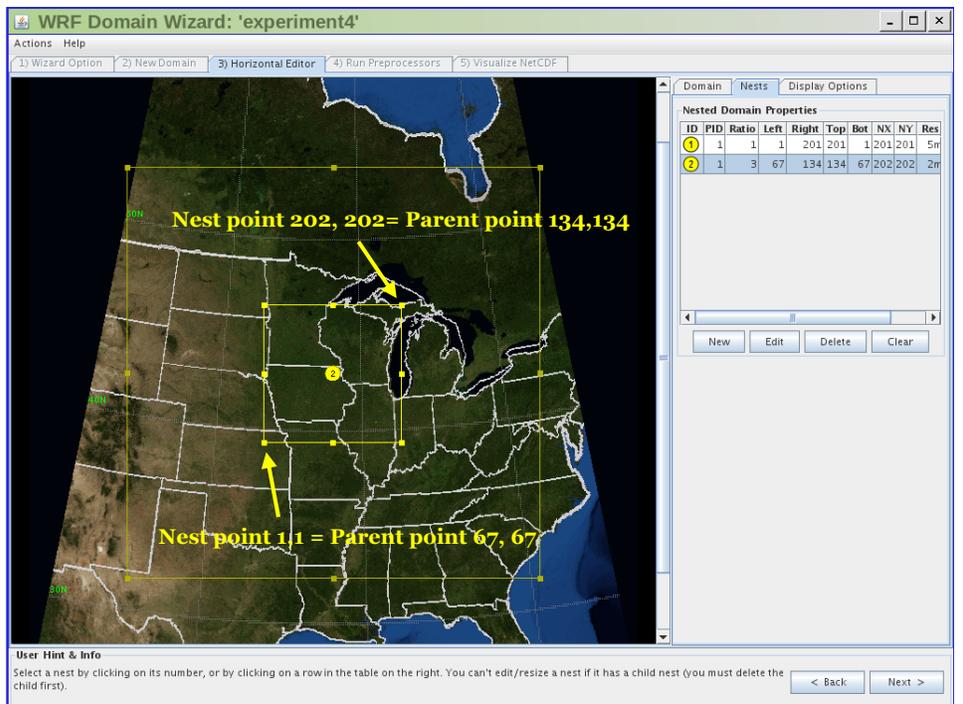


Next, select the “New” button to create a nested domain:

WRF EMS Workshop Exercises – Running a Nested Simulation

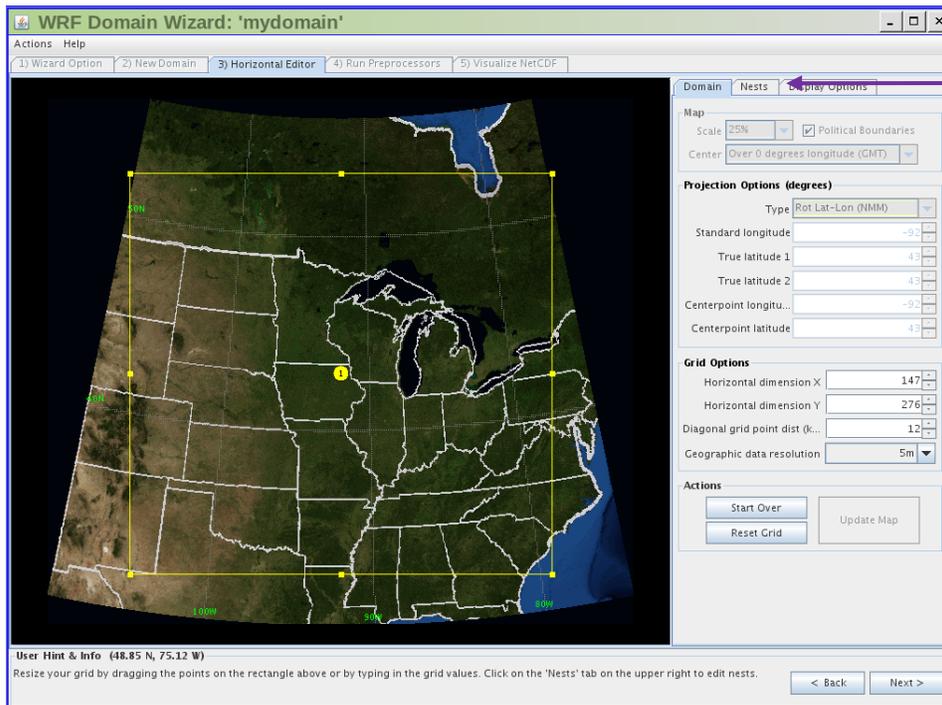


In the “Nest Coordinate” window, edit the Left, Right, Top, and Bottom to reflect the following; Left=67, Right=134, Top=134, Bottom=67 and then select “OK”:



WRF EMS Workshop Exercises – Running a Nested Simulation

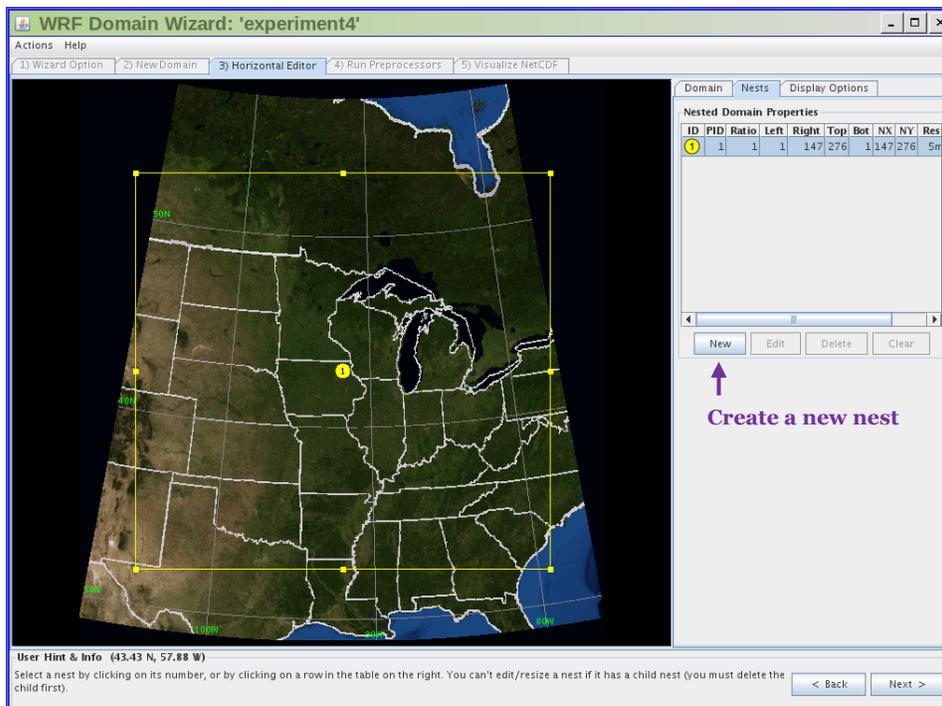
If you are creating a nested domain for the NMM core:



Hey, look here, a “Nests” tab – Go ahead and click it

Modify for the NMM core only

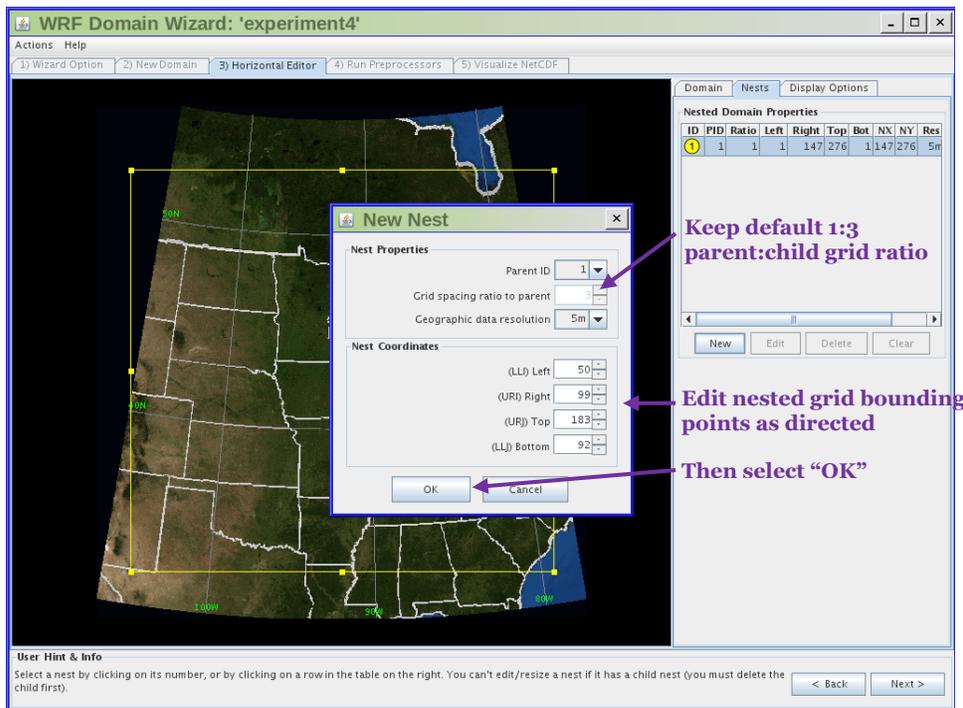
Selecting the “Nests” tab will get you the “Nested Domain Properties” menu:



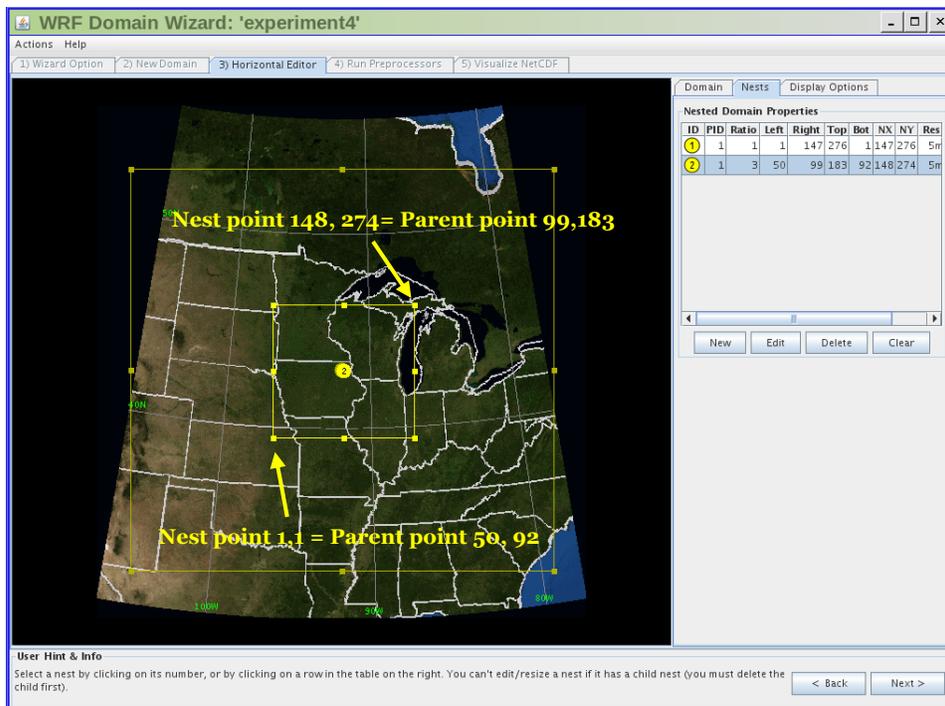
Create a new nest

Next, select the “New” button to create a nested domain:

WRF EMS Workshop Exercises – Running a Nested Simulation



In the "Nest Coordinate" window, edit the Left, Right, Top, and Bottom to reflect the following; Left=50, Right=99, Top=183, Bottom=92 and then select "OK":



When your domain is configured, go ahead and run the localization, which should take slightly longer than it did for Experiment #2 since you are including a second sub domain. When the localization has completed, exit out of the DW with a “Next” and “Exit” (Yes, you are still cool) and then change directories to your domain.

Do: % `cd $EMS_RUN/<your domain>`

Step II Process the initialization files

If this were not the highly-controlled laboratory exercise that it is, then this would be the point that you decide which, if any, of the nested domains will be used in your simulation. With the EMS, you can create multiple levels of nested domain; however, just because you create four domains doesn't mean that you must to use them all. *That's just another example of the beauty of the WRF EMS.*

For this exercise, the two domains are identified as d01 and d02. Domain 01 (d01) is always the primary or outer-most domain and does not require any special options to be passed to the run-time routines for inclusion in a simulation. If you want to use any of the nested domains however, you must use the “**--domains**” flag followed by the domain you want to initialize, and optionally, the start forecast hour, separated by a colon (:). Multiple nested domains may be included, separated by a comma (.). The “**--domains**” flag tells the **ems_prep** routine to process the initialization files for a nested run and update the configuration files as necessary.

For example (*but not necessarily for this exercise*):

```
% ems_prep --domains 2 [other arguments]
Or
% ems_prep --domains 2:06 [other arguments]
Or
% ems_prep --domains 2:06,3 [other arguments]
```

Got it? Now let's run **ems_prep** for your nested domain.

This experiment will be using the same 18 hour forecast from the 0.5 degree Global Forecast System (GFS) data set (`--dset labgfs`) as Experiment 2 for the model initial and boundary conditions.

Do: run **ems_prep** to process the grib files:

```
% ems_prep --dset labgfs --date 20070817 --cycle 18 --length 36 --domains 2
```

Question #1: **What is the meaning of the “--domains 2” flag in the above command? What if you had used “--domains 2:12”? Hint: See Chapter 7 or “ems_prep --help”.**

WRF EMS Workshop Exercises – Running a Nested Simulation

Running **ems_prep** should take a few minutes depending on the speed of your system. Following successful completion, you will find the processed files for both domains located in the “**wpsprd**” directory. These files are in netCDF format and the contents may be scanned with the “**ncview**” utility:

Do: % cd wpsprd

Do: % ncview <select a domain #2 file>

Step III Configure the nested simulation

The **conf/ems_run** directory contains the configuration files with the default settings for both the primary and nested domains. If you look inside these files you will notice some parameters that are designated with “**NESTING**”. These parameters can have different settings for each nested domain in a simulation.

Note: *You will not need to edit the configuration files for this exercise as you will be using the default values.*

The convention by which values for each of the domain is specified follows:

PARAMETER = d01, d02, d03, d04, ..., dN

Where d01, d02, d03, ... represent the value for domains 01 (Primary), 2, 3, etc.
Thus, for example:

CU_PHYSICS = 1, 3, 1, 0

Specifies the use of cumulus scheme **1** (Kain-Fritsch) for domains 1 and 3, cumulus scheme **3** (Grell-Devenyi) for domain 2, and NO cumulus scheme (**0**; explicit) for domain 4.

Note that if you do not have enough values specified i.e.

CU_PHYSICS = 1 <- Value only specified for domain 1

And you want to include domain 2 in your simulation, then **ems_run** will default to the value defined for the parent domain. In the above example, the parent of domain 2 (d02) is domain 1, so domain 2 will also use the Kain-Fritsch cumulus scheme. You may receive a warning message when you execute the **ems_run** command although it can be ignored.

Question #2: Please identify the following default settings as specified in the configuration files. Hint: use “runinfo --domain 2”

Model Physics	Parent	Nest
Cumulus scheme		
Microphysics scheme		
PBL Scheme		
Land Surface Scheme		
Short Wave Radiation		
Long Wave Radiation		
Model Output Information		
	Value	
Forecast Output Frequency		
Precip Accumulation Frequency		
Model Grid Information		
	Value	
Feedback		

Step IV Run the nested simulation

The purpose of **ems_run** is to create the initial and boundary conditions and then execute the model simulation. By default, **ems_run** will only run the primary domain unless the **--domains** flag is passed. Passing the **--domains** flag not only tells **ems_run** which nest(s) to run but also (*optionally*) the length of the forecast. Note that while the **--domains** flag in **ems_run** is similar to that used with **ems_prep**, the arguments are slightly different. *With **ems_run**, the value specified after the colon (:)* represents the length of a nested domain simulation, whereas in **ems_prep**, the value represented when (number of hours) to **start** the nested simulation following the start of the primary domain.

The arguments to the “**--domains**” flag in **ems_run** are:

```
% ems_run --domains domain#[[:Length<s|m|h|d>],...
```

Where,

- Domain#** The nested domain you wish to include
- Length** The length of the nested simulation. The units <s|m|h|d> must be included with the length option

Note that:

1. “Domain#” is mandatory but “Length” is optional.

2. The Length value is preceded by a colon (:) and must include the time units.
3. Multiple Domains are separated with a comma.
4. In the absence of a length value, the nested simulation will default to the stop date/time of the parent domain

Do: Start the nested simulation

```
% ems_run --domains 2
```

**Question #3: What does the “--domains 2” flag mean?
What if you had used “--domains 2:12”?**

At this point the model will take some time to finish depending on the number and configuration of the nested domains. While the model is running, you may follow along with its progress by executing the command specified in the window.

Step V Process the nested simulation output data files

The purpose of the **ems_post** routine is to process output data files generated from a WRF NMM or ARW core simulation. By default, these files are in netCDF format on native model levels, which may not meet the ever-demanding needs of EMS users. Running the **ems_post** routine allows users to further process these data into a variety of formats and to export the files to other systems.

The **ems_post** processing options include, but are not limited to, converting the WRF netCDF to GRIB 1 or 2 format, writing to GEMPAK and GrADS files, and creating BUFR, GEMPAK, and BUFKIT sounding files. Users also have the option to export files to remote systems via secure copy (SCP), file transfer protocol (FTP) secure file transfer protocol (SFTP), or a simple copy command (CP).

Following a successful nested simulation, forecast files for each domain will be located in the **wrfprd** directory with each domain identified by a “_do#” in the filename. You can process each domain individually by passing the **--domain #** option to **ems_post**. For example:

```
% ems_post --domain 2 [other options]
```

Passing “**--domain 2**” as in the above example will process only those files for (nested) domain 2. In the absence of the “**--domain**” option, **ems_post** will default to domain 1. If you have multiple nests to process you must run **ems_post** for each domain.

You can also turn ON/OFF the post proceeding options for individual domains in the **ems_post.conf** file. Similar to the **ems_run** configuration convention, in the absence of an explicit specification for a parameter, **ems_post** will default to the value of the parent domain.

For example:

```
GRADS = Primary Domain, Nest d02, Nest d03, Nest d04, ..., Nest dN
```

Or

WRF EMS Workshop Exercises – Running a Nested Simulation

GRADS = Yes, No, Yes, No

The above example specifies that you want to convert forecast files from the primary and domain 3 to GrADS format, but not nested domains 2 and 4. Remember that you need to tell **ems_post** which domain to process with the **--domain** flag. Passing any command line flag to **ems_post** overrides the corresponding setting in a configuration file. So, passing "**--domain 2 --grads**" will turn ON the GrADS processing for nested domain d02.

Do: % **ems_post --domain 2 --grads**

While you are waiting for wrfpost to process the nested domain you can view images of both the parent and nested domain by pointing your browser to:

For the Primary & Nested ARW and NMM domains:

% **firefox <path>/wrfems/util/workshop/wrfems/lab04/html/index.htm**

Just use the space below for notes

Appendix A: Exercise #4 - Nested Run Model Configurations

Initialization Dataset:

Date:	17 August 2007
Cycle run:	1800 UTC GFS Model forecast
Data set	0.5 degree Global GFS Grib 2 format
BC frequency:	3 hourly

ARW Model Domain:

	Primary	Nest
Grid spacing:	12km	4km
Grid points (IM x JM):	201 x 201	202x202
Parent I, J Start	1, 1	67, 67
Parent I, J Stop	201, 201	134, 134
Model levels:	45	45
Grid Center:	92W 43N	
Map Projection Type	Lambert Conformal	Lambert Conformal

NMM Model Domain:

	Primary	Nest
Grid spacing:	12km	4km
Grid points (IM x JM):	147 x 276	148x274
Parent I, J Start	1, 1	50, 92
Parent I, J Stop	147 x 276	99, 183
Model levels:	45	45
Grid Center:	92W 43N	
Map Projection Type	Rotated Lat-Lon	Rotated Lat-Lon

Model Forecast Details:

Forecast length:	36 hours	36 hours
Dynamics	Non-Hydro	Non-Hydro
Output Frequency	1-Hour	30 Minute

Workshop Exercise #5 - Real-time NWP made (almost) effortless

For this exercise, you will set up and run the WRF EMS for the purpose of making a real-time forecast. You will be using the `ems_autorun` routine initiated from a cron that will (hopefully) execute each of the steps from downloading the initialization files, processing the data, running the model, and post processing the forecasts. You will have (almost) complete control over the configuration of your computational domain and forecast run. The only objective is to have a 24 hour forecast completed and available within 4 hours from the start of the process. That is your mission, which you must embrace or your “coolness” will wear off. The run will be initiated during the first period and analysis done during a later period. Each student/group will then present their forecasts to the class in a brief 10-minute presentation.

Period 1: Set up, test & test & test, and then initiate the forecast

Important! - Testing to make sure your real-time run will be successful is critical to this exercise. Therefore, you will be required make two short test forecasts before committing your real-time attempt to cron.

Step I. Create a computational domain and read this guidance carefully

Just like in the previous exercises, you need to create your computational domain using the Domain Wizard (DW) utility. Unlike the other exercises however, you will have complete control over the configuration of your domain. It is important that you not get too ambitious. Remember that the goal of this exercise is to complete a 24 hour forecast within two hours of actual time. **The two hour window includes** the time required to download and process the initialization files, run the model, and process the forecast files. Everything in two hours! **It does not include** the time to read these directions, set up and configure your simulation or run the tests.

Question #1: How much time do you have from start to finish of your real-time simulation? Just checking if you were paying attention.

The 4 hour window for completion of the model run is an attempt to replicate the real-time forecasting environment, in which the model forecast must be available to the users within a set amount of time. It may be longer than 4 hours in the real world but your time in the classroom is also limited. So, for this exercise, it is quasi-arbitrarily set at 4 hours.

To help you meet this two hour time limit, here a few suggestions to consider when setting you your domain:

1. You will be using the 1 degree GFS “**personal tiles**” data set for model initialization. The size of the individual files in this data set is smaller than the full “**gfs**” data set you have been using for the previous experiments, yet retain the full spatial and temporal resolution. While the files reside on a remote server in the US, the bandwidth required to download these data should not be an issue.
2. You may choose to run the ARW or NMM core for this experiment. Recall which core was faster in the benchmark case? This may factor into your decision.

- Remember that doubling the number of vertical levels results in a 2-fold increase in computation time, while doubling the horizontal grid spacing (over the same computational domain) results in an 8-fold increase in computation time.
- Do not** set up a nested domain for this experiment. Start with a simple configuration to improve your chances of success. You don't have enough time for a nested run anyway.
- Do not compromise areal coverage for resolution! The area coverage should be the first thing you consider when setting up a computational domain. Afterwards, determine the grid spacing that you can afford to use over your forecast period (24 hours).
- After doing the initial tests, you find that you cannot make a 24 hour forecast within the required 4 hour window, you should reconfigure to your computational, i.e., increase the grid spacing.
- If you run the model with a grid spacing less than 4km you should use the ARW core. But chances are that you will not approach that resolution for this exercise.
- Your choice of model physics will **not** have a large impact on time required to run the model, although you should still select schemes that are appropriate for the resolution.
- Hourly output of forecast files is better than 3-hourly output, but it takes 3 times longer to post process.

10. Again, the `ems_autorun` routine must finish within two hours!

Ok, back to the creation of your computational domain:

Do: % `dwiz`

The name that you provide for your domain can be whatever you like provided you can remember it. When your domain is configured, go ahead and run the localization, which should take a few minutes. When the localization has completed, exit out of the DW with a "Next" and "Exit".

Do: % `cd <path>/wrfems/runs/<your domain>`

Step II. Test the download and processing of the initialization files

As stated earlier, you will be using the personal tiles (`--dset gfsptile`) for the real-time forecast. Consequently, you will have to access the US National Weather Service (NWS), Science and Operational Officer (SOO), Science and Training Resources Center (STRC) data server. So the first task is to check the amount of time required to download and process the initialization data.

Do: Run `ems_prep` to test the download and processing of the grib files

% `ems_prep --dset gfsptile --length 24`

Question #2: Should you ever use `--date` or `--cycle` flags for real-time modeling?

Note the amount of time required to download and process the files. You can expect that it will take approximately the same time during your real-time run.

Question #3: How much time did it take to complete `ems_prep`?

After `ems_prep` has successfully completed you can move on to configuring and testing the `ems_run` routine.

Step III. Configure your real-time model forecast

While you may configure the model for your real-time forecast any way you desire, just leave the default values if you have any reservations about a particular setting. If, during the testing, you encounter a problem then you should consider reverting back to the default values. *Remember, the default values are your friends.*

Do: Determine if any changes are needed to the following files:

`conf/ems_run/run_physics.conf`
`conf/ems_run/run_ncpus.conf`
`conf/ems_run/run_wrfout.conf`

Question #4: Identify the following settings for your real-time run.

Model Dynamics	Scheme
Model Core (NMM or ARW)	
Model Physics	Scheme
Cumulus scheme	
Microphysics scheme	
PBL Scheme	
Land Surface Scheme	
Model Output Information	Value
Forecast Output Frequency	

Step IV. Make a short test run of your real-time forecast

You will now make a brief test run of your real-time forecast over a 3-hour period. You will be able to get an approximation of the total time required to run a 24 hour forecast by simply multiplying the 3-hour forecast time by 8. Note that your actual real-time forecast will require slightly less time to complete as there is a small amount of overhead at the beginning of a forecast.

Do: % `ems_run --length 3h` (Note the use of the “h” in “--length 3h”)

Question #5: How many minutes did it take to complete `ems_run`?

Amount of minutes x 8 =

When `ems_run` has completed you can move on to configuring and testing `ems_post`.

Step V. Test the processing of the forecast files

The `ems_post` routine will be used to process the output forecast files. The `ems_post` routine along with additional programs relies on various configuration files and tables to complete the task of processing your WRF forecast data. These files provide the default `ems_post` options and may be edited to meet the needs of the user. Additionally, many of these options can be overridden by command line arguments to `ems_post`.

The `ems_post.conf` file

In the absence of any command-line options, `ems_post` will use the `ems_post.conf` file in determining how to process the WRF EMS forecast files. Current options include processing forecast files into GRIB 1 & 2, GEMPAK, GrADS, BUFR, and BUKIT formats. There are also options for exporting files to other systems. Again, most of the parameters contained in the `ems_post.conf` file may be overridden by command line options. Please see “`ems_post --help`” or **Chapter 9** for more information on the command line options.

The other configuration files

In addition to the `ems_post.conf` file, there are other configuration files in the `conf/ems_post` directory that are used to further your post-processing experience. These files should be reviewed and edited as necessary. The files are:

<code>post_export.conf</code>	- File export options
<code>post_bufr.conf</code>	- Converting forecast files to BUFR format
<code>post_grads.conf</code>	- Options for processing GrADS files
<code>post_gempak.conf</code>	- Options for processing GEMPAK files
<code>post_grib.conf</code>	- Options for processing GRIB 1 and 2 files

For this exercise you will only need to edit the `ems_post.conf` and `post_grads.conf` files for the purpose of turning ON and configuring the generation of GrADS files and images.

Do: Edit `ems_post.conf` and set “GRADS = Yes”

Do: Review the `post_grads.conf` and uncomment (remove the “#”) in front of the following line:

```
# POSTSCR = <path>/wrfems/util/grads/products/kumar/plot_grads.tesh
```

Do: % `ems_post`

Comments:

- The **ems_post** routine will read the **ems_post.conf** file and see the user's request for GrADS file processing.
- It will read the **post_grads.conf** file for any additional directions.
- GRIB file processing will automatically be turned on since GRIB files are needed for the creation of GrADS files.

Question #6: How many minutes did it take to complete ems_post?

Amount of minutes x 8 =

Question #7: Time to run ems_prep _____ **minutes**
Time to run ems_run x 8 _____ **minutes**
Time to run ems_post x 8 _____ **minutes**

Total time for simulation (Approx) _____ **minutes**

Question #8: Is the total time calculated in #7 significantly greater than 120 minutes?

If the answer to # 8 is "Yes", then go back to Step I and reconfigure your domain. **DO NOT make it smaller!** Simply increase the grid spacing.

Step VI Configure and test the ems_autorun routine

The **ems_autorun** routine is intended for use in automating the process of running a simulation and processing the output files. It is designed to read a user-controlled configuration file and then execute **ems_prep**, **ems_run**, and **ems_post** in succession. The routine is ideally suited for use in real-time forecast applications; however, it may be used to run case studies as well. For real-time forecasting, there are various options to improve the reliability of your forecasts. Additionally, there is an option for processing the output data files concurrent with model run.

As with the other WRF EMS run-time scripts, **ems_autorun** includes a help menu and users guide. The help menu provides a brief summary of all the options available to **ems_autorun**. To view the brief help menu:

Do: **% ems_autorun --help**

The ems_autorun configuration file

There are a number of controllable parameters that are used to drive the **ems_autorun** routine, all of which are described ad nauseam in the **ems_autorun.conf** configuration file. When you create a new domain, either with the Domain Wizard or manually, a copy of the default configuration is placed in the **conf/ems_autorun** directory. The configurable parameters in this file are exceptionally well-documented, so prior to your initial **ems_autorun** attempt; you should take the time and review the contents of this file. You will be glad you did.

Question #9: Identify the following settings for your real-time run from the `ems_autorun.conf` file.

Parameter	
Data set	
Forecast hour to use for initial condition	
Boundary Condition Frequency	
Forecast Length	

At this point almost everything is configured to run your 24 hour real-time simulation, but before the job is placed in a cron you should run an additional test using the `ems_autorun` script.

When you reviewed the help menu (“--help”) you should have noticed the `--length` flag, which is used to override the default value for the length of the run (24 hours) as (should be) defined in the `ems_autorun.conf` file. Nearly all the command line flags in `ems_autorun` are used to override the default values defined in the `ems_autorun.conf` file. For this test of the `autorun` routine you are reducing the length of the run to 3 hours to expedite the process. If it runs for three hours it should run for 24 hours.

Do: `% ems_autorun --length 3`

Question #10: How many minutes did it take to complete `ems_autorun`?

Amount of minutes x 8 =

If the total amount of time required to run `ems_autorun` is well in excess of four hours, you will have to modify your domain configuration to reduce the time.

Step VII Initiating the `ems_autorun` from cron

The final step in preparing your real-time run for “prime time” is to create a crontab entry that will be initiated at a specified time. Fortunately, much of the work has already been completed for you during the WRF EMS installation process. First, a tesh wrapper file, `ems_autorun-wrapper.csh`, was placed in the `wrfems/strc/ems_bin` directory to facilitate the running of `ems_autorun` from a cron. In addition, a deactivated entry was placed in your crontab. Using the `ems_autorun-wrapper.csh` file will make sure that your environment variables are set correctly prior to running `ems_autorun`.

To review the entries in your crontab file:

Do: `% crontab -l`

You should see an entry for `ems_autorun-wrapper.csh` that looks something like:

```
#31 7 * * * /<path>/wrfems/strc/ems_bin/ems_autorun-wrapper.csh --rundir  
/<path>/wrfems/runs/<your domain> >& /<path>/wrfems/logs/ems_autorun.log 2>&1
```

Although it is probably on a single line.

The entry should be inactive meaning that there is a “#” symbol at the beginning of the line.

The **ems_autorun-wrapper.csh** file is simply a Tshell wrapper that is used to initialize the WRF EMS environment variables before running **ems_autorun**. *Not executing **ems_autorun** this way via cron will result in the model failing to run.*

In order to configure the **ems_autorun** crontab entry for real-time modeling:

1. Remove the leading “#”
2. Replace the “<your domain>” with the name you gave to your real-time domain, i.e., the directory under wrfems/runs.
3. Set the time that you wish the run to start.

This requires some knowledge of how to edit the time information in a crontab file. If the machine is using UTC then also specify the UTC time in the crontab file. If the machine is using local time then do likewise in the crontab file. Also, the initial column is reserved for the start minute; the second entry is start hour. In the above example the start time is 0731 UTC or local time.

You will be making one more test before leaving for the evening, just to make sure everything is working as expected.

Do: Deactivate and configure the crontab entry and set the start time for 5 minutes from the current time

Also Do: Add “--length 3” to the entry for this test, i.e.

```
16 10 * * * /<path>/wrfems/strc/ems_bin/ems_autorun-wrapper.csh --length 3 ...
```

Notice: You must remove the “--length 3” entry before starting the actual real-time run!!

After the time that the simulation should have started check the progress by tailing the wrfems/logs/ems_autorun.log. If the file does not exist then the cron failed to activate. If there were other problems they should be specified in the file.

A comment on simulation start times

You want to start a real-time simulation as soon as the initialization files are available on the server. Most of the initialization files are identified by the cycle time of the originating model from which they are derived. For example, the GFS runs four times per day with cycle times of 00, 06, 12, and 18 UTC. The forecast files for these runs are available some time after the cycle time, depending upon the time required to run the model and post process the forecast files into GRIB format. For the case of the GFS this “delay” is typically about four hours after a cycle time. Thus, the 12 UTC forecast files for the GFS will not be available until approximately 1600 UTC. This

WRF EMS Workshop Exercises – Real-time NWP made (almost) effortless

processing delay is build into the **ems_prep** routine through the DELAY parameter in the <data set>_gribinfo.conf files. The **ems_prep** routine knows when files should be available on the server and you should take this fact into account when specifying the time to start your model run. If you plan on using the 1800 UTC cycle of the GFS to start your run then set the start time in the crontab files sometime after 2200 UTC. Again, this time should be specified in local time if your system uses local time.

Do: Remove the “--length 3” entry from the crontab file and specify the start time for your real-time forecast to be 5 minutes from the current time.

Congratulations you are done for now

Workshop Exercise #5 - Real-time NWP made (almost) effortless

Period 2: Data interrogation and presentations

If all went well then your real-time forecast files should be located in the emsprd directory beneath your domain directory. You will see two subdirectories, grib and grads, which contain the GRIB 1 and GrADS formatted forecast files respectively. During this period you will review your forecasts and prepare a 10 minute presentation on your real-time model configuration and the results of your forecast. The presentations should include the following:

- 1) **Domain configuration:** Core used, location, navigation, areal coverage of your domain, number of grid points, grid spacing
- 2) **Model configuration:** Model physics, Output frequency
- 3) **Total time required to run the forecast**
- 4) **Time required for downloading the initialization files**
- 5) **Time required for processing of the initialization files**
- 6) **Time required to run the model**
- 7) **Time required for post processing**
- 8) **Finally, your forecast:** Any fields you feel are of interest

End of Exercise #5

