ARL-TR-8533 ● SEP 2018

US Army Research Laboratory

The Weather Running Estimate – Nowcast Realtime (WREN_RT) System, Version 1.03

by Brian P Reen and Leelinda P Dawson

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-8533 ● Sep 2018

US Army Research Laboratory

The Weather Running Estimate – Nowcast Realtime (WREN_RT) System, Version 1.03

by Brian P Reen and Leelinda P. Dawson
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
September 2018	Technical Report	February 2016 – September 2018

4. TITLE AND SUBTITLE
The Weather Running Estimate – Nowcast Realtime (WREN_RT) System, Version 1.03

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Brian P Reen and Leelinda P Dawson

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
US Army Research Laboratory
ATTN: RDRL-CIE-M
2800 Powder Mill Road
Adelphi, MD 20783-1138

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-8533

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The Weather Running Estimate – Nowcast Realtime System (WREN_RT) is a scripted system that runs the Advanced Research version of the Weather Research and Forecasting (WRF-ARW) model. It performs all the data gathering and processing to prepare the input data for Real and WRF itself as well as running Real and WRF. WREN_RT is designed to allow both beginner and advanced users to execute WRF more efficiently. WREN_RT retrieves and processes coarse-grid model data to provide boundary and initial conditions along with downloading and using alternative sea surface temperature and snow databases. Additionally, WREN_RT downloads, quality controls, and applies observations to improve the initial conditions via an objective analysis and observation nudging.

15. SUBJECT TERMS
WRF, Weather Research and Forecasting, numerical weather prediction, WRE-N, Weather Running Estimate-Nowcast, WREN_RT, mesoscale modeling

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Brian P Reen
Unclassified	Unclassified	Unclassified	UU	74	19b. TELEPHONE NUMBER (Include area code) 301-394-3072

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

Acknowledgments

The Real-Time Mesoscale Analysis use/reject lists provided by Steve Levine at the National Centers for Environmental Prediction's Environmental Modeling Center greatly facilitated making full use of the Meteorological Assimilation Data Ingest System observational data set.

1. Introduction

The Weather Running Estimate – Nowcast Realtime (WREN_RT) is software that facilitates the application of the Advanced Research version of the Weather Research and Forecasting (WRF-ARW; Skamarock et al. 2008) model by providing a scripted system that retrieves input data, runs preprocessors, and executes WRF-ARW (hereafter WRF-ARW is referred to simply as WRF for brevity). In addition, it includes the ability to apply observation nudging data assimilation. Obtaining the input data needed to run WRF, preprocessing these data for use in WRF, and properly configuring WRF can be a complicated, error-prone, and time-consuming process, particularly when multiple sources of input data are being used and observation nudging is used. Thus, a properly designed scripted system to carry out these tasks can simplify the production of WRF weather forecasts, reduce the chances of errors occurring in the process, and allow weather forecasts to be more rapidly generated. By incorporating best practices, it also increases the chances that these best practices are implemented by modelers. For example, if a user suspects that assimilating observations would improve the model forecast, the user is more likely to assimilate the observations if the process is simple and quick via a scripted system than if the process is complicated and time consuming. The software is designed with the goal of assisting users with both limited and advanced WRF modeling experience. This is accomplished by defaulting to reasonable settings where possible, which removes the requirement that users specify these variables, but includes the ability for them to easily modify these settings based on their unique requirements. Therefore, WREN_RT lowers the barriers (e.g., time, complication, and skill) to creating a quality WRF weather forecast. By automating the process, WREN_RT also allows for the unattended creation of a WRF weather forecast.

The RUNWPSPLUS software package formed the starting point for WREN_RT. RUNWPSPLUS is software developed by the US Army Research Laboratory (ARL) that prepares input data for WRF simulations but does not have the capability to obtain coarse-grid model (CGM) data (for use as initial and lateral boundary conditions) and cannot run WRF itself. RUNWPSPLUS runs the WRF Preprocessing System (WPS) software in addition to other software used to process data for input into WRF. RUNWPSPLUS was incorporated into the WRF end-to-end (WRFEE) software (Kirby et al. 2013) and has been used independently of WRFEE in various ARL studies.

WREN_RT consists of scripts that run WRF and orchestrate the application of various software in order to prepare input data for it. Many of the components are

software provided by the National Center for Atmospheric Research (NCAR) as part of or to support WRF. In some cases, these components have been modified to provide the best functionality in WREN_RT.

Since WREN_RT is continually evolving, this report describes a snapshot of its current capabilities. The version of WREN_RT described in this report is version 1.03.

2. WREN_RT Description

The basic steps undertaken by WREN_RT are the following:

- Read main user configuration file.
- Download CGM data.
- Obtain other input data and prepare them for WRF (carried out by RUNWPSPLUS component).
- Run Real.
- Execute WRF.
- Archive WREN_RT output data.

The main scripting in WREN_RT is written in Python 3, while RUNWPSPLUS is written in Perl. Many of the programs called by the scripts are written in Fortran.

2.1 User Configuration

For most users, the main WREN_RT configuration file should be the only configuration that will need to be modified to use WREN_RT. This configuration file consists of three sections labeled WREN_RT, RUNWPSPLUS, and WRF. The [WREN_RT] section is the main section that the user will need to modify and is where users set details such as the start time of the model and the size, number, and locations of the model grids. These settings are used by WREN_RT; some of the settings are used to determine settings in the configuration files used by RUNWPSPLUS and WRF, while others are used in other portions of WREN_RT. For running RUNWPSPLUS and WRF, WREN_RT starts with default configuration files and then alters them based on the user settings in the [WREN_RT] section of the main configuration file. If the user wishes to customize how RUNWPSPLUS or WRF are configured beyond the configuration that results from the user settings in the [WREN_RT] section of the main configuration file, the user can directly specify settings in the relevant section of the main

configuration file. For example, if the user wants to directly specify a WRF setting, then in the [WRF] section of the main WREN_RT configuration, the user can specify this setting.

This process is illustrated in Fig. 1 where the main configuration file *wren.config* is used to update the RUNWPSPLUS template configuration file *runwpsplus.config_wren_template* into *runwpsplus.config* for use in RUNWPSPLUS and the WRF template configuration file *namelist.input_wren_template* into *namelist.input* for use in Real and WRF. The settings in Fig. 1 (*setting1*, *setting2*, etc.) are not actual settings, but are used here for purpose of illustration.

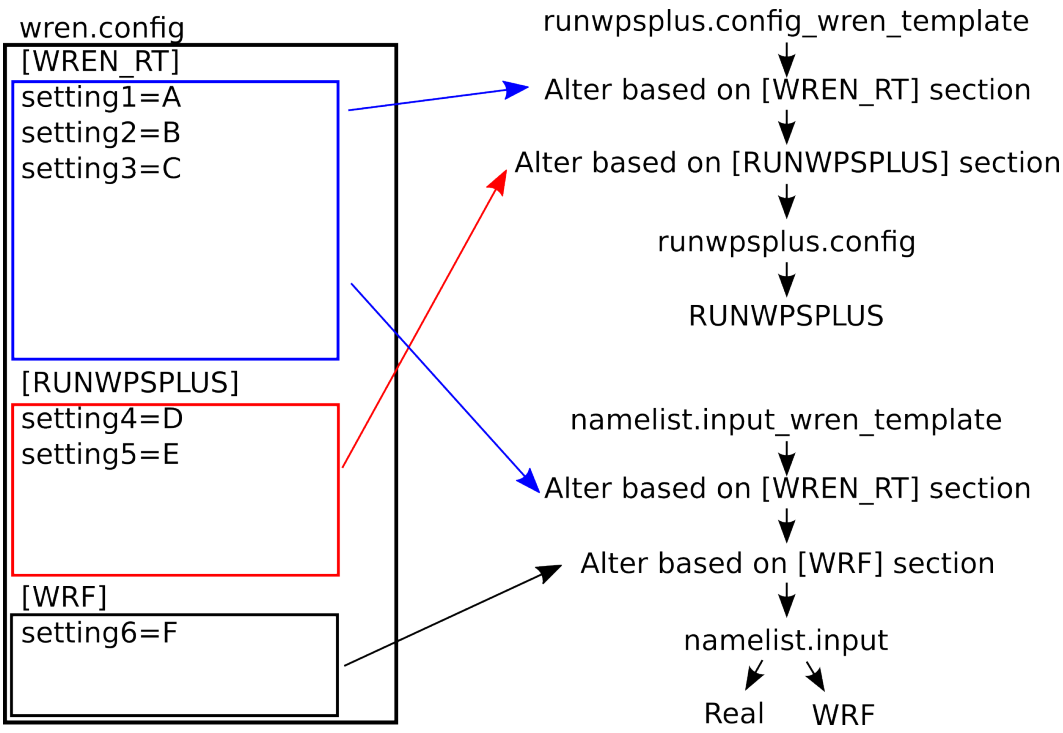

Fig. 1 Diagram of use of the WREN_RT user configuration file to modify the RUNWPSPLUS and WRF configuration files. The settings shown are not actual settings in WREN_RT, but are created for this report for purpose of illustration.

In Fig. 1, in order to build the configuration file used by RUNWPSPLUS (*runwpsplus.config*), WREN_RT starts with the RUNWPSPLUS template configuration file (*runwpsplus.config_wren_template*). Some of the setting values read in from this file are replaced based on values of settings in the [WREN_RT] section of the WREN_RT configuration file (*wren.config*), such as *setting1*, *setting2*, and *setting3*. For brevity, subsequently in this discussion "the [X] section" is used to mean "the [X] section of the WREN_RT configuration file (*wren.config*)". While some settings in the [WREN_RT] section may directly

specify a setting in the RUNWPSPLUS configuration file (i.e., *setting1=A* means that a setting in the RUNWPSPLUS configuration file is also set to *A*), others are used to derive settings in the RUNWPSPLUS configuration file (i.e., *setting2=B* and *setting3=C* together are used to determine that a setting in the RUNWPSPLUS configuration is set to *X*). At this point in the processing of the settings, some of the RUNWPSPLUS settings WREN_RT would use are based on the RUNWPSPLUS template configuration file whereas others are based on settings in the [WREN_RT] section. Any settings in the [RUNWPSPLUS] section are used to override the configuration created based on the RUNWPSPLUS template configuration file and the settings in the [WREN_RT] section. This means that since *setting4=D* in the [RUNWPSPLUS] section, RUNWPSPLUS will be configured with *setting4=D* regardless of the contents of the RUNWPSPLUS template configuration file and regardless of the settings in the [WREN_RT] section. The configuration file actually used by RUNWPSPLUS is created by this process and is named *runwpsplus.config*.

The process used to determine WRF settings is analogous. The first-guess WRF settings are based on the WRF template configuration file (*namelist.input_wren_template*). Then, the settings in the [WREN_RT] section (e.g., *setting1*, *setting2*, and *setting3*) are used to update some of the WRF settings. Finally, any settings in the [WRF] section (e.g., *setting6=F*) are used to override whatever is currently set. The end result of this file is the *namelist.input* file used by Real and WRF.

The [RUNWPSPLUS] and [WRF] sections of the WREN_RT configuration file allow the user to directly specify any RUNWPSPLUS and WRF setting (respectively) as long as it is present in the template file. All RUNWPSPLUS settings should be present in the RUNWPSPLUS template file (*runwpsplus.config_wren_template*). However, all WRF settings are not present in the WRF template file (*namelist.input_wren_template*), since there are a multitude of possible WRF settings. Thus, it is possible that the user will want to specify a WRF setting not in the WRF template configuration file. In that case, the user will need to modify the WRF template configuration file to include the setting the user is attempting to set. This will then allow a value to be specified for this setting in the WREN_RT configuration file in the [WRF] section. WREN_RT cannot automatically add a variable to the WRF configuration file that is not in the template version, since WREN_RT does not know which section of the WRF configuration file a given setting should be placed.

More details about the settings in the WREN_RT configuration file can be found in the Appendix.

2.2 Input Data

WREN_RT relies on both static and case-specific input data. Once WREN_RT is installed, it assumes that the static data are available locally, while it will obtain the case-specific input data as needed.

2.2.1 Static Input Data

The default static input data are distributed by NCAR for use with WRF and includes fields such as terrain height, soil type, and land use type. Some of the sources are available at various resolutions, and WREN_RT generally will choose the coarsest resolution that is finer than the resolution of the model domain (or the finest-resolution data set if none are finer than the resolution of the model domain).

For land use, WREN_RT allows the user to choose between two standard data sets and two nonstandard data sets. The standard data sets have worldwide coverage and are distributed by NCAR specifically for use in WRF, whereas the nonstandard data sets only cover a portion of the Earth and are not distributed with the WRF static geography data. The standard data sets are the US Geological Survey (USGS) land use and the International Geosphere-Biosphere Programme Moderate Resolution Imaging Spectroradiometer (IGBP MODIS) land use. The USGS data set was the default in versions of WPS immediately preceding WPSV3.8 and is available at various resolutions with the highest resolution being 30 arc seconds (≈925 m at the equator). WREN_RT chooses the coarsest-resolution data set that has finer grid spacing than the model domain to which the data set will be applied. The MODIS data set is the default in WPSV3.8.x and WPSV3.9.x and is only available at 30-arc second resolution.

The first of the nonstandard land use databases is based on the National Land Cover Database 2006 (NLCD2006) but translated to USGS categories and including the additional categories of playa, white sands, and lava. This data set has been created by NCAR (but it is not distributed on the WRF website) and is available only over the contiguous United States. A significant advantage of this data set is that it is provided at 1-arc second grid spacing. If this data set is chosen but does not cover part of the user's domain or the data set is not present in the user's installation of WREN_RT, the USGS data set will be used where NLCD2006 is not available. The 1-arc second database can provide much better classification of land versus water points in sub-1-km simulations. One disadvantage of this database is that the very high resolution can result in very slow application of the data to chosen domains via the WPS component Geogrid, particularly for relatively coarse domains. If repeated simulations are planned over the same domain, it is recommended that after running Geogrid for one simulation, the Geogrid output be reused (via settings

in the [RUNWPSPLUS] section of the configuration file, namely, *use_user_geo_em_files=1*, and *USER_GEO_EM_DIR="/location/of/geo_em_files"*). If the user chooses to use this special NCLD2006 land use data set, an altered version of the standard WRF soil database will be used, one that includes the three new land use types (white sands, lava, and playa). The version of WRF in WREN_RT has been altered to include these new land use types in the lookup tables.

The second nonstandard land use database that can be used by WREN_RT is the approximately 4-arc second Coordinate Information on the Environment (CORINE) data set. This data set covers parts of southwestern Europe, including Portugal, most of Spain, and parts of France. This land use data set is mapped to the USGS categories. If this land use is chosen but does not cover part of the user's domain or the data set is not present in the user's installation of WREN_RT, the USGS data set will be used where CORINE is not available.

For terrain height, WREN_RT generally uses the standard WRFV3.8.x–WRFV3.9.x database, which is a 30-arc second Global Multi-resolution Terrain Elevation 2010 (GMTED2010) database. However, if the domain horizontal grid spacing is finer than 1 km, then the 3-arc second Shuttle Radar Topography Mission (SRTM) data set will be used where available (it appears to cover approximately 60°S–60°N).

2.2.2 Case-Specific Input Data

WREN_RT requires data from a CGM to provide lateral boundary conditions and initial boundary conditions, and attempts to enhance the simulation using additional case-specific input data.

Data from three CGMs can be downloaded and processed by WREN_RT and a fourth CGM can be processed if the user manually downloads the files (the CGM is chosen via *coarse_grid_model*). The three CGMs that can be downloaded by WREN_RT are provided by the US National Oceanic and Atmospheric Administration's (NOAA) National Centers for Environmental Prediction (NCEP) and are the Global Forecast System (GFS) model, the North American Model (NAM), and the High-Resolution Rapid Refresh (HRRR) model. WREN_RT obtains the data from the NOAA real-time servers for recent cases and from archival servers for older cases where available. WREN_RT can obtain 1.00°, 0.50°, and 0.25° GFS data (chosen via *coarse_grid_model_resolution*) for the most recent approximately 9 days from the real-time server. For older cases, WREN_RT can obtain 1.00° data for the most recent 12 months and 0.50° data for the most recent 23 months. For 12-km NAM data, the real-time server is used to obtain the newest approximately 6 days, while the archive server can provide the newest

11 months. For 1-km HRRR data only, the current day and previous day are available for download. Note that the length of data available on the real-time and archive servers varies with time (i.e., the periods of time over which data are available for automated download vary, thus the user should expect that the time periods of availability described may no longer valid). For some CGM data, older data are available via manual download and WREN_RT can be told to look for CGM data in a local directory (via settings *already_downloaded_cgm_directory* and *use_all_cgm_data_in_directory*). Note that while NAM and HRRR provide higher-resolution CGM data than GFS, only GFS is global in coverage.

WREN_RT can also process model data from the US Air Force Global Air-Land Weather Exploitation Model (GALWEM) for initialization. While WREN_RT can use GALWEM data, it cannot obtain the data and relies on the user or some other process to have already obtained the data. Users who have the need to use GALWEM data should consult with the WREN_RT developers to ensure that the GALWEM fields necessary to properly use GALWEM are obtained. To use GALWEM data, both WPS and WRF software were altered. One issue contributing to the need to modify these software was that GALWEM did not provide a volumetric soil moisture but instead provided a vertically integrated soil moisture.

For data assimilation, WREN_RT obtains observations from NCEP's Meteorological Assimilation Data Ingest System (MADIS; madis.noaa.gov). By default, WREN_RT will download surface, profiler, radiosonde, and aircraft observations. Specifically, the MADIS surface observation data sets it will download include mesonet, Météorologique Aviation Régulière (METAR), maritime, and surface airways observations (SAOs) (this can be overridden by adding settings in the [RUNWPSPLUS] section of the WREN_RT configuration file of the form *use_madis_**). The aircraft observations are referred to Aircraft Communications Addressing and Reporting System (ACARS). Optionally, the user can also enable use of satellite-derived wind (by adding *use_madis_satwind=1* and *use_madis_satwind1h=1* in the [RUNWPSPLUS] section of the WREN_RT configuration file). The user can also provide the location of a file containing observations in little-r format as an additional source of observations (via settings *include_user_littler* and *USER_LITTLE_DIR_FNAME*). WREN_RT automatically determines whether to retrieve MADIS data from the MADIS retrospective or real-time directory structure based on how far the case is in the past. WREN_RT defaults to public access to MADIS, which does not require the user to have a MADIS account, but can also support research and government-level access (*madis_access_type*). These levels of access require an account but may provide access to additional observations.

Sea surface temperature (SST) can be obtained from the CGM data and used to initialize the SST field in WREN_RT. However, due to the potentially coarse spatial resolution of the CGM data (e.g., 0.50° GFS data) and the potential importance of SST heterogeneity on the weather forecast, the use of a higher-resolution SST field is recommended. WREN_RT obtains and uses SST from the 1/12° horizontal grid spacing Real-Time Global Sea Surface Temperature (RTG SST) product (Gemmill et al. 2007) from NCEP. If RTG SST cannot be obtained, then the CGM data are used to define SST.

The CGM also provides fields that can be used to specify the snow depth for the WRF initial conditions, but these fields can be quite coarse compared to the resolution of the WRF domains. For example, consider a 1-km simulation using 0.50° GFS data. At 45° latitude, the GFS snow field will be specified on cells approximately 111 × 79 km, and thus one GFS value for snow will be applied to approximately 8,769 1-km grid cells. There can be substantial heterogeneity in snow cover, especially in areas with significant terrain variability, but also in other areas including regions with significant lake-effect snow. The presence of snow can have significant impacts on the near-surface conditions, and thus it is important to provide the model with a better-resolved snow field than that available from the 0.50° GFS. WREN_RT uses 30-arc-second resolution snow fields from NOAA's National Operational Hydrologic Remote Sensing Center (NOHRSC) Snow Data Assimilation System (SNODAS; NOHRSC 2004) where available (parts of North America). Where NOHRSC SNODAS data are not available, CGM snow fields are used but are adjusted via National Snow and Ice Data Center (NSIDC) daily 4-km snow cover fields from the Interactive Multisensor Snow and Ice Mapping System (IMS) Daily Northern Hemisphere Snow and Ice Analysis (National Ice Center 2008), where this field is available. The IMS field does not specify snow depth, but rather specifies the presence or absence of snow cover. Thus, the IMS snow cover field is used to trim and add to the CGM snow field. If the CGM indicates snow but IMS indicates no snow, then the snow depth is set to zero at that point. If the CGM indicates no snow, but IMS indicates snow, then a default snow depth is set at that point. The default snow depth is set such that the snow water equivalent is 3 cm. In the Noah Land Surface Model, there is a threshold snow depth at which snow first has its full effect on the land surface model equations. This threshold varies by land use type, but a water equivalent of 3 cm is approximately in the middle of the range of the thresholds.

2.3 Incorporating Observations

Incorporating observations into the WRF initial conditions can be a crucial step toward improving the model forecast. WREN_RT provides two ways that the user can choose to incorporate observations into WRF: 1) objective analysis and 2) observation nudging.

Observations are obtained by WREN_RT from MADIS, but the user can also provide other observations to WREN_RT as long as they are in little-r format (Section 2.2). MADIS observations are converted into little-r format using MADIS2LITTLER (which relies on the MADIS application programming interface [API]). MADIS2LITTLER is altered so that only data that are marked as having passed the highest level of available MADIS quality control are converted to little-r format (via setting *qcall=99* in *madis_to_little_r.f90*). The MADIS observations and any observations provided by the user in little-r format are combined into a single little-r file that is passed to littler_filter.

The program littler_filter has various filtering capabilities, but its default use in WREN_RT is to apply mesonet observations use/reject lists provided by NCEP. Because mesonet observations tend to have more data quality issues (e.g., due to poor siting) than other observations, mesonet observations were analyzed by NCEP to determine which mesonet observations should be used. This resulted in use/reject lists that were designed for possible use in the Real-Time Mesoscale Analysis (De Pondeca et al. 2011). Mesonet moisture observations from a station are removed by littler_filter if the station is in the overall moisture reject list or if it is daytime and the station is in the daytime-only moisture reject list. Mesonet temperature observations from a station are similarly removed if the station is in the overall temperature reject list, if it is daytime and the station is in the daytime-only temperature reject list, or if it is nighttime and the station is in the nighttime-only temperature reject list. Rather than only rejecting mesonet observations that are on a reject list, as is done for temperature and moisture, wind mesonet observations are rejected unless accept lists indicate that they should not be rejected. All observations from stations on the universal accept list are kept, but there is also a direction-specific accept list for which wind observations are retained as long as their wind direction is within a wind direction range that the list specifies should be retained for that specific station.

The observations that are not removed by littler_filter are passed into Obsgrid_sort_only to be sorted chronologically and then into Obsgrid (NCAR 2016) for further processing including quality control. For quality control, Obsgrid performs some gross error checks, compares observations to the CGM, and compares observations to nearby observations. Obsgrid performs quality control of

temperature, relative humidity, sea-level pressure, and surface pressure (Reen 2015; NCAR 2016). When Obsgrid compares multi-level observations (e.g., radiosondes) against the CGM for quality control, it interpolates the observations to the pressure levels for which CGM is available. Since the quality controlled, multi-level observations have been vertically interpolated, the vertical structure present in the original observation may not be well represented in what is assimilated. Therefore, the WPS preprocessor Ungrib is configured to interpolate the CGM to additional vertical levels, which provide the multi-level observations with additional verticals levels to be interpolated to and thus more of the vertical structure of the observations will be retained.

In addition to quality controlling the observations, Obsgrid is also used by WREN_RT to create an analysis. The analysis uses the CGM as the first-guess field and analyzes observations onto this first-guess field. WREN_RT is configured to default to using the Cressman-scheme option of Obsgrid to carry out the objective analysis. The observations used in the objective analysis are quality controlled by Obsgrid prior to being applied to the analysis.

When creating an objective analysis for a model domain, one issue that arises is that observations just outside the domain will not normally be included in the analysis even if they were only slightly outside the model domain. WREN_RT allows the user to choose to utilize Obsgrid's capability to use an expanded domain to do the analysis so that such observations can be included. This capability is also important in ensuring that initial conditions do not significantly differ between the outer edge of a domain and the bordering area in its parent domain. This could occur if an observation just outside the child domain is included in the objective analysis for the parent domain but not the objective analysis for the child domain. The resulting contrast at the edge of the child domain could reduce the accuracy of the model forecast. Obsgrid was modified to properly deal with more than one domain using an expanded domain to carry out the objective analysis.

Separate executions of Obsgrid are used to quality control observations for observation nudging and create the objective analysis. When creating the objective analysis, the CGM is used on its original pressure levels rather than being first interpolated to additional pressure levels as is done for the quality control of observations for observation nudging. This decreases the size of the WRF input files since the objective analysis used to initialize WRF is only on the original CGM levels rather than on the higher-vertical-resolution CGM levels.

After all of the executions of Obsgrid have completed, we have both an objective analysis that can be used as WRF initial conditions and observations that can be ingested into the model simulation via observation nudging. The user can choose

to apply one or both of these (via *use_obs_in_ic_and_bc_analyses* and *length_of_obs_nudging_minutes*).

Observation nudging is often applied to a preforecast time period in a configuration known as dynamic initialization. Observation nudging (Reen 2016) gradually nudges the model toward the observations by adding a nonphysical term to the tendency equation. The nonphysical term is based on the difference between the observation and the model value, the strength at which nudging is being applied, and spatiotemporal weighting. The user chooses the length of the observation nudging period at the beginning of the simulation (via *length_of_obs_nudging_minutes*) and WREN_RT applies its default nudging configuration unless the user overrides this value. By default, nudging is applied to temperature, water vapor mixing ratio, and winds. After the specified nudging period finishes, WREN_RT ramps down the influence of nudging during the next 60 min, but does not allow any observations with a valid time after the start of this rampdown period to be assimilated. The WREN_RT determines the radius of influence (ROI) of observations in observation nudging based on the horizontal grid spacing of each domain (Table 1). The most appropriate ROI for a given case may differ markedly from the default values here. The user is encouraged to experiment to determine the best ROI for the case under consideration; the default values are intended to provide a reasonable first guess of appropriate ROI. Note that the ROIs described here correspond to the WRF setting *obs_rinxy*. The ROI of above-surface observations is determined by multiplying *obs_rinxy* by a term that linearly increases with decreasing pressure from 1 at the surface to 2 at 500 hPa, and is 2 for all pressures lower than 500 hPa (Reen 2016). The ROI of surface observations is determined by multiplying *obs_rinxy* by the WRF setting *obs_sfcfacr*. The obs_rinxy values chosen by WREN_RT based on the domain horizontal grid spacing can be overridden by setting obs_rinxy in the [WRF] section of the WREN_RT configuration file. The surface ROI multiplier (*obs_sfcfacr*) determined via the template WRF configuration file can also be overridden via the [WRF] section of the WREN_RT configuration file.

Table 1 Base ROI used for observation nudging as a function of the WRF model domain horizontal grid spacing. Note that this is the WRF setting obs_rinxy and the actual ROI applied to an observation depends on whether the observation is a surface observation and for nonsurface observations depends on the pressure of the observation. See the text for details.

Horizontal grid spacing (km)		ROI (km)
≥	<	
…	2.5	20
2.5	4.0	45
4.0	9.0	60
9.0	18.0	120
18.0	24.0	180
24.0	…	240

2.4 Components

WREN_RT is a scripted system that executes various programs. In this section, we describe the component programs of WREN_RT.

2.4.1 WPS

The WPS is provided by NCAR and is an integral part of WREN_RT. WREN_RT currently uses a modified version of WPSV3.8.1, but uses a modified version of the WPSV3.9.0.1 version of Ungrib. WPS includes Geogrid, Ungrib, and Metgrid.

Geogrid interpolates static data such as terrain fields to the model grid. Geogrid was altered to allow WREN_RT to more easily specify which data sources to use and this modification was provided to NCAR and included in WPSV3.9. Another alteration made to Geogrid was to prevent users from combining within a domain land use data sets with different categories. Previously, if one used a land use data set that used some set of categories X but this data set did not cover the whole domain and the data set used to cover the rest of the domain used a set of categories Y, then when the input file stated that a location had a land use of N that could mean a different land use in different parts of the domain and WRF would have no knowledge of this difference. This modification was also provided to NCAR and included in WPSV3.9.

Ungrib converts Gridded Binary (GRIB) files into a format known as the WRF intermediate format for further processing by WPS. WREN_RT uses Ungrib to convert CGM data as well as SST data. The format of GFS data changed on 19 July 2017 at 1200 Coordinated Universal Time (UTC), and NCAR released

WPSV3.9.0.1 with changes to allow GFS data on or after that time to be properly processed. Ungrib was updated to this new version. The Ungrib code used in WREN_RT was also modified to properly ingest GALWEM data.

Metgrid horizontally interpolates data onto the WRF model domains. In WREN_RT, Metgrid processes CGM, SST, and snow data. Metgrid was modified to allow data on the World Geodetic System 1984 (WGS84) datum to be read in (i.e., data where Earth is an oblate spheroid rather than a sphere) so the NSIDC IMS snow data can be processed. Note that the code to process WGS84 data was already present and Metgrid needed to be modified to use this code and thus read the data correctly.

2.4.2 SNODAS2WRF

SNODAS2WRF was created by ARL and converts the NOHRSC SNODAS data into WRF intermediate format so that it can be processed by the WPS tool Metgrid.

2.4.3 NESDIS2WRF

NESDIS2WRF was created at ARL and converts the NSIDC IMS snow data into WRF intermediate format so that it can be processed by the WPS tool Metgrid. The "NESDIS" portion of the name refers to the National Environmental Satellite, Data, and Information Service, which established the NSIDC.

2.4.4 Edit_intermediate

The WPS utility Rd_intermediate reads WRF intermediate format files, and Edit_intermediate is a modified version of Rd_intermediate, which adds the capability to edit WRF intermediate format files. Specifically, Edit_intermediate creates a snow cover flag field based on the snow depth field (SNOWH). The snow cover flag field only indicates whether snow is absent or present. Since the NSIDC IMS snow data solely provides a snow cover flag field, the calculation of individual snow cover flag fields based on the snow information from the CGM and NOHRSC SNODAS provides the fields needed for Metgrid to merge the snow cover flag field among the data sets using the desired order of precedence.

2.4.5 MADIS2LITTLER

MADIS2LITTLER is provided by NCAR and is used to convert the MADIS observational data into the little-r format required for processing by Obsgrid. A modified version of MADIS2LITTLER V1.2 is being used. One modification is that if a data type specified for processing is not available at a given time, then data types not yet processed for that time will still be processed. In the standard version,

as soon as a data type chosen for processing is found to have no data for the given time, all data types not yet processed for that time are skipped. Another modification is to specify that the *is_sound* flag should be set to TRUE for single-level above-surface observations. This is how Obsgrid expects *is_sound* to be set in this case. These and other modifications have been submitted to NCAR for possible inclusion in the next version of MADIS2LITTLER. Note that MADIS2LITTLER relies on the MADIS API.

2.4.6 MADIS API

The MADIS API is provided by NOAA and provides an API to use in interacting with MADIS data files. WREN_RT uses MADIS API V4.3 with a bug fix to prevent ACARS profile (ACARSP) observations from erroneously being returned when ACARS (standard ACARS) observations are requested.

2.4.7 Littler_filter

Littler_filter was developed by ARL to implement the use/reject lists provided by NCEP to control which MADIS mesonet observations are processed. The application of the use/reject lists is how Littler_filter is used in the default configuration of WREN_RT, although the software has additional capabilities to filter out a certain percentage of observation or to include or exclude observations whose station ID are listed in include/exclude files.

2.4.8 Obsgrid_sort_only

Obsgrid_sort_only takes a single little-r observation file and chronologically sorts the observations and removes duplicates, outputting hourly little-r observation files. The program was written by NCAR but is not part of the standard WRF distribution.

2.4.9 Obsgrid

Obsgrid (NCAR 2016) is distributed by NCAR and quality controls observations and creates an analysis using observations and CGM data as described in Section 2.3. WREN_RT uses a modified version of the non-Flux-Adjusting Surface Data Assimilation System (FASDAS) version of Obsgrid V3.8.

Obsgrid allows one to trim the domain after the analysis is completed so that the analysis can include observations outside the model domain. However, in the standard version of Obsgrid, it does not update the metadata correctly when this is done. Obsgrid was modified to correctly update the metadata including in cases where an expanded domain was used on multiple domains.

Aspects of the Cressman scan analysis for the surface were modified for the case where the user chooses to use a smaller radius of influence at the surface. The changes were aimed at improving matching at the boundaries of domains, to allow larger-scale structures to be analyzed and incorporate fine-scale structures better.

Other changes have also been made to Obsgrid including changes to decrease the runtime of Obsgrid, thus making the code more efficient.

2.4.10 WRF

WREN_RT currently uses a modified version of WRF-ARW V3.9.1.1 as its mesoscale numerical weather prediction model. WRF-ARW (www2.mmm.ucar.edu/wrf/users/; Skamarock et al. 2008) is a widely used community model managed and distributed by NCAR. WRF has been widely applied in both research and operational settings. WRF within WREN_RT can be compiled with or without limited WRF-Chem (Grell et al. 2005) capabilities. There are two steps to running WRF: 1) run the Real program to convert the CGM data processed by Metgrid (in WPS) or the analyses constructed by Obsgrid from the CGM data and observations into initial condition and boundary condition files for WRF, and 2) run WRF.

One modification in the version of WRF used in WREN_RT is the changes to allow GALWEM data to be ingested. GALWEM data use vertically integrated soil moisture, and while WRF contains code to process these data, WRF does not recognize that the soil moisture provided by GALWEM is vertically integrated. Thus, WRF was modified to recognize that the GALWEM data processed within WREN_RT contains vertically integrated soil moisture.

Observation nudging is modified in how it treats surface observations within soundings. In the standard version of WRF, surface observations within soundings are treated as another level within the sounding. This is done even though surface observations are applied differently than above-surface observations. After the modification, surface observations within soundings are detached from the sounding and treated as surface observations.

Ingestion of snow data is modified to allow a snow cover flag field to be ingested. If the snow cover flag field is present, it is used to adjust the other snow fields (snow water equivalent depth and physical snow depth). In particular, the snow cover flag field is use to trim or add snow using the assumption that the snow cover flag field either more accurately or at least as accurately represents the area covered by snow. This allows data sources with only snow cover flags to be used to enhance the snow fields in WRF.

The Mellor-Yamada-Janjić (MYJ) planetary boundary layer (PBL) scheme (Janjić 2001) is modified to decrease the background turbulent kinetic energy (TKE) from 0.10 to 0.01 J kg^{-1} and adjust the PBL depth diagnosis method. Note that the altered PBL depth diagnosis is not used to replace what is used internally within the MYJ PBL but rather replaces what is passed out of this scheme for inclusion in model output and use in controlling the vertical spreading of the influence of observations in observation nudging. The goal of these modifications are to allow lower-TKE conditions to be better resolved and better diagnose the top of the PBL. The PBL depth diagnosis methodology is largely based on that used in the nonhydrostatic fifth-generation Pennsylvania State University–National Center for Atmospheric Research Mesoscale Model's (MM5) Gayno–Seaman PBL parameterization (Stauffer et al. 1999; Shafran et al. 2000). The modification was found to remove numerical noise in the TKE field in one case (Reen et al. 2016). This modified version has also been used in other studies (e.g., Lee et al. 2012, Reen et al. 2014).

2.5 Program Flow

WREN_RT is a scripted system using Python 3 scripts and a single Perl script. Figure 2 shows the basic program flow of the main WREN_RT Python script wren.py. After reading the WREN_RT configuration file (via *get_wren_config*), it uses the information from this file to determine directories (*prepare_run_dir*) and filenames (*set_file_names*) and derive additional settings (*get_wren_derived_settings*). After doing this, it calls a driver (*run_driver*) that will call the code needed to download CGM data and run the preprocessors via RUNWPSPLUS (a more detailed description of the driver is given later in this section). WREN_RT then derives WRF settings for which it relies on output from the preprocessors (WPS) to set (*get_wren_derived_settings_post_wps*). WREN_RT and then configures the WRF namelist and creates any configuration files needed by the namelist (*configure_wrf*). Based on the user's settings, WREN_RT then may run Real only or both Real and WRF. Finally, WREN_RT carries out any final tasks needed before WREN_RT finishes (*do_final*).

Basic Program Flow of wren.py

get_wren_config: read WREN_RT configuration file

prepare_run_dir: determine directory locations

get_wren_derived_settings: derive additional settings based on those read in

set_file_names: determine file names

run_driver(preproc): call a driver and instruct it to run "preproc", which downloads CGM data and runs preprocessors (via RUNWPSPLUS)

get_wren_derived_settings_post_wps: derive WRF settings that are best determined after WPS completes

configure_wrf: configure WRF namelist and create run-time I/O settings files

Run Real/WRF (as there are a number of function calls involved with this, the program flow is shown in Fig. 4)

do_final: complete any final tasks needed before WREN_RT finishes

Fig. 2 Overview of the program flow within the Python program wren.py. The bold type indicates the name of functions.

As noted previously, after the WREN_RT configuration file is read, the information contained in this file will be used to help determine the names of directories and files as well as several additional settings. There are a wide variety of additional settings determined at this time; the following are some examples:

- **Domain placement.** Based on the number of grid points the user chose for each domain, it will determine how to set settings related to placement of subdomains within the domain to ensure that the domains have the same center point. If needed, it adjusts the number of grid points within a domain to allow the domains to be co-centered. It will also ensure that there is enough room between the edges of the domains.
- **Grid expansion.** When it is specified that the grid should be expanded to include observations outside the domain in the initial conditions, WREN_RT will here calculate how much to expand the domain (assuming the user did not explicitly specify this) and calculate the values for grid-related variables that need to be changed in order to implement this.
- **Resolution of geographic data.** Based on the horizontal grid spacing of each domain, it determines the coarsest data set that has finer resolution than the horizontal grid spacing.

- **Map projection.** Based on latitude, this determines which projection to use. If the center latitude of the model is less than 2.0° latitude from the equator, it will use the Mercator projection. If the northernmost edge of the outer domain is at or farther north than latitude 83.0° or if the southernmost edge of the outer domain is at or farther south than latitude –83.0°, then the polar projection is used. Otherwise, the Lambert conformal projection will be used. This algorithm was based on MacCall et al. (2013). For the Lambert conformal projection, a tangent version was used. For a tangent version, the plane the Earth is being projected onto only touches the Earth at one latitude (in contrast to the secant version where it touches it at two latitudes). The equivalency of the tangent and secant projections is described by Glahn (1990).
- **Observation nudging.** Based on the horizontal grid spacings of each domain, it determines ROI setting used (see Table 1). The specified ROI is applied near the surface and increases linearly with pressure to twice this value at 500 hPa (Reen 2016). Note that these are default settings, and the user should consider adjusting these settings for the case under consideration.

After the user settings have been used to set various settings in WREN_RT, the next step in WREN_RT is to download CGM data and run preprocessors. A driver (*run_driver*) is used instead of calling the functions themselves that carry out the downloading of CGM data and running of the preprocessors because of the complications caused by some computing environments expecting different types of tasks to be completed in different means or on different portions of the machine. In some computing environments, the node the user directly interacts with (login node) is not intended for executing programs that take nontrivial resources, but is instead intended to allow the user to submit jobs to a queue that will run the program on another node intended for running programs (compute node). The driver used in WREN_RT to carry out the task of downloading CGM data and running the preprocessors will cause Python to be launched to run a script that carries out these tasks. Depending on whether the user specified a job submittal file for this task, the driver will either submit a job to the queuing system that will cause Python to be launched or will launch Python outside of the job queuing system. Since a new instance of Python is launched to start this task and thus we cannot directly pass variables to the script being launched, a Python "shelf file" (https://docs.python.org/3/library/shelve.html) is created to store the variables needed by this task. The Python script running this task can then read the variables from the shelf file. Note that for the task of downloading CGM data and running the preprocessors the use of standard compute nodes may not be possible. This task requires access to the

Internet to download data and compute nodes may not have access to the Internet; on the queue-based system on which WREN_RT was tested this task had to be submitted to a queue that ran jobs on nodes other than the standard compute nodes so that input files could be downloaded.

Whether the preprocessors are run in a queue or not, the preprocessors are run via the RUNWPSPLUS component of WREN_RT, which is driven by a Perl script (*runwpsplus.pl*). RUNWPSPLUS includes various components described previously including WPS, SNODAS2WRF, NESDIS2WRF, Edit_intermediate, MADIS2LITTLER, MADIS API, Littler_filter, Obsgrid_sort_only, and Obsgrid. The program flow of RUNWPSPLUS is summarized in Fig. 3. To better visualize the big picture, some aspects are included in the diagram that deal with non-RUNWPSPLUS portions of WREN_RT (shaded blue) or that are not part of WREN_RT at all (shaded red).

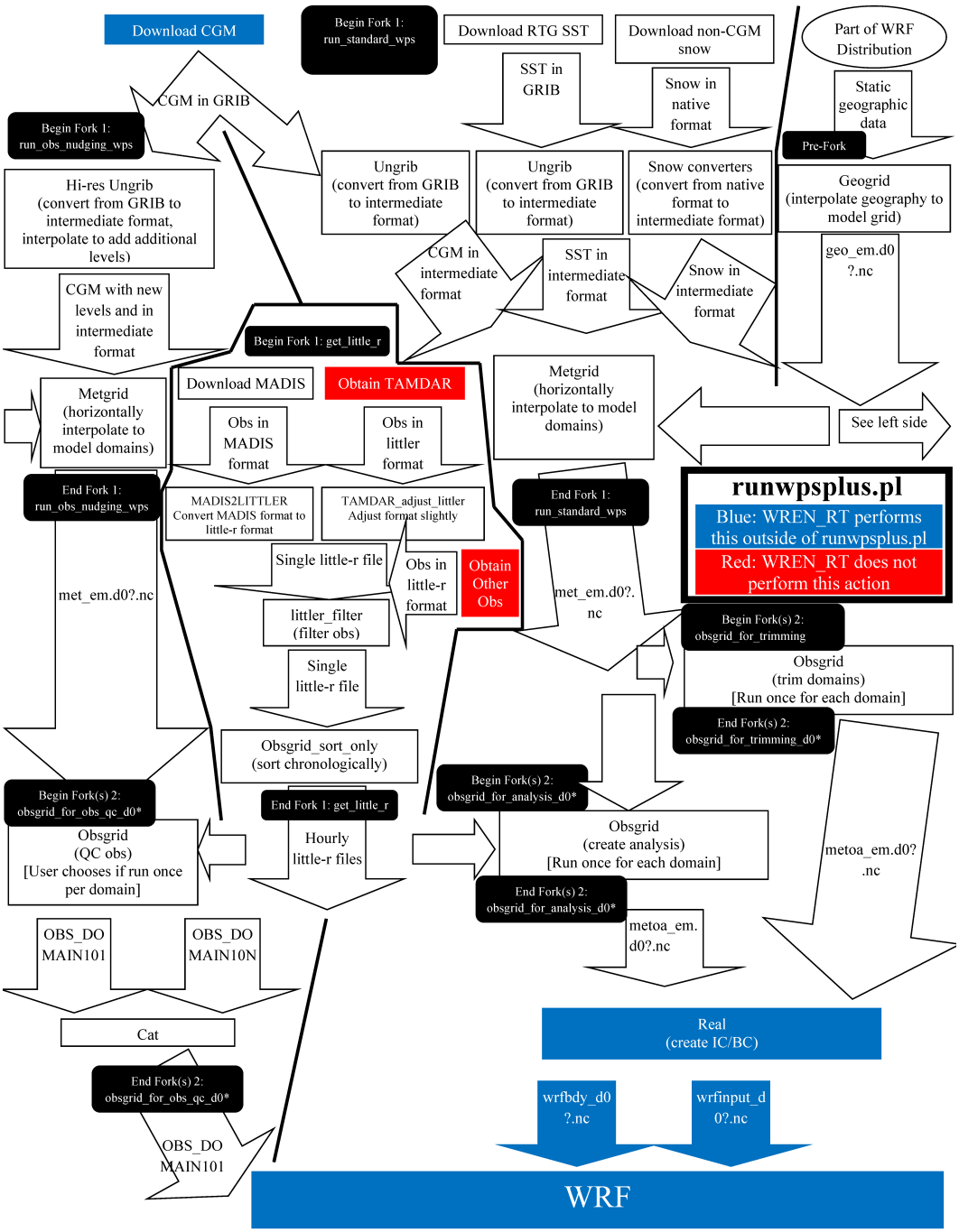

Fig. 3 Flowchart of the program flow for the RUNWPSPLUS component of WREN_RT

To more quickly complete the preprocessing, some programs in RUNWPSPLUS are run concurrently through use of Perl's fork capability. Much of the processing carried out by RUNWPSPLUS occurs during the two times in the program flow where multiple components of RUNWPSPLUS are being executed. Before the first fork occurs, the preprocessor Geogrid is run to create the static geographical data (e.g., terrain height), since these data are needed for more than one fork. The

different processes that ran concurrently at a given time are shown in Fig. 3 with black boxes indicating the start and end of each fork. All of the processes that start with the label "Begin Fork 1: …" and end with the label "End Fork 1: …" are run concurrently. After launching the processes in a given fork, the script waits until all of these processes finish before moving farther forward in the script.

The first time in the program flow where multiple components of RUNWPSPLUS are running concurrently is labeled "Fork 1" in Fig. 3 and involves three concurrent activities labeled *run_obs_nudging_wps*, *run_standard_wps*, and *get_little_r*. Both *run_obs_nudging_wps* and *run_standard_wps* execute the WPS components Ungrib and Metgrid, but *run_obs_nudging_wps* interpolates the data to additional pressure levels to provide additional information for the quality control of observations and does not include the special snow and SST data sources downloaded and used in *run_standard_wps* (since those data sources should not influence the quality control). These additional pressure levels allow additional vertical structure to be retained in multi-level observations (e.g., radiosondes) since the quality-control process vertically interpolates these observations to the pressure levels of the field being used to quality control them. These additional pressure levels also facilitate quality control of above-surface single-level observations (e.g., aircraft data) since it allows the vertical separation to be smaller on average between these observations and the pressure levels on which data used for the quality control are available. The *run_standard_wps* process downloads the RTG SST and the NOHRSC SNODAS and NSIDC IMS snow data, and processes them to the point that they are included in the Metgrid output files (named met_em.d0?.nc, where ? is the domain number) created by this process.

The final process in this first fork (*get_little_r*) obtains observations, may perform some filtering on the observations, and creates a list of chronologically sorted observations. In addition to downloading MADIS observations, if the user has specified other little-r formatted observations are available, then it will read in those observations. Also, in the special case where the user has obtained Tropospheric Airborne Meteorological Data Reporting (TAMDAR; Daniels et al. 2004) observations in little-r format, since some modifications to these data may be needed to fully utilize these data, WREN_RT applies TAMDAR_adjust_littler to make these adjustments (e.g., calculate dew point since Obsgrid ignores relative humidity and the only moisture variable provided in the file may be relative humidity). All of these observations are combined, filtered (e.g., using use/reject lists on mesonet data to remove stations known to provide bad data), and put in chronological order.

The second fork involves running Obsgrid for three separate purposes. The first purpose is to perform quality control on the observations (labeled *obsgrid_for_obs_qc_d0**). The user chooses whether this should be performed only on the outermost domain or for each domain separately. The end result is that for each domain there is one file of observations in the format needed for observation nudging in WRF. If the user wants to perform quality control only using the outermost domain, then the observation files created for each domain will be identical; this is not problematic since WRF will ignore any observations outside the current domain. The second purpose of running Obsgrid (labeled *obsgrid_for_analysis_d0**) is to create analyses that combine the CGM data downloaded by WREN_RT (e.g., GFS) and the observations. For this second purpose, Obsgrid will be run once for each domain being processed since each domain will need an analysis tailored to the grid spacing and area covered by the domain. If the user chooses, these analyses can be used in specifying the initial conditions used to start WRF (see setting *use_obs_in_ic_and_bc_analyses*). The third and final purpose of running Obsgrid (labeled *obsgrid_for_trimming*) is so that the files provided to Real cover the intended WRF computation domain. Since the user may have used an expanded domain (see setting *domain_expansion_for_ analysis*) so that observations slightly outside the domain are included in the analysis in the previous running of Obsgrid (*obsgrid_for_analysis_d0**), there is a requirement to ensure that the domains are trimmed to the intended size. Obsgrid has the capability to do this trimming when it is invoked for creating the analysis. However, the trimming is also required for times at which there is no need to use output from the *obsgrid_for_analysis_d0** runs of Obsgrid. Thus, this final run of Obsgrid is solely for the purpose of trimming the domains to the intended size. Note that when Obsgrid is run for more than one domain in any of these three purposes of running Obsgrid, each of the domains Obsgrid runs are also concurrent.

After RUNWPSPLUS is finished, WREN_RT then derives certain WRF settings for which it relies on output from the preprocessors (WPS) to set. Specifically, to determine the number of levels in the WPS output and the number of soil and land use categories, it relies on the tool NetCDF tool ncdump to extract information from the WPS output.

WREN_RT then configures the WRF namelist and creates files specifying how the input/output (I/O) of WRF should be adjusted compared to that specified in the WRF Registry files. The WRF Registry files determine which fields are output and input (and from which file), but changes to the Registry require recompiling WRF for them to take effect. An alternative is to use run-time I/O settings files to adjust the I/O. WREN_RT can use this capability to cycle the dust field; the dust field predicted by a previous simulation is used as the initial conditions for the dust field

for the current simulation. This allows for longer range dust transport and provides already spun up dust fields to start the simulation.

Based on the user's settings, WREN_RT then may run Real only or Real and WRF together. The user chooses via *run_real_location* and *run_wrf_location* whether to not run Real and WRF (*none*), run them on the current machine (*local*), or run them on a different machine (*remote*). A different choice can be made in this regard for WRF compared to Real, but not all combinations of choices will work. Figure 4 describes the program flow related to running Real and WRF.

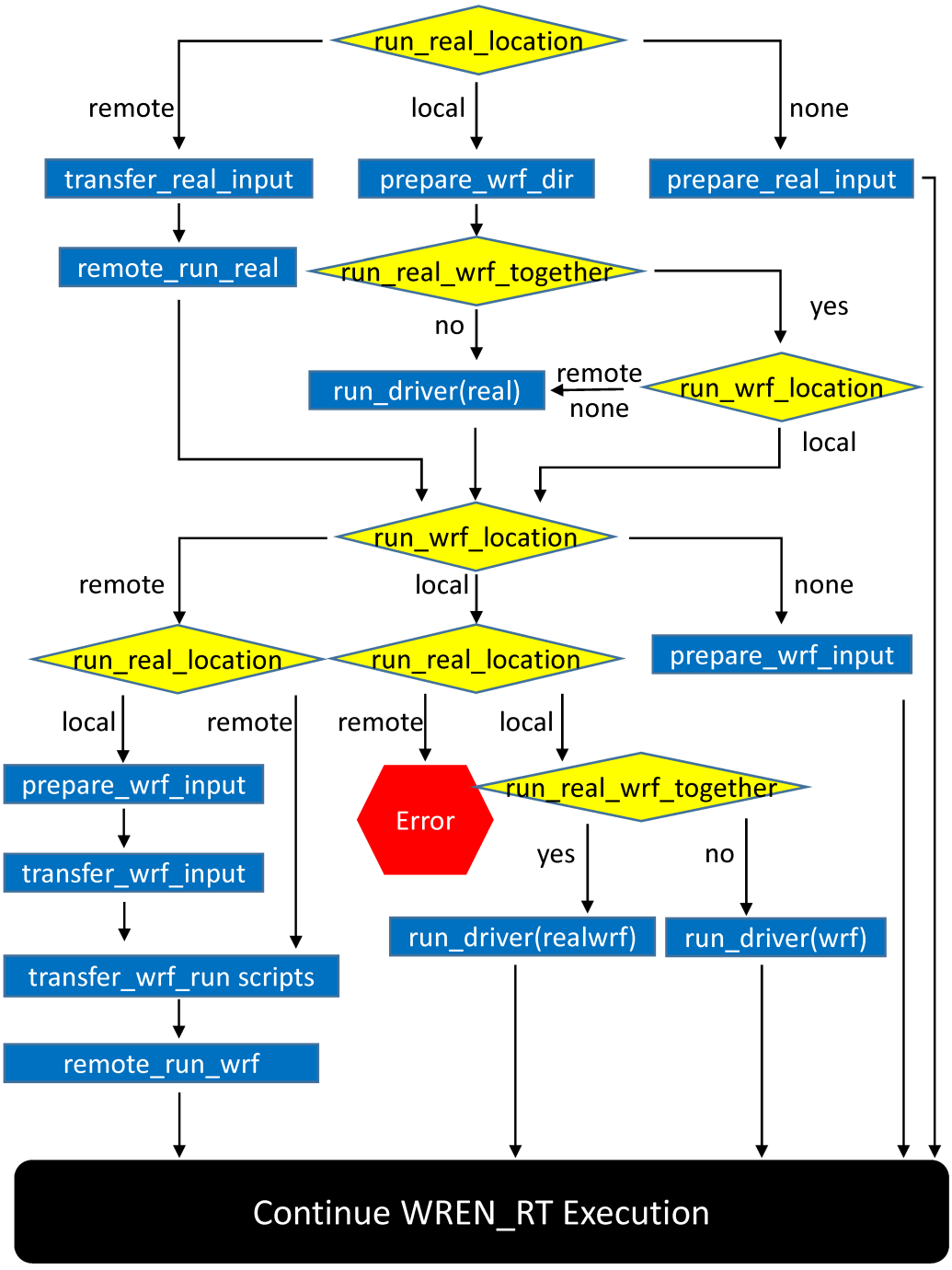

Fig. 4 Overview of the portion of the program flow in wren.py used for running Real/WRF. The yellow rhombuses represent decision points based on the setting of the variable shown in the box. The arrows proceeding from these are labeled to indicate which setting of this variable the arrow represents. The blue rectangles represent calls to functions; arguments to these functions are only shown if they are crucial to the program flow. Note that while the "Error" shown in the red octagon occurs if the user chooses the combination of run_wrf_location=local and run_real_location=remote, the error would actually be triggered much earlier in the WREN_RT program flow than the location shown here.

If the user chooses not to run Real and not to run WRF, then WREN_RT will create a file containing the input files for Real so that Real and WRF can be run outside of WREN_RT.

The capability to run Real and WRF remotely was designed for use on a specific pair of machines and so will likely require further development in order to apply it to other pairs of machines. It transfers the Real input data to the remote computer and sends commands to that machine to prepare the data and launch Real. It then monitors the job queue to determine when the process completes. After this, it sends a command to the remote computer to execute WRF and watches for the run to finish and then copies the WRF output from the remote computer to the local computer.

If the user chooses to run Real or WRF locally, then the same driver (*run_driver*) used to execute the preprocessors is used to execute real and WRF. In this case, the driver runs Real/WRF either directly (including any Message Passing Interface [MPI] command specified via *mpi_command_if_no_job_file*) or through a job script. This is in contrast to what was done in the driver (*run_driver*) for the process of downloading the CGM data and running the preprocessors where it instead launched Python directly or through a job script in order to run a Python script orchestrating the task. For the running of Real or WRF locally, if a job script is used then WREN_RT relies on the use of blocking to determine when Real and WRF complete; blocking ensures that the command to execute the job script does not return until the job completes. For example, on the system this capability was tested on, in order to implement blocking, *job_submit_command* should be set to "*qsub -Wblock=true*". The use of blocking simplifies the job of monitoring the completion of Real/WRF since WREN_RT only needs to wait until the job script returns to determine when the process completed (successfully or unsuccessfully). This is in contrast to the methodology used in the remote run capability of WREN_RT, which does not use blocking and instead checks the status of the job to determine when it finishes.

The user can choose whether or not to run Real and WRF together (via *run_real_wrf_together*) to minimize wait time in queues when submitting a job file to execute Real and WRF. This option allows a single job file to execute both Real and WRF so that a single job file can be submitted to the queue rather than submitting a job script to execute Real, waiting for this job to run and complete, and then submitting a job script to execute WRF, and then waiting for this job to run and complete. Real normally requires much fewer computational resources than WRF and thus when run in a single job file with WRF has available more resources than needed to complete execution in a timely manner. However, since Real

normally takes very little time to execute, the time period of which we are using more resources than needed is quite small. The disadvantage of running Real and WRF via a single job file is that diagnosis of problems in running Real can be made more difficult because WRF is executed immediately after Real with no chance for WREN_RT to diagnose potential problems in the execution of Real before WRF executes. Thus, if it appears that Real fails to execute correctly, rerun WREN_RT and specify that WRF should not be run (*run_wrf_location=none*).

Finally, after Real and WRF complete (or, if the user chose not to execute them, after passing the place in the code where they would have been run), WREN_RT carries out any final tasks needed before WREN_RT finishes (*do_final* in Fig. 2). These final tasks involve moving and copying files so that they are in the expected locations. The data are saved to an archive location if one is specified via *archive_dir*.

Note that multiple instances of WREN_RT can be running simultaneously from the same installation since basically all of the output from the simulation is in the run-specific output directory (specified via *output_directory*) chosen by the user. An exception to this is that while WREN_RT is being executed the main diagnostic log file for WREN_RT does not reside in *output_directory* so that the log file can contain diagnostic output from before the output directory is created. However, this main diagnostic log file is given a temporary name, which includes a 32-character random string to minimize the chance that two concurrent WREN_RT executions would try to use the same log file (the user is notified of the name of this file via a print to the terminal). Before the WREN_RT execution ends, the main diagnostic log file has the random characters removed and is placed in *output_directory*.

3. Installing and Running WREN_RT

3.1 Installing WREN_RT

WREN_RT is developed within a Git (https://git-scm.com/) repository. Git is a widely used source code management tool and its use has been critical in the development and maintenance of WREN_RT. RUNWPSPLUS and WRF are separate Git repositories from the WREN_RT repository and thus are treated as submodules within WREN_RT. If the repository is accessible to the user, Git is the best method to obtain the source code. Since the repository contains submodules, the user should use the "*--recursive*" option when executing the Git clone command to retrieve a copy of the source code.

A script to install WREN_RT (*install_wren.py*) is part of the repository (in the *install* subdirectory). In addition, a separate script (*install_config_only_wren.py*) is available for users to create their own "config-only" installation that relies on an existing installation. The config-only installation creates a directory with user configuration files copies from an existing full installation and creates a link to the main WREN_RT executable (*wren.py*) in the full installation. This allows the user to run WREN_RT from this directory (and configure WREN_RT as needed), but use the executables and scripts from the full installation. This means the config-only installation requires no compilation. The config-only installation is much simpler to complete and thus makes it easier for users to get started with using WREN_RT. It also allows a single full installation to be updated as needed rather than every user needing to complete an update of a full installation. When the full installation is updated, the config-only installations will automatically start using the updated version since they rely on the files in the full installation. The one exception to this is updates to the configuration files since the config-only installation necessarily relies on local copies of those files. However, rerunning the config-only installation should provide updated example versions of these files for the user to reference. It is recommended that only a single full installation of WREN_RT be completed on a computer and that subsequently users should complete config-only installations relying on that single full installation.

For either installation type, Python 3.x must be available on the machine on which WREN_RT is being installed. For the full install, the gfortran compiler (https://gcc.gnu.org/fortran/) must also be available. The full installation script installs various programs needed by WREN_RT components as well as the WREN_RT components themselves.

For the full install, in the install subdirectory, there is an example install configuration file (*install.config_example*) that the user can copy over to *install.config* and alter as needed before executing the full install script.

Note that the static geographical data are not included in the Git repository due to the large size of these data. While the full-installation script includes code to automatically download these data, the user may want to consult with the developers to ensure the proper data are obtained since the data on the NCAR website are sometimes updated when WRF is updated and because WREN_RT can utilize special geographical data sets that are not available from the NCAR website.

The user should determine whether WRF-Chem capability is desired and set *install_wrf_chem* appropriately. If the user does not anticipate using WRF-Chem, then this should be set to "*no*" because WRF-Chem can greatly increase memory requirements and output file size. Note that at least one version of the gfortran

compile (v4.4.7) appears to be unable to compile WRF-Chem even though that version of gfortran is capable of compiling WRF without chemistry. Currently, the only application of WRF-Chem in WREN_RT that has been tested is a dust-only configuration.

To run the full installation, go to the install subdirectory and run

python3 install_wren.py

Note that on some machines, the Python 3 executable may have a different name. The "*python*" executable very likely is Python 2, which is incompatible with WREN_RT and thus cannot be used to launch the WREN_RT install script.

To run a config-only installation:

python3 /example/wren_rt/install/install_config_only_wren.py local_wrenrt

Note: */example/wren_rt* is the location of the full install and *local_wrenrt* is the directory in which the user wishes to install the config-only installation.

3.2 Running WREN_RT

To run WREN_RT, first configure the WREN_RT configuration file. Example configuration files are included in the WREN_RT Git repository (*wren.config_example* and *wren.config_*_example*) and a description of the settings can be found in the Appendix. To run WREN_RT, it is necessary that Python 3, Perl, and Network Common Data Form (netCDF) are available (since WREN_RT uses Python 3 and Perl scripts and relies on the netCDF executable ncdump). If running Real/WRF with MPI, then MPI software must also be available. The command to run WREN_RT is the following:

python3 wren.py –config_file myconfigfile.config

Note: *myconfigfile.config* is the name of the user's WREN_RT configuration file.

4. Applications of WREN_RT

There are multiple ways in which the WREN_RT software can be applied. Some examples include that WREN_RT 1) may be applied as an automated real-time modeling system, 2) as a tool to execute individual simulations as might be done to carry out research, 3) as a cycled dust-modeling system, and 4) as a tool to create quality-controlled observations for model verification.

4.1 Real-Time Modeling System

WREN_RT has been designed to be applied as an automated real-time modeling system. All processes needed to download the necessary input data, process them, and run the models have been automated.

One aspect of WREN_RT that supports this ability is the design of the configuration file, which allows the user to choose the start date and time of the model simulation relative to the current time. The user can choose start year, month, day, hour, and minute normally; for example, for a three-domain configuration, to start the simulation on each domain at 0600 UTC on 5 July 2018, one could specify the following:

- start_year = 2018, 2018, 2018
- start_month = 7, 7, 7
- start_day = 5, 5, 5
- start_hour = 6, 6, 6
- start_minute = 0, 0, 0

Note that there are three comma-separated values in each line since there will be a value for each domain. While the preceding example is probably the most straightforward way to set the start date and time for a single simulation, if one is trying to run an automated real-time simulation system, this method is less helpful. The user would somehow need to determine a way to replace the values in the configuration file based on the current day. To avoid this complication and potential source of errors, WREN_RT also allows the user to specify the date and time relative to the current date and time. The current date and time are specified by entering -100. Thus, if one wanted to run WREN_RT daily at 0600 UTC one could setup a method to automatically start WREN_RT each day (e.g., via crontab) and use the following settings:

- start_year = -100, -100, -100
- start_month = -100, -100, -100
- start_day = -100, -100, -100
- start_hour = 6, 6, 6
- start_minute = 0, 0, 0

For values other than year or month, one can also specify values relative to -100. For example, *start_day* being set to -101 would indicate the simulation should start one day ago.

Another aspect of WREN_RT relevant to real-time automated simulations is that it is designed with the ability to try to successfully complete even if expected input data do not download on the first attempt. Part of this is the ability to set certain errors regarding obtaining external data to be nonfatal (i.e., the inability to download certain data sets will not cause WREN_RT to crash). For example, if one sets *madis_transfer_error_fatal=0* in the [RUNWPSPLUS] section of the WREN_RT configuration file, then even if WREN_RT cannot access the MADIS data set to download observations, WREN_RT will continue to execute. Similar settings (all in the [RUNWPSPLUS] section) are in place for retrieval of RTG SST data (*sstrtg_retrieve_error_fatal*), SNODAS data (*snodas_retrieve_error_fatal*), and NESDIS data (*nesdis_retrieve_error_fatal*). However, WREN_RT cannot run without some sort of CGM data, since it needs lateral boundary conditions and initial conditions. For CGM data, WRF will look for an earlier cycle of the data if the current cycle is not available and will continue to work backward through the CGM cycles to find one with the correct data available until the maximum number of cycles have been checked. WREN_RT also works to retry downloading of files if the first attempt to download them fails.

Future work could enhance WREN_RT by adding the capability to create graphics of the output and run verification on the output (work on the latter component is described in Dawson et al. [2016] and Dawson and Raby [2018]).

4.2 Individual Simulations

While WREN_RT can be applied to automated real-time simulations, it has also been designed to support running individual simulations including in support of research. The design of WREN_RT aims to allow the user to choose how involved to become in the configuration of the simulations. At one end of the continuum, some users may desire to obtain a simulation of a specific case for investigation and do not have specific ideas on how the simulation should be configured, whereas at the other end of the continuum, users may have detailed plans on exactly how the model should be configured. WREN_RT aims to support users along this continuum. The methodology by which users configure WREN_RT (described in Section 2.1) is designed to support this continuum of users.

4.3 Cycled Dust Modeling

WREN_RT has been used to run cycled dust-modeling simulations. In this configuration, WRF-Chem is run in a configuration in which the only chemistry component activated is dust. Simulations rely on dust predictions from a previous model integration to supply the dust initial conditions in order to allow for a spun-up dust field at the beginning of the model simulation and account for long-distance dust transport.

To apply WREN_RT in this manner, one must first ensure that when WREN_RT is being installed the installation of WRF-Chem is enabled. This is done by setting *install_wrf_chem= yes* in *install/install.config*. When WREN_RT is configured, one must ensure that processing of the erodibility fields is enabled. This is set in the RUNWPSPLUS configuration file by setting wrf_chem=1; however, in the default RUNWPSPLUS template configuration file, this should already be set in this manner, so no action by the user should be required. If it is not set correctly, one can add an override in the WREN_RT configuration file in the [RUNWPSPLUS] section by adding a line with the following:

- wrf_chem=1

WRF itself must then be configured to model dust. In the *&chem* section of the WRF namelist, the following settings should be included:

- chem_opt = 401, 401, 401, 401, 401, 401
- dust_opt = 3,
- chem_conv_tr = 1, 1, 1, 1, 1, 1,
- chem_in_opt = 0, 0, 0, 0, 0, 0,
- aer_ra_feedback = 1, 1, 1, 1, 1, 1,
- gas_drydep_opt = 0, 0, 0, 0, 0, 0,
- dust_smois = 1,
- wetscav_onoff = -10, -10, -10, -10, -10, -10

The settings with repeated comma-delimited values only require one per domain, so the configuration shown here would be appropriate for any configuration with up to six domains. These settings should already be set in this manner in the default WRF template configuration file. If they are not set in this manner in the WRF template configuration file that is being used, then they can be set in the [WRF] section of the WREN_RT configuration file. Note that if WRF is not compiled with

chemistry capabilities, then the *&chem* section of the namelist is ignored; thus, it is not problematic to have these settings in the namelist even if WRF-Chem is not being used. Note that this configuration does not allow the radiation scheme to see the effects of the dust, since enabling this capability significantly increases the time WRF takes to complete. To allow the radiation scheme to see the effects of the dust, *chem_opt* should be set to *300* or *301*; however, testing indicated that this increased runtime by a factor of approximately equal to 5 and approximately equal to 6, respectively.

Implementing the settings described up to this point in this section should result in dust modeling being enabled, but cycling of dust fields may not being enabled. To allow cycling of dust fields to occur, the user must set *do_dust_cycling_prelim=yes* in the WREN_RT configuration file and set via *cycling_input_directory*, the location of the files with the dust fields to use as initial conditions in the current simulation. WREN_RT will look for single-time *wrfout** files for each domain whose filename indicates is valid at the time the model is starting. If it finds these files, it will set up WRF to extract the following fields from those files to use as the dust initial conditions: DUST_1, DUST_2, DUST_3, DUST_4, and DUST_5 (each of these fields represents dust of a certain size). If it does not find these files, then dust cycling will be disabled. To enable dust cycling, WREN_RT ensures the following lines are in the *&time_control* section of the WRF namelist:

- iofields_filename = “iofields.txt”, “iofields.txt”, “iofields.txt”
- ignore_iofields_warning = .TRUE.,
- io_form_auxinput16 = 2,
- auxinput16_interval = 999999, 999999, 999999
- auxinput16_inname = “wrf_dustcycle_d<domain>”,

These settings indicate that the run-time I/O capability of WRF should be used to adjust the I/O configuration. To change which fields are input or output from WRF and from or to which files, normally a Registry file is changed and WRF is recompiled. The run-time I/O capability allows these settings to be changed without recompiling WRF. The setting *iofields_filename* indicates that the file “*iofields.txt*” will determine how the I/O configuration should be changed. This file is created by WREN_RT and contains the following:

-:i:0:DUST_1,DUST_2,DUST_3,DUST_4,DUST_5
+:i:16:DUST_1,DUST_2,DUST_3,DUST_4,DUST_5

This file indicates that the five dust fields listed should not be input from input stream 0 (*wrf_input**) but instead from input stream 16. The WRF namelist settings indicate that stream 16 is a netCDF formatted file (*io_form_auxinput16=2*), is named *wrf_dustcycle_d<domain>* where *<domain>* is the two digit representation of the domain number, and new values should be read from this file every 999999 min (the value is set to an arbitrary large number to ensure that it is only read in at the beginning of the simulation). WREN_RT creates a copy of the *wrfout** files to be used for dust cycling and names them *wrf_dustcycle_d<domain>* where *<domain>* is the two-digit representation of the domain number.

In the future, the cycling mechanism applied here could be expanded to cycle additional variables (e.g., soil moisture).

4.4 Verification

In addition to preparing the input data for WRF and running the WRF model, WREN_RT can also be used as a tool to provide quality-controlled observations for verification. WREN_RT normally obtains quality controls and other processes observations so that the observations can be assimilated in two possible ways. The observations may be assimilated in the initial conditions via Obsgrid analyzing the observations onto the first-guess field determined from the CGM output (e.g., GFS), or via observation nudging during the beginning of the model integration. However, observations obtained for these purposes will only be obtained for the time period over which they are needed (the initial time for analyzing onto the first-guess field and up until the end of the observation nudging time period if observation nudging is being applied). For real-time simulations, observations will not be available for all or most of the forecast period (the time after observation nudging is applied), since the forecast period will be in the future. Thus, WREN_RT will usually not try to obtain and quality control observations for the entire time period of the simulation.

After the end of the simulated period is in the past, the user can choose to obtain, process, and quality control observations for the entire simulation period (both during and after the end of the observation nudging preforecast) by specifying *process_obs_for_verification=yes*. Obsgrid was modified to produce little-r output files with Earth-relative winds (files named *qc_obs_used_earth_relative.** that can be used by the Model Evaluation Tools (MET; Fowler et al. 2018) via ASCII2NC. These files are in addition to the little-r output files with grid-relative winds previously output by Obsgrid. Note that this modification was provided to NCAR and is now in the standard version of Obsgrid.

5. Conclusion and Future Work

WREN_RT is a software tool that allows automated execution of a WRF simulation and all of the data gathering and preprocessing needed to create the files to run this simulation. The software is designed to allow those with limited WRF experience to be able to run WRF using a reasonable configuration while allowing experienced WRF users the ability to customize the configuration in more detail. WREN_RT includes the ability to download CGM data (GFS, NAM, and HRRR) and use GALWEM data if already downloaded. WREN_RT can download higher-resolution SST and snow data to augment the information coming from the CGM. WREN_RT can obtain MADIS observations and perform quality control on them. It can apply these observations to an objective analysis in Obsgrid to create updated initial conditions and boundary conditions, and can also apply the observations though observation nudging during a preforecast period of the model to improve the model conditions at the end of the preforecast nudging period; these conditions will then be used as the initial conditions for the model forecast. WREN_RT is designed for application both in environments that use job scripts to submit programs to a queue and in environments without job queues. WREN_RT can be applied for real-time automated modeling, the running of individual simulations, or the production of quality-controlled observations for use in verification. In addition, WREN_RT has been designed to run dust simulations using WRF-Chem that cycle the dust fields from one simulation to another.

One way in which WREN_RT could be improved would be to strengthen capabilities to ingest data from the Air Force. Currently, Air Force CGM GALWEM data can be ingested by WREN_RT, but Air Force observations cannot be processed by WREN_RT, nor any higher-resolution SST and snow data than what are available via GALWEM. The ability to run solely on Air Force data would strengthen the applicability of WREN_RT by allowing it to run in environments where NCEP data are not available.

Currently, while WREN_RT supports observation nudging, analysis nudging is not supported. Whereas observation nudging nudges the model solution toward observations during a preforecast time period, analysis nudging nudges the model solution toward an analysis during a preforecast time period. Future work to implement analysis nudging would provide a capability that might improve WREN_RT's forecast skill.

WREN_RT currently can assimilate observations from the MADIS database including surface, radiosonde, aircraft, satellite-derived wind, and so on, but cannot assimilate radar data. Where radar data are available, they provide higher spatial

and temporal density than the observations currently ingested by WREN_RT. However, the assimilation of radar observations cannot easily be completed through the existing assimilation techniques in WREN_RT (observation nudging and objective analysis via Obsgrid), and thus inclusion of radar data assimilation in WREN_RT is not a trivial task. However, due to the potential high value of assimilating radar data, expansion of WREN_RT to include this capability should be considered. Lidar is another potentially valuable data source whose inclusion in WREN_RT data assimilation should be considered.

The ability to use output from a previous WRF simulation as the initial conditions for a new WRF simulation would allow for cycled simulations. Currently, the capability exists to cycle dust fields, but the capability could be expanded to cycle other fields (e.g., soil moisture). To cycle all fields, WREN_RT could be altered to use WRF's restart capability. The restart capability offers the ability to start a new run that proceeds exactly as if the previous run had continued executing. For example, consider the case where run A is integrated from 1200 to 1800 UTC, and then a restart file created by run A at 1500 UTC is used to initialize run B, which is integrated to 2100 UTC. The run A and run B forecasts at 1800 UTC would then be identical. However, in the context of WREN_RT, observation nudging could be applied in run A from 1200 to 1500 UTC and in run B from 1500 to 1800 UTC. Then run B would benefit from the model spinup that took place in run A between 1200 and 1500 UTC, but include observations between 1500 and 1800 UTC that were not available when run A executed. There are advantages and disadvantages of applying cycling, but having a restart capability in WREN_RT could allow the user the opportunity to explore the value of such configurations.

Another area of potential future work is translating the Perl script in the RUNWPSPLUS component into Python, the language used by non-RUNWPSPLUS portions WREN_RT. If the RUNWPSPLUS components were in Python, it would allow refactoring of the code to take place more easily and allow for more code reuse within WREN_RT and thus greater efficiency in development.

Increasing the robustness of WREN_RT would increase its value, particularly in real-time applications. In real-time applications, the end user may be depending on WREN_RT output to be available at a specific time and may have limited expertise on how to diagnose situations where WREN_RT output is not available when expected. Methods to increase the robustness of WREN_RT may include automatically rerunning portions of the system that crash unexpectedly or more extensive capabilities to switch from one data source to another.

Another area of potential future work is in extending the products created by WREN_RT. Scripted creation of graphics of WRF output would allow users to

easily see what the model is predicting soon after the simulation completes. Graphics could even be produced as soon as the model produces output for a particular time, rather than waiting for the entire model simulation to complete. Integrated verification of the WRF output would allow the value of WREN_RT to be better understood and could guide future WREN_RT improvement efforts. A tool to automate part of the verification effort was described by Dawson et al. (2016), and work to improve part of this process has been described in Dawson and Raby (2018).

6. References

Daniels TS, Murray JJ, Grainger CA, Zhou DK, Avery MA, Cagle MF, Tsoucalas G, Schaffner PR, Neece RT. Validation of tropospheric airborne meteorological data reporting (TAMDAR) temperature, relative humidity, and wind sensors during the 2003 Atlantic THORPEX regional campaign and the Alliance Icing Research Study (AIRS II). In: 11th AMS Conference on Aviation, Range, and Aerospace, P8.2. Hyannis (MA): American Meteorological Society; 2004 [accessed 2018 Aug 16]. https://ams.confex.com/ams/pdfpapers/81761.pdf.

Dawson L, Raby J, Smith J. The automation of nowcast model assessment processes. Adelphi (MD): Army Research Laboratory (US); 2016. Report No.: ARL-MR-0940.

Dawson LP, Raby JW. Automated postprocessing for the Weather running estimate-nowcast (WRE-N) model. Adelphi (MD): Army Research Laboratory (US); 2018. Report No.: ARL-MR-0988.

De Pondeca MSFV, et al. The real-time mesoscale analysis at NOAA's national centers for environmental prediction: Current status and development. Weather and Forecasting. 2011;26:593–612.

Fowler T, Halley Gotway J, Newman K, Jensen T, Brown B, Bullock R. Model evaluation tools version 7.0 (METv7.0) user's guide. Boulder (CO): Developmental Testbed Center; 2018 [accessed 2018 Aug 13.]. https://dtcenter.org/met/users/docs/users_guide/MET_Users_Guide_v7.0.pdf.

Gemmill W, Katz B, Li X. Daily real-time, global sea surface temperature—High-resolution analysis: RTG SST HR. College Park (MD): NOAA/National Weather Service, National Centers for Environmental Prediction, Environmental Modeling Center; 2007. Report No.: 260.

Glahn HR. 1990: The equivalency of the tangent and secant lambert conformal map projections. Monthly Weather Review. 1990;118:2781–2783.

Grell GA, Dudhia J, Stauffer DR. A description of the fifth generation Penn State/NCAR Mesoscale Model (MM5). Boulder (CO): National Center for Atmospheric Research (US); 1995. Report No.: NCAR/TN-3981STR.

Grell GA, Knoche R, Schmitz R, McKeen SA, Frost G, Skamarock WC, Eder B. Fully-coupled "online" chemistry within the WRF model. Atmospheric Environment. 2005;39:6957–6976.

Janjić Z. Nonsingular implementation of the Mellor-Yamada level 2.5 scheme in the NCEP Meso model. College Park (MD): National Centers for Environmental Prediction; 2001. Report No.: 437 [accessed 2018 June 28]. http://www.lib.ncep.noaa.gov/ncepofficenotes/files/on437.pdf.

Kirby SF, Reen BP, Dumais RE Jr. An automated weather research and forecasting (WRF)-based nowcasting system: Software description. Adelphi (MD): Army Research Laboratory (US); 2013. Report No.: ARL-SR-0283.

Lee JA, Kolczynski WC, McCandless TC, Haupt SE. An objective methodology for configuring and down-selecting an NWP ensemble for low-level wind prediction. Monthly Weather Review. 2012;140:2270–2286.

MacCall BT, Haines PA, Reen BP, Cogan JL. Computation considerations for the weather research and forecasting (WRF) and Pennsylvania State University / National Center for Atmospheric Research Mesoscale Model Fifth Generation (MM5) Map Projections. Adelphi (MD): Army Research Laboratory (US); 2013. Report No.: ARL-TR-6480.

National Center for Atmospheric Research (NCAR). User's guide for the Advanced Research WRF (ARW) modeling system version 3.8. Boulder (CO): National Center for Atmospheric Research; 2016 [accessed 2018 June 27]. http://www2.mmm.ucar.edu/wrf/users/docs/user_guide_V3.8/contents.html.

National Operational Hydrologic Remote Sensing Center. Snow Data Assimilation System (SNODAS) data products at NSIDC. Boulder (CO): National Snow and Ice Data Center; 2004. doi: https://doi.org/10.7265/N5TB14TC.

National Ice Center. IMS daily northern hemisphere snow and ice analysis at 4 km resolution, version 1. Boulder (CO): National Snow and Ice Data Center; 2008. doi: https://doi.org/10.7265/N52R3PMC.

Reen BP. Improving weather research and forecasting model initial conditions via surface pressure analysis. Adelphi (MD): Army Research Laboratory (US); 2015 Sep. Report No.: ARL-TR-7447. http://www.arl.army.mil/arlreports/2015/ARL-TR-7447.pdf.

Reen BP, Schmehl KJ, Young GS, Lee JA, Haupt SE, Stauffer DR. Uncertainty in contaminant concentration fields resulting from atmospheric boundary layer depth uncertainty. J Applied Met and Climatology. 2014;53:2610–2626.

Reen BP, Dumais RE Jr, Passner JE. Mitigating excessive drying from the use of observations in mesoscale modeling. J Applied Met and Climatology. 2016;55:365–388.

Reen BP. A brief guide to observation nudging in WRF. Boulder (CO): University Corporation for Atmospheric Research; 2016 [accessed 27 June 2018]. http://www2.mmm.ucar.edu/wrf/users/docs/ObsNudgingGuide.pdf.

Shafran PC, Seaman NL, Gayno GA. Evaluation of numerical predictions of boundary layer structure during the Lake Michigan Ozone Study (LMOS). J Applied Met. 2000;39:412–426.

Stauffer DR, Muñoz RC, Seaman NL. In-cloud turbulence and explicit microphysics in the MM5. Preprints, Ninth PSU/NCAR MM5 Model Users' Workshop; 1999; Boulder, CO. pp. 177–180.

Skamarock WC, Klemp JB, Dudhia J, Gill DO, Barker DM, Duda M, Huang XY, Wang W, Powers JG. A description of the advanced research WRF version 3. Boulder (CO): National Center for Atmospheric Research; 2008. Report No.: 475 [accessed 2018 June 25]. http://www2.mmm.ucar.edu/wrf/users /docs/arw_v3.pdf.

Appendix. Weather Running Estimate – Nowcast Real Time (WREN_RT) Settings

This appendix describes settings in the main Weather Running Estimate – Nowcast Real Time (WREN_RT) configuration file, which has the default location of *../config/wren.config* (i.e., where the path is relative to the location where wren.py is executed from). The user may specify an alternative name via the command-line option "--*config_file*". Since in the config-only installation, all of the users WREN_RT files are in a single directory and ../config does not exist, for config-only installations, the user should use "--*config_file*" to specify the location of the configuration file. For example, the default configuration filename *../config/wren.config* will be used if WREN_RT is invoked as follows:

python3 wren.py

However, if WREN_RT is instead invoked as the following:

python3 wren.py –config_file new_wren_config_file

then the file *new_wren_config_file* will be read and used as the configuration file.

The configuration file has three sections. The [WREN_RT] section contains settings specific to WREN_RT, and the [RUNWPSPLUS] section contains any settings in the RUNWPSPLUS template namelist that the user wishes to manually override. Similarly, the [WRF] section contains any settings in the template Weather Research and Forecasting model (WRF) namelist that the user desires to manually override.

A-1 [WREN_RT] Section of WREN_RT Configuration File

Within the [WREN_RT] section, some settings are domain-specific, and in this case, first the value for the outermost domain is listed, then a comma occurs, then the value for the next domain is listed, and so on. Note that comments are included within the example WREN_RT configuration files to describe each setting and to note whether a setting is domain-specific or not.

While some settings have been discussed or referred to in the main text, this appendix provides a complete description of the settings in a single location for reference.

- *output_directory* = The directory in which output from WREN_RT will be placed while the program is being executed. This setting is not domain-specific.
- *start_year* = The four-digit integer year (Coordinate Universal Time [UTC]) in which the model should start. Alternatively, *-100* can be specified

as a placeholder for the current year. No other negative values are allowed. This setting is domain-specific.

- *start_month* = The integer representing the month (UTC) in which the model should start. Alternatively *-100* can be specified as a placeholder for the current month. No other negative values are allowed. This setting is domain-specific.
- *start_day* = The integer representing the day of the month (UTC) in which the model should start. Alternatively *-100* can be specified as a placeholder for the current day, and other negative values can be used to specify days relative to the current day. For example, *-101* indicates the day before today. This setting is domain-specific.
- *start_hour* = The integer representing the hour of the day (UTC) in which the model should start. Alternatively *-100* can be specified as a placeholder for the current hour, and other negative values can be used to specify hours relative to the current hour. For example, *-101* indicates the hour before the current hour. This setting is domain-specific.
- *start_minute* = The integer representing the minute of the hour (UTC) in which the model should start. Alternatively *-100* can be specified as a placeholder for the current minute, and other negative values can be used to specify minutes relative to the current hour. For example, *-101* indicates the minute before the current minute. This setting is domain-specific.
- *run_length_minutes* = The length of time in integer minutes between the model time at which the model is to start integrating until the model time it is to stop integrating. This setting is domain-specific.
- *wrf_coarse_grid_timestep_seconds* = The timestep in decimal seconds that the outermost model domain will use. WREN_RT will use the relationship between the horizontal grid spacing on the outermost domain and on subsequent domains to calculate the timestep on that domain, and continue the process through the nests. For example, given 9-, 3-, and 1-km domains, and this setting set to *27.0* s, the ratio between the horizontal grid spacing on the outermost domain and the next domain is 3 and thus the ratio will be the same for the timestep and thus the timestep on the 3-km domain will be 1/3 of 27.0 s and thus 9 s. Similarly the 1-km domain would have a timestep of 3 s. This setting is not domain-specific.
- *output_interval_minutes* = The interval between output times in integer minutes. This setting is domain-specific.

- *dx_meters* = The horizontal grid spacing in decimal meters. This setting is domain-specific.
- *grid_points_x_prelim* = The preliminary number of grid points in the x-direction. WREN_RT will use this as a starting point and as long as *manually_place_domains=no* will adjust the number of grid points in the x-direction as needed to ensure that the grids align (i.e., the corners of a child grid align with a grid point in the parent grid) and that the grids are co-centered. If this setting is omitted then it is set to "*no*". This setting is domain-specific.
- *grid_points_y_prelim* = The preliminary number of grid points in the y-direction. WREN_RT will use this as a starting point and as long as *manually_place_domains=no* will adjust the number of grid points in the y-direction as needed to ensure that the grids align (i.e., the corners of a child grid align with a grid point in the parent grid) and that the grids are co-centered. This setting is domain-specific.
- *manually_place_domains* = If set to "*no*" WREN_RT will create co-centered domains with dimensions close to those chosen in *grid_points_x_prelim*, *grid_points_y_prelim*. If set to "*yes*", WREN_RT will use those dimensions exactly and align the grids as specified by the user in *start_x_in_parent_grid* and *start_y_in_parent_grid*; if the corners of the grids do not align properly, WREN_RT will stop running. This is not a domain-specific setting.
- *start_x_in_parent_grid* = The grid point in the parent domain where the left edge of the domain starts. This setting only has an effect if the user set *manually_place_domains=yes*. This setting can be omitted from the namelist if *manually_place_domains=no*. This is a domain-specific setting, and the value for the first domain should be set to *1* (although the value for the first domain may not have any effect).
- *start_y_in_parent_grid* = The grid point in the parent domain where the bottom edge of the domain starts. This setting only has an effect if the user set *manually_place_domains=yes*. This setting can be omitted from the namelist if *manually_place_domains=no*. This is a domain-specific setting, and the value for the first domain should be set to *1* (although the value for the first domain may not have any effect).
- *latitude_of_center* = The decimal latitude at the center of outermost domain. Positive values are used for latitudes north of the equator and negative values for latitudes south of the equator. If *manually_place_domains=no*,

this is also the latitude at the center of all other domains. This is not a domain-specific setting.

- *longitude_of_center* = The decimal longitude at the center of outermost domain. Positive values are used for longitudes east of the Prime Meridian and negative values for longitudes west of the Prime Meridian. If *manually_place_domains=no*, this is also the longitude at the center of all other domains. This is not a domain-specific setting.

- *number_of_wrf_vertical_levels_prelim* = Either the number of vertical levels in the model or set to "*0*" if the vertical levels will be specified via *wrf_sigma_levels_prelim*. If set to a value other than "*0*", WREN_RT will look for a file named *wrf_predefined_X_levels.txt* (where X is the value *number_of_wrf_vertical_levels_prelim* is set to) in the directory where the WREN_RT configuration file resides and will read X comma-separated values starting with 1.0 and ending at 0.0 and use these as the model sigma-levels. This is not a domain-specific setting.

- *wrf_sigma_levels_prelim* = The sigma values used to determine the vertical placement of levels in WRF. These values should be comma-separated decimals and have the first value of 1.0, and the final value of 0.0. This setting only has an effect if *number_of_wrf_vertical_levels_prelim=0*. This is not a domain-specific setting, but is a comma-separated list.

- *pressure_at_model_top_pa* = The pressure at the top of the model in pascals (note that 1 mb = 1 hPa = 100 Pa). This setting determines the upper extent of the model. This is not a domain-specific setting.

- *use_obs_in_ic_and_bc_analyses* = Should observations be used to create an analysis to enhance the model initial conditions and boundary conditions. If "*yes*", then Obsgrid will be used to create analyses by combining whatever observations are available based on user settings and a first-guess based on the coarse-grid model (CGM) data. If "*no*", then the CGM data itself will be used to create initial conditions and boundary conditions. This is not a domain-specific setting.

- *domain_expansion_for_analysis* = Should a larger domain be used during preprocessing than will be eventually used for WRF; this larger domain is for the purpose of enhancing the initial condition analyses created if *use_obs_in_ic_and_bc_analyses=yes*. By using a larger domain for the analyses, observations just outside the edge of a given domain can still influence the analysis, thus allowing for creation of a better analysis and for analyses that are more consistent among the domains. After the analysis is

completed, Obsgrid will trim the domain back to the intended size for ingestion into WRF. If this setting is omitted from the configuration file, it will be set to *-1*. This is a domain-specific setting.

  - If this is set to "*0*", the domain will not be expanded for analysis.
  - If this is set to "*-1*", then WREN_RT will choose how much the domain will be expanded. In this case, WREN_RT will first attempt to expand the domain by 520 km on each side; this is done because the largest Cressman scan in the default Obsgrid configuration is 520 km and thus this would allow any observations within this radius to be available for inclusion in the analysis. However, to prevent the domains from containing too many points and thus taking too much memory, time, or disk space, the maximum expansion is limited to 150 grid cells per edge. The number of grid points to expand by found by this procedure is then adjusted to the smallest value equal or larger than this that will work with the current grid geometry.
  - If this is set to any positive value, then WREN_RT will attempt to expand the domain by the specified amount. However, if this does not work with the current grid geometry then WREN_RT will choose the smallest expansion larger than this value that will work with the current grid geometry.

- *landuse_source* = which land use database should be used to specify land use. This is not a domain-specific option. Current options are the following:
  - *0* = Use the default in WPSV3.8.x and WPSV3.9.x International Geosphere-Biosphere Programme Moderate Resolution Imaging Spectroradiometer (IGBP MODIS) 30-arc-second resolution data with global coverage.
  - *1* = Use the default pre-WPSV3.8 US Geological Service (USGS) 30-arc-second resolution data with global coverage.
  - *2* = Use an National Center for Atmospheric Research (NCAR)-provided data set that uses National Land Cover Database (NLCD) 2006 mapped to USGS categories with added categories for lava, playa, and white sand. This data set has 1-arc-second resolution, but is not available as part of the standard WRF geographical data and is not part of the WREN_RT Git repository. Since it only covers part

of the contiguous United States, for areas not covered by this data set the USGS data set will be used.

  - *3* = Use the Coordinate Information on the Environment (CORINE) data set mapped to USGS categories. This data set has an approximately 4-arc-second resolution. This data set is not part of the standard WRF geographical data and is not part of the WREN_RT Git repository. This data set only covers parts of southeastern Europe (Portugal, Spain except for the northeastern corner, and parts of France). For areas not covered by this data set, the USGS data set will be used.

- *microphysics_scheme* = which microphysics scheme should WRF use. See the WRF documentation to determine the mapping between the integer chosen here and the microphysics scheme. This setting is not domain-specific.
- *longwave_scheme* = which longwave scheme should WRF use. See the WRF documentation to determine the mapping between the integer chosen here and the longwave scheme. This setting is not domain-specific.
- *shortwave_scheme* = which shortwave scheme should WRF use. See the WRF documentation to determine the mapping between the integer chosen here and the shortwave scheme. This setting is not domain-specific.
- *pbl_scheme* = which planetary boundary layer (PBL) scheme should WRF use. See the WRF documentation to determine the mapping between the integer chosen here and the PBL scheme. This setting is not domain-specific.
- *surface_layer_scheme* = which surface layer scheme should WRF use. See the WRF documentation to determine the mapping between the integer chosen here and the surface layer scheme. This setting is not domain-specific.
- *land_surface_scheme* = which land surface scheme should WRF use. See the WRF documentation to determine the mapping between the integer chosen here and the land surface scheme. This setting is not domain-specific.
- *cumulus_scheme* = which cumulus scheme should WRF use. See the WRF documentation to determine the mapping between the integer chosen here and the cumulus scheme. This setting is domain-specific as may cumulus

parameterizations are only intended to be applied on relatively coarse domains.

- *shallow_cumulus_scheme* = should the shallow cumulus parameterization by enabled. This is the WRF *ishallow* option and only works with certain deep cumulus parameterizations (i.e., this only works with certain values of *cumulus_scheme*). This setting is not domain-specific.
- *length_of_obs_nudging_minutes* = the length of the time period over which observation nudging will be used in integer minutes. Note that the rampdown time period occurs after the end of this length of time. Also note that this time interval starts at the start time of the coarsest domain; if finer domains start at a later time, then the actual time nudging will be enabled on the finer domain will be less than the value specified. This setting is domain-specific.
- *coarse_grid_model* = the CGM (e.g., Global Forecast System [*GFS*] model, North American Model [*NAM*], High-Resolution Rapid Refresh [*HRRR*], or Global Air-Land Weather Exploitation Model [*GALWEM*]) that will be used to determine model boundary conditions and initial conditions. This setting is not domain-specific.
- *coarse_grid_model_time_interval* = the time interval (in hours) of the CGM data used to initialize WRF (e.g., are the data from a given cycle available every 3 h or every 6 h). If this is set to "*0*", then the default interval for the chosen CGM data set will be used. This setting is not domain-specific.
- *coarse_grid_model_resolution* = the horizontal resolution of the CGM data used to create the initial conditions and boundary conditions. This value is only used for GFS (i.e., *coarse_grid_model=GFS*) and currently the only values recognized are *0.25*, *0.50*, and *1.00* degrees. This setting is not domain-specific.
- *use_already_downloaded_cgm* = should CGM data downloaded outside of WREN_RT be used or should WREN_RT download the data themselves. If "*no*", then WREN_RT will download the data themselves. If "*yes*", then WREN_RT will use the settings *already_downloaded_cgm_directory* and *use_all_cgm_data_in_directory* to determine what data to process. This setting is not domain-specific.
- *already_downloaded_cgm_directory* = where should WREN_RT find CGM data if it is not to download them (i.e., *use_already_downloaded_cgm=yes*). This setting is not a domain-specific setting.

- *use_all_cgm_data_in_directory* = should WREN_RT attempt to process all of the CGM data in the specified directory or should it only process the files it expects. If "*yes*", then WREN_RT will attempt to process all CGM data in that directory. If "*no*", then WREN_RT will determine the filenames it expects given the chosen CGM, the time period to process, and so on, and only attempt to process those. Note that the filenames of the CGM data obtained may not match up with the filenames of the data that WREN_RT would have downloaded; thus one should not rely on the data manually obtained having the same naming convention as the data WREN_RT would have attempted to download. This setting is not a domain-specific setting.

- *madis_account_type* = what type of MADIS account type should be used to obtain observations. If the user has no account, then the "*public*" account type should be chosen. The "*research*" and "*government*" account types should provide access to additional observations but require the user to obtain an account and to enter the MADIS user name and password while WREN_RT is executing, which prevents automated non-user attended execution. This setting is not a domain-specific setting.

- *process_obs_for_verification* = should observations be obtained and quality controlled throughout the entire model time period for purposes of verification. If "*yes*", then observations will be processed for the time period of the entire model run. If "*no*", then observations will only be processed for the time period needed for observation nudging at the beginning of the simulation (*length_of_obs_nudging_minutes*) and for enhancement of initial conditions and boundary conditions (if *use_obs_in_ic_and_bc_analyses=yes*). This is not a domain-specific setting.

- *do_dust_cycling_prelim* = should dust cycling be enabled. If "*yes*", then it will look for files from which to obtain the dust fields to cycle in *cycling_input_directory*. Specifically, it will look for single-time *wrfout** files for each domain whose filename indicates that it is valid at the time the model is starting. If it finds this, it will set up WRF to extract the following fields from those files to use as the dust initial conditions: DUST_1, DUST_2, DUST_3, DUST_4, and DUST_5. If "*no*", then it will not look for dust cycling files. This is not a domain-specific setting.

- *cycling_input_directory* = the directory in which WREN_RT will look for files to use for dust cycling if *do_dust_cycling_prelim=yes*. This is not a domain-specific setting.

- *run_real_location* = where should Real be executed. If "*none*", then Real will not be executed. If "*remote*", then it will run Real on *remote_hostname*. If "*local*", then Real will be executed on the current machine. This is not a domain-specific setting.
- *run_wrf_location* = where should WRF be executed. If "*none*", then WRF will not be executed. If "*remote*", then it will run WRF on *remote_hostname*. If "*local*", then WRF will be executed on the current machine. This is not a domain-specific setting.
- *run_real_wrf_together* = should Real and WRF be executed as a single job. This option is only used if *run_real_location* and *run_wrf_location* are both set to "*local*" and a job file is being used to run Real and WRF. In this case, if set to "*yes*", then both the execution of Real and the execution of WRF occur from the same job script, and thus within the same job. This can be beneficial in that two waits in the queue (one for Real and one for WRF) are not needed. If it is set to "*no*" and Real and WRF are executed via a job script, then each of them will be executed via a separate job script and thus Real will be submitted to the queue and after Real finishes, WRF will be submitted to the queue. If this setting is omitted, it will default to "*no*". This is not a domain-specific setting.
- *template_job_script_preproc* = The name of the template job script to use for running preprocessors. Set this to "*None*" if the preprocessors should not be submitted to a queue via a job script. Note that WREN_RT expects certain placeholders to be in this file, and so one should base the job script on the template included with WREN_RT (*qsub_hpc_preproc_template_example*). If this setting is omitted, then it is set to "*None*". This is not a domain-specific setting.
- *template_job_script_real* = The name of the template job script to use for running Real (when *run_real_wrf_together=no* or for some reason Real and WRF cannot be run together even though *run_real_wrf_together=yes*). Note that WREN_RT expects certain placeholders to be in this file, and so one should base the job script on the template included with WREN_RT for running Real and WRF together (*qsub_hpc_realwrf_template_example*). Set this to "*None*" if Real should not be submitted to a queue via a job script. If this setting is omitted, then it is set to "*None*". This is not a domain-specific setting.
- *template_job_script_wrf* = The name of the template job script to use for running WRF (when *run_real_wrf_together=no* or for some reason Real

and WRF cannot be run together even though *run_real_wrf_together=yes*). Note that WREN_RT expects certain placeholders to be in this file, and so one should base the job script on the template included with WREN_RT for running Real and WRF together (*qsub_hpc_realwrf_template_example*). Set this to "*None*" if WRF should not be submitted to a queue via a job script. If this setting is omitted, then it is set to "*None*". This is not a domain-specific setting.

- *template_job_script_realwrf* = The name of the template job script to use for running Real and WRF together (when *run_real_wrf_together=yes* and nothing prevents Real and WRF from being run together). Note that WREN_RT expects certain placeholders to be in this file, and so one should base the job script on the template included with WREN_RT (*qsub_hpc_realwrf_template_example*). Set this to "*None*" if Real and WRF should not be submitted to a queue via a job script. If this setting is omitted, then it is set to "*None*". This is not a domain-specific setting.

- *job_submit_command* = the command that should be used to submit jobs to the queue. This will be used for any portions of WREN_RT where *template_job_script_** is not set to "*None*". This command should be constructed so that it does not return until the job finishes so that WREN_RT can properly diagnose when the job finished. On the system on which WREN_RT was developed, passing the option "*-Wblock=true*" to qsub will accomplish this. If this setting is omitted, then it is set to "*None*". This is not a domain-specific setting.

- *mpi_command_if_no_job_file* = the command that should be used to execute certain commands via Message Passing Interface (MPI) when the corresponding *template_job_script_** setting is set to "*None*". For Real and WRF, if the corresponding *template_job_script_** setting is set to "*None*", then the MPI command will be applied. However, for the preprocessors, even if *template_job_script_preproc=None*, the MPI command will not be used. Set this to "*None*" if *template_job_script_*=None* and the user does not wish to invoke MPI for programs that normally use MPI (Real and WRF). If this setting is omitted, then it is set to "*None*". This is not a domain-specific setting.

- *wrf_output_base* = directory under which run-specific directory will store WRF output. If set to "*None*", then the WRF output stays in the directory in which it is created. Otherwise, WRF output is moved to the subdirectory specified by *wrf_output_directory_sub* within *wrf_output_base*. This is not a domain-specific setting.

- *wrf_output_directory_sub* = the subdirectory under *wrf_output_base* where WRF output will be moved to (note that this is only the WRF output and not the other output created by WREN_RT). The user can directly specify the name of the subdirectory, or specify "*YYYYMMDD_HHUTC"* to indicate that the subdirectory is to be the start date/time of the simulation. If *wrf_output_base=None*, then the WRF output is not moved and *wrf_output_directory_sub* has no effect. This is not a domain-specific setting.

- *archive_dir* = the directory under which WREN_RT output should be archived. Note that this includes not only any WRF output files but also all other output created by WREN_RT. The archive is created by applying tar and gzip to create a single compressed file containing all of the output of WREN_RT. If *archive_dir=None*, then the WREN_RT output is not transferred to an archive location. If this setting is omitted, then it is set to "*None*". This is not a domain-specific setting.

- *archive_name* = the name of the file (without suffixes) wherein the archive of WREN_RT output should be placed. If this is set to the string "*YYYYMMDD_HHUTC*", then the filename will be the start date/time of the simulation; otherwise, the string specified will be the pre-suffix portion of the name. The actual name of the file will have .tar.gz appended to note that tar and gzip have been applied to gather the files into a single file and compress the data. The file will be placed in the directory specified by *archive_dir*, and if *archive_dir=None*, then the setting *archive_name* has no effect because no archive is made of the WREN_RT output. If this setting is omitted, then it is set to "*None*". This is not a domain-specific setting.

- *archive_server* = the name of the server on which should be placed in the directory *archive_dir* the archive file specified via *archive_name* containing all of the WREN_RT output. If this is set to "*None*", then the archive file will be kept on the local computer, whereas if this is not set to "*None*", WREN_RT will use rcp to copy the archive data to the specified directory (*archive_dir*) on the specified by server *archive_server*. However, if *archive_dir=None*, then the setting *archive_name* has no effect because no archive is made of the WREN_RT output. If this setting is omitted, it will be set to "*None*". This is not a domain-specific setting.

- *ignore_sanity_checks* = should WREN_RT ignore the results of sanity tests it completes to check if certain settings are configured reasonably. If "*no*", then WREN_RT will produce a warning and exit if certain settings are

configured in way that WREN_RT considers unreasonable. For example, if any grid has fewer than 100 grid points in a direction, WREN_RT would exit. Two other conditions the sanity check looks at are whether the time step is specified to be too large or the edge of one domain is too close to the edge of another grid. If this is set to "*yes*", then WREN_RT ignores the sanity tests. This is not a domain-specific setting.

- *user_email_address* = when Real or WRF is run on a remote machine (*run_real_location=remote* or *run_wrf_location=remote*), where should an email be sent regarding completion status. If set to "*None*", no email is sent. If this setting is omitted, then it is set to "*None*". This is not a domain-specific setting.
- *remote_hostname* = the machine on which Real or WRF should be run if *run_real_location=remote* or *run_wrf_location=remote.* This is not a domain-specific setting.
- *remote_input_directory* = the directory on *remote_hostname* where input data for Real or WRF should be sent if Real or WRF are to be run on *remote_hostname* (*run_real_location=remote* or *run_wrf_location= remote*). This is not a domain-specific setting.
- *remote_run_directory* = the directory on *remote_hostname* where Real and WRF executables are located. This is only used if Real or WRF are to be run on *remote_hostname* (*run_real_location=remote* or *run_wrf_location= remote*). This is not a domain-specific setting.

A-2 [RUNWPSPLUS] Section of WREN_RT Configuration File

The [RUNWPSPLUS] section of the WREN_RT configuration file is used to specify variables in the RUNWPSPLUS template configuration file that the user would like to override. This section may be left blank if no such manual overrides are desired. The settings placed in this section will override the settings found in the RUNWPSPLUS template configuration file *runwpsplus.config_wren_template* and will also override any settings WREN_RT sets in this file (WREN_RT uses the settings in the [WREN_RT] section to determine some of the RUNWPSPLUS settings, but this is done before the [RUNWPSPLUS] section settings are applied). The user should be careful in overriding RUNWPSPLUS settings, since it also prevents WREN_RT from ensuring that RUNWPSPLUS settings are consistent with one another.

The RUNWPSPLUS template configuration file contains three types of settings: 1) settings used by RUNWPSPLUS itself, 2) settings used by the WRF

Preprocessing System (WPS), and 3) settings used by Obsgrid. Comments are present within the RUNWPSPLUS template configuration file to describe the settings.

Note that while settings in the RUNWPSPLUS template configuration are of the following form:

$use_madis_mesonet = 1;

When overriding a setting in the [RUNWPSPLUS] section of the WREN_RT configuration file, the dollar sign ($) at the beginning of the line and the semicolon (;) at the end of the line are omitted:

use_madis_mesonet = 1

Other than the removal of the symbol at the beginning and ending of the line, the override setting in the [RUNWPSPLUS] section of the WREN_RT configuration file should be of the exact same format as that in the RUNWPSPLUS configuration file.

A-2.1 RUNWPSPLUS-Specific Settings

The settings used by RUNWPSPLUS itself are listed in the RUNWPSPLUS configuration file first. Here, we only describe a subset of these settings, since the user is more likely to want to modify a subset of the settings. For example, some settings indicate the directories in which various RUNWPSPLUS components exist within the RUNWPSPLUS installation, and it is unlikely that the user would have any reason to change the RUNWPSPLUS directory structure. Another example of settings the user is unlikely to need to change are those which are set by WREN_RT based on settings in the WREN_RT configuration file. In light of this, of the settings in the RUNWPSPLUS configuration file used by RUNWPSPLUS itself, only a sampling of those settings that the user may want to modify is described.

WREN_RT sets some of the RUNWPSPLUS-specific settings in the RUNWPSPLUS configuration file. These settings are the following:

SCRIPT_CODE_DIR, *SCRIPT_DATA_DIR*, *SCRIPT_RUN_DIR*, *WPS_INPUT_3DMODEL_LOC*, *process_obs_for_verification*, *use_obs_in_ic_and_bc_analyses*, *force_cgm_ungrib_to_startend_at_multiple_of_xh*, *length_of_obs_nudging*, *madis_access_type*, *madis_retrospective*, *obs_time_padding_backward*, *obs_time_padding_forward*, *force_obsgrid_to_startend_at_multiple_of_xh*, *cgm_name*

Since the settings just listed are set by WREN_RT, the user should be aware that if these settings are manually set in the [RUNWPSPLUS] section of the WREN_RT configuration file, the value that is manually set will override the value the WREN_RT determines should have been used for the setting. However, changing how the settings just listed are set in the template RUNWPSPLUS configuration file will not be effective, since WREN_RT will override whatever setting the template file has for these specific variables.

Description of selected settings in the RUNWPSPLUS configuration file that are specific to RUNWPSPLUS (i.e., not WPS or Obsgrid settings) are the following:

- *use_user_geo_em_files* = should the pre-created Geogrid output files (*geo_em**) files be used instead of those produced by WREN_RT via RUNWPSPLUS. If "*1*", then Geogrid output files in the directory *USER_GEO_EM_DIR* are used. If "*0*", then the Geogrid output files are created by WREN_RT via RUNWPSPLUS. This setting can be helpful in cases where special data have been used to create the Geogrid output files. It can also be helpful in cases where Geogrid is slow to execute (because of the use of high-resolution data), and thus the user wants to speed up WREN_RT execution by not needing to run Geogrid every time WREN_RT is run. Since the Geogrid output files are not time-specific, the same files can be used for different dates.

- *USER_GEO_EM_DIR* = the directory in which RUNWPSPLUS will look for Geogrid output files to use if *use_user_geo_em_files=1*. If *use_user_geo_em_files=0*, then USER_GEO_EM_DIR has no effect.

- *include_user_littler* = should a user-supplied observation file in little-r format be ingested in addition to whatever other observations are ingested. If "*1*", then the little-r file specified by *USER_LITTLER_DIR_FNAME* will be ingested prior to the application of observational quality control and filtering. If "*0*", then only the observations otherwise specified will be ingested.

- *USER_LITTLER_DIR_FNAME* = the name of the little-r observation file that will be ingested if *include_user_littler=1*.

- *use_tamdar* = are Tropospheric Airborne Meteorological Data Reporting (TAMDAR) observations available in little-r files that require special processing. If "*0*", then there are no TAMDAR observations available in little-r files that require special processing; in this case, TAMDAR observations may still be processed but must have been obtained via MADIS or in a little-r file that does not need special processing and can be

ingested via *include_user_littler=1*. If "*1*", then TAMDAR observations are available in little-r files that require special processing in the directory *TAMDAR_RAW_FILES_DIR*.

- *TAMDAR_RAW_FILES_DIR* = the directory where TAMDAR observations in little-r format will be obtained from for special processing if *use_tamdar=1*.

- *use_madis_mesonet* = should Meteorological Assimilation Data Ingest System (MADIS) mesonet observations be processed. Note that availability of some mesonet observations may depend on the MADIS account type used to access the data. If "*1*", then RUNWPSPLUS will attempt to download these observations and process them. If "*0*", then RUNWPSPLUS will not attempt to download the files with these observations.

- *use_madis_profiler_npn* = should MADIS National Oceanic and Atmospheric Administration (NOAA) profiler network observations be processed. Note that this network no longer exists; it appears that this data set is available in MADIS from 1 July 2001 until approximately 30 August 2014 (https://madis.noaa.gov/madis_npn.shtml). If "*1*", then RUNWPSPLUS will attempt to download these observations and process them. If "*0*", then RUNWPSPLUS will not attempt to download the files with these observations.

- *use_madis_profiler_map* = should MADIS multi-agency profiler Cooperative Agency Profilers (CAP) network observations be processed. If "*1*", then RUNWPSPLUS will attempt to download these observations and process them. If "*0*", then RUNWPSPLUS will not attempt to download the files with these observations.

- *use_madis_acars* = should MADIS aircraft-based observations be processed. The name of the directory on the MADIS server from which these data are obtained is "acars", which refers to Aircraft Communications Addressing and Reporting System (ACARS) observations, and thus the setting name in RUNWPSPLUS contains "acars". However, while this data set contains ACARS data, it also contains aircraft data that are not part of ACARS. Note also that some observations from this data set may not be available until at least a certain amount of time has passed since they were taken (perhaps 48 h); this restriction may depend on the type of MADIS account being used to access the data. If "*1*", then RUNWPSPLUS will attempt to download these observations and process them. If "*0*", then

RUNWPSPLUS will not attempt to download the files with these observations.

- *use_madis_maritime* = should MADIS maritime observations (ship and buoy) be processed. If "*1*", then RUNWPSPLUS will attempt to download these observations and process them. If "*0*", then RUNWPSPLUS will not attempt to download the files with these observations.

- *use_madis_metar* = should MADIS METAR (surface) observations be processed (the acronym METAR is from the French MÉTéorologique Aviation Régulière but in English is referred to as an aviation routine weather report). These data include Automated Surface Observing System (ASOS), Automated Weather Observing System (AWOS), and nonautomated reports in METAR and SPECI format (https://madis.ncep.noaa.gov/madis_metar.shtml). Note that SPECI appear to be METAR reports that are not at regular intervals but rather prompted by changes in the weather. This data set offers global coverage starting on 13 March 2006 (https://madis.ncep.noaa.gov/madis_metar.shtml). If "*1*", then RUNWPSPLUS will attempt to download these observations and process them. If "*0*", then RUNWPSPLUS will not attempt to download the files with these observations.

- *use_madis_raob* = should MADIS radiosonde observations be processed. If "*1*", then RUNWPSPLUS will attempt to download these observations and process them. If "*0*", then RUNWPSPLUS will not attempt to download the files with these observations.

- *use_madis_sao* = should MADIS surface airways observations (SAOs) be processed. It appears that there may no longer be any observations being added to this data set; thus, for recent cases, this data set may not contribute any observations. If "*1*", then RUNWPSPLUS will attempt to download these observations and process them. If "*0*", then RUNWPSPLUS will not attempt to download the files with these observations.

- *use_madis_satwind* = should MADIS 3-h satellite-derived wind observations be processed. If "*1*", then RUNWPSPLUS will attempt to download these observations and process them. If "*0*", then RUNWPSPLUS will not attempt to download the files with these observations.

- *use_madis_satwind1h* = should MADIS 1-h satellite-derived wind observations be processed. If "*1*", then RUNWPSPLUS will attempt to download these observations and process them. If "*0*", then

RUNWPSPLUS will not attempt to download the files with these observations.

- *madis_transfer_error_fatal* = should an error retrieving observations from MADIS be considered a fatal and stop the execution of RUNWPSPLUS/WREN_RT. If "*1*", then the error will be considered fatal and prevent RUNWPSPLUS/WREN_RT from completing. If "*0*", then the error be considered nonfatal and RUNWPSPLUS will continue executing. For research simulations, the user may want to set this to "*1*" so that simulations are not completed that are unexpectedly missing MADIS observations. However, for more operational applications, the user may want to set this to "*0*" so that even if MADIS data cannot be obtained, RUNWPSPLUS/WREN_RT is still able to complete and produce a forecast.

- *zero_observations_fatal* = should an unexpected lack of observations be considered a fatal error that will cause RUNWPSPLUS/WREN_RT to stop executing. If "*1*", then if the user specifies that observations should be used in some manner and specifies that observations should be ingested from at least one source, and after combining all of the observations over the entire time period there are no observations, then RUNWPSPLUS/WREN_RT will stop executing. If "*0*", then the lack of observations will not prevent RUNWPSPLUS/WREN_RT from continuing. For research simulations the user may want to set this to "*1*" so that simulations are not set up that are unexpectedly missing observations. However, for more operational applications, the user may want to set this to "*0*" so that even if observations cannot be obtained, RUNWPSPLUS/WREN_RT is still able to complete and produce a forecast.

- *littler_filter_option* = what type of filtering should be completed on the observations. The recommended setting for this is "*3*", which indicates that use-reject lists prepared for potential use in Real-Time Mesoscale Analysis (RTMA) are applied to the MADIS mesonet observations. However, these lists may not be available to the user, the user may consider these lists to be too out of date, or the user may wish to be more conservative in which observations are applied. Other options include "*1*" to mark all MADIS mesonet observations as bad, "*-1*" to not do any filtering but update a parameter in the file that appears to be unimportant, or "*0*" to do nothing. Note that this setting does not impact whether or not the littler_filter option to retain a certain percentage of observations is invoked or not as that is controlled via *littler_filter_additional_options*.

- *littler_filter_diagfile* = should a diagnostic file be created by littler_filter. If "*1*", then it will create the file, and if "*0*", then it will not create the file. Note that this file can be quite large.

- *littler_filter_additional_options* = specify any additional command-line parameters that should be passed to littler_filter. See the littler_filter README file for details on potential options. This software can be used to specify use/reject lists independent of the RTMA lists or to keep a certain percentage of observations. If there are no desired additional options then set this to blank (i.e. "").

- *obs_time_padding_backward* = how many seconds prior to the first time at which observations are applied should observations be collected. Two factors contribute to the desirability of obtaining observations prior to first time at which observations are applied. First, for observation nudging, observations are applied as long as their valid time is sufficiently close to the current time being modeled. Thus, observations prior to the start of the observation nudging time period may be recent enough that they can be used in observation nudging. Secondly, it appears that different MADIS observation types may use different conventions regarding what time periods are included in a file valid at a given time. Since a constant named *ONE_HOUR* is available that represents the number of seconds in an hour, it may be simpler to specify this in terms of hours. For example "*3*ONE_HOUR*" may be used to specify that observations should be ingested from files claiming a time up to 3 h prior to the first time at which observations are needed. Note that WREN_RT will modify the default value of this setting found in the RUNWPSPLUS template configuration file. Thus, changing this setting in the RUNWPSPLUS template configuration file will not necessarily change the value that is used on execution; the user must manually override the setting my putting it in the [RUNWPSPLUS] section of the WREN_RT configuration file. WREN_RT will set this to one hour if the user chose to process the observations for verification (via WREN_RT setting *process_obs_for_verification*). Following this, if the user chose to carry out observation nudging (WREN_RT setting *length_of_obs_nudging_minutes* > 0) or use an analysis with observations for the initial conditions and boundary conditions (*use_obs_in_ic_and_bc_analyses*=yes), then WREN_RT will set this to 3 h.

- *obs_time_padding_forward* = how many seconds after the last time at which observations are applied should observations be collected. When

running WREN_RT for purposes of creating verification data sets one may want to set this to a nonzero value in case there are observations close enough to the end of the simulation to be verified against that are contained in a file marked as being after the end of the simulation. Since a constant named *ONE_HOUR* is available that represents the number of seconds in an hour, it may be simpler to specify this in terms of hours. For example "*1*ONE_HOUR*" may be used to specify that observations should be ingested from files claiming a time up to 1 h prior to the first time at which observations are needed. Note that WREN_RT will modify the default value of this setting found in the RUNWPSPLUS template configuration file. Thus, changing this setting in the RUNWPSPLUS template configuration file will not necessarily change the value that is used on execution; the user must manually override the setting by putting it in the [RUNWPSPLUS] section of the WREN_RT configuration file. WREN_RT will set this to 1 h if the user chose to process the observations for verification (via WREN_RT setting *process_obs_for_verification*).

- *use_coarse_grid_obs_nudging_file_for_all_domains* = should an Obsgrid run for the coarsest domain be used to provide observation nudging files (*OBS_DOMAIN**) for each domain. If "*1*", then when Obsgrid is run for the coarsest domain, the observation nudging file created will be applied to all domains. Note that WRF itself will only use observations within the current domain and so there is no risk of observations outside a domain being assimilated on that domain. If "*0*", then Obsgrid will be run for each domain individually to create domain specific observation nudging files.

- *sstrtg_retrieve_error_fatal* = should errors retrieving the Real-Time Global Sea Surface Temperature (RTG SST) data be considered fatal. If the user chooses to use RTG SST data and attempts to download these data are unsuccessful, then if "*1*", then this will cause RUNWPSPLUS/WREN_RT to stop execution prematurely, whereas if "*0*", then this error will not cause RUNWPSPLUS/WREN_RT to stop execution prematurely. Setting this to "*1*" may be preferable for research simulations since the user will likely want to know if a given simulation unexpectedly does not have access to RTG SST data. Setting this to "*0*" may be preferable for more operational simulations since the user may want to complete the simulation with degraded SST data rather have no WRF forecast at all.

- *snodas_retrieve_error_fatal* = should errors retrieving the Snow Data Assimilation System (SNODAS) snow data be considered fatal. If the user chooses to use SNODAS snow data and attempts to download these data

are unsuccessful, then if "*1*", then this will cause RUNWPSPLUS/ WREN_RT to stop execution prematurely, whereas if "*0*", then this error will not cause RUNWPSPLUS/WREN_RT to stop execution prematurely. Setting this to "*1*" may be preferable for research simulations since the user will likely want to know if a given simulation unexpectedly does not have access to SNODAS snow data. Setting this to "*0*" may be preferable for more operational simulations since the user may want to complete the simulation with degraded snow data rather have no WRF forecast at all.

- *nesdis_retrieve_error_fatal* = should errors retrieving the National Environmental Satellite, Data, and Information Service (NESDIS) snow data be considered fatal. If the user chooses to use NESDIS snow data and attempts to download these data are unsuccessful, then if "*1*", then this will cause RUNWPSPLUS/WREN_RT to stop execution prematurely, whereas if "*0*", then this error will not cause RUNWPSPLUS/WREN_RT to stop execution prematurely. Setting this to "*1*" may be preferable for research simulations since the user will likely want to know if a given simulation unexpectedly does not have access to NESDIS snow data. Setting this to "*0*" may be preferable for more operational simulations since the user may want to complete the simulation with degraded snow data rather have no WRF forecast at all.

- *wrf_chem* = should the preprocessors be run in such a way that the files produced will be able to be used in a dust-only WRF-Chem simulation. If "*1*", then an alternative GEOGRID.TBL file is used that has dust erodibility fields (the user must have the geographic data sets erod, clayfrac_5m, and sandfrac_5m). Assuming that the proper fields are available, besides Geogrid taking longer to run, there is no reason that this option cannot be enabled even for non-WRF-Chem simulations. If "*0*", then the dust erodibility fields are not processed and the resultant files are not appropriate for running a dust-only WRF-Chem simulation.

- *deleted_unwanted_files_level* = how many of the output files should be deleted. If "*0*", then none of the output files will be deleted. If "*5*", then a set of the larger files that are unlikely to be helpful (as long as the user keeps the raw data needed to do reruns) are deleted. Using "*5*" reduces the amount of disk space taken by the output, but "*0*" will allow for enhanced troubleshooting.

A-2.2 WPS Settings

The following divider is present between the end of the RUNWPSPLUS-specific settings and those used by WPS:

###
#BEGIN SETTING OF NAMELIST.WPS OPTIONS

The WPS settings are placed in the WPS namelist (namelist.wps) by RUNWPSPLUS. For more details on the WPS settings, see the WPS chapter in the WRF User's Guide (http://www2.mmm.ucar.edu/wrf/users/docs/user_guide _V3.9/users_guide_chap3.html).

One setting the user may want to be aware of is *constants_name*. WREN_RT assumes that the user will want to use RTG SST, SNODAS snow, and NESDIS snow if they are available. Thus, *constants_name* is set as follows:

constants_name = "'SST_RTG','SNOW_CGM','SNOWC_NESDIS', 'SNOW_SNODAS','SNOWH_SNODAS'"

In this list, items later in the list take precedence. Also, note that RUNWPSPLUS will remove items from this list if it is unable to obtain them. However, if one does not want to use one of these even if they are available, one would want to add an updated value for *constants_name* without the unwanted fields in the WREN_RT configuration file in the [RUNWPSPLUS] section so that it can override the default setting.

The following WPS settings in the RUNWPSPLUS configuration file are set by WREN_RT:

geog_data_path, *max_dom*, *start_date*, *end_date*, *interval_seconds*, *parent_id*, *parent_grid_ratio*, *i_parent_start*, *j_parent_start*, *e_we*, *e_sn*, *geog_data_res*, *dx*, *dy*, *map_proj*, *ref_lat*, *ref_lon*, *truelat1*, *truelat2*, *stand_lon*, *geog_data_path*, *new_plvl_hires_ungrib*

Since the settings just listed are set by WREN_RT, the user should be aware that if these settings are manually set in the [RUNWPSPLUS] section of the WREN_RT configuration file, the value that is manually set will override the value the WREN_RT determines should have been used for the setting. However, changing how the settings just listed are set in the template RUNWPSPLUS configuration file will not be effective, since WREN_RT will override whatever the template file has for these specific variables.

A-2.3 Obsgrid Settings

A divider similar to that at the beginning of the WPS settings is found at the beginning of the Obsgrid settings:

###
#BEGIN SETTING OF OBSGRID NAMELIST OPTIONS

The Obsgrid settings are placed in the Obsgrid namelist (*namelist.oa*) by RUNWPSPLUS. The Obsgrid settings in the RUNWPSPLUS template configuration file are named with the standard Obsgrid namelist setting name but with the prefix "obsgrid_nl_". For more details on the Obsgrid settings, see the Obsgrid chapter in the WRF User's Guide (http://www2.mmm.ucar.edu/wrf/users/docs/user_guide_V3.9/users_guide_chap7.html#obsNamelist).

The following Obsgrid settings in the RUNWPSPLUS configuration file are set by WREN_RT:

obsgrid_nl_interval, *obsgrid_nl_trim_domain*, *obsgrid_nl_trim_value_each_dom*, *obsgrid_nl_oa_psfc_each_dom*

Since the Obsgrid settings just listed are set by WREN_RT, the user should be aware that if these settings are manually set in the [RUNWPSPLUS] section of the WREN_RT configuration file, the value that is manually set will override the value WREN_RT determines should have been used for the setting. However, changing how the settings just listed are set in the template RUNWPSPLUS configuration file will not be effective, since WREN_RT will override whatever the template file has for these specific variables.

A-3 [WRF] Section of WREN_RT Configuration File

The [WRF] section of the WREN_RT configuration file is used to manually override the settings in the WRF template configuration file (namelist.input_wren_template). This section may be left blank if no such manual overrides are desired. The settings placed in this section will override the settings found in the WRF template configuration file namelist.input_wren_template and will also override any settings WREN_RT sets in this file (WREN_RT uses the settings in the [WREN_RT] section to set some of the WRF settings). The user should be careful in overriding WRF settings, since it also prevents WREN_RT from ensuring that WRF settings are consistent with one another. For more information of WRF settings, see the WRF model section of the WRF User's Guide (http://www2.mmm.ucar.edu/wrf/users/docs/user_guide_V3.9/users_guide_chap5.htm).

Note that the [WRF] section of the WREN_RT configuration cannot be used to override settings that are not present in the WRF template configuration file. This is because WREN_RT would not know which section of the namelist the variable should be placed in. Thus, if one wishes to set a WRF setting not present in the WRF template configuration file, one must first add that setting to the WRF template configuration file, and then it can be set in the [WRF] section of the WREN_RT configuration file.

The following WRF settings are set by WREN_RT:

run_days, *run_hours*, *run_minutes*, *run_seconds*, *start_year*, *start_month*, *start_day*, *start_hour*, *start_minute*, *start_second*, *end_year*, *end_month*, *end_day*, *end_hour*, *end_minute*, *end_second*, *interval_seconds*, *history_interval_s*, *fine_input_stream*, *iofields_filename*, *ignore_iofields_warning*, *io_form_auxinput16*, *auxinput16_interval*, *auxinput16_inname*, *p_top_requested*, *eta_levels*, *time_step*, *time_step_fract_num*, *time_step_fract_den*, *max_dom*, *s_we*, *e_we*, *s_sn*, *e_sn*, *s_vert*, *e_vert*, *num_metgrid_levels*, *num_metgrid_soil_levels*, *num_soil_cat*, *num_land_cat*, *dx*, *dy*, *grid_id*, *parent_id*, *i_parent_start*, *j_parent_start*, *parent_grid_ratio*, *parent_time_step_ratio*, *mp_physics*, *ra_lw_physics*, *ra_sw_physics*, *bl_pbl_physics*, *sf_sfclay_physics*, *sf_surface_physics*, *cu_physics*, *cu_diag*, *cu_rad_feedback*, *cudt*, *cugd_avedx_perdomain*, *ishallow*, *radt*, *slope_rad*, *topo_shading*, *obs_nudge_opt*, *fdda_start*, *fdda_end*, *obs_dtramp*, *obs_rinxy*

Since the settings just listed are set by WREN_RT, making changes in the template WRF namelist will not be effective in changing which setting is actually used by WREN_RT when WRF is executed. The user will need to manually override the value WREN_RT sets by setting it in the [WRF] section of the WREN_RT configuration file.

List of Symbols, Abbreviations, and Acronyms

ACARS	Aircraft Communications Addressing and Reporting System
ACARSP	ACARS profile
API	application programming interface
ASOS	Automated Surface Observing System
AWOS	Automated Weather Observing System
CAP	Cooperative Agency Profilers
CGM	coarse-grid model
CORINE	Coordinate Information on the Environment
FASDAS	Flux-Adjusting Surface Data Assimilation System
GALWEM	Global Air-Land Weather Exploitation Model
GFS	Global Forecasting System
GMTED2010	Global Multi-resolution Terrain Elevation 2010
GRIB	Gridded Binary
HRRR	High-Resolution Rapid Refresh
IGBP MODIS	International Geosphere-Biosphere Programme Moderate Resolution Imaging Spectroradiometer
IMS	Interactive Multisensor Snow and Ice Mapping System
I/O	input/output
MADIS	Meteorological Assimilation Data Ingest System
MET	Model Evaluation Tools
METAR	Météorologique Aviation Régulière (Aviation Routine Weather Report)
MM5	Fifth-generation Pennsylvania State University–National Center for Atmospheric Research Mesoscale Model
MPI	Message Passing Interface
MYJ	Mellor-Yamada-Janjić

NAM	North American Model
NCAR	National Center for Atmospheric Research
NCEP	National Centers for Environmental Prediction
netCDF	Network Common Data Form
NESDIS	National Environmental Satellite, Data, and Information Service
NLCD2006	National Land Cover Database 2006
NOAA	National Oceanic and Atmospheric Administration
NOHRSC	National Operational Hydrologic Remote Sensing Center
NSIDC	National Snow and Ice Data Center
PBL	planetary boundary layer
ROI	radius of influence
RTG SST	Real-Time Global Sea Surface Temperature
RTMA	Real-Time Mesoscale Analysis
SAO	surface airways observation
SNODAS	Snow Data Assimilation System
SRTM	Shuttle Radar Topography Mission
SST	sea surface temperature
TAMDAR	Tropospheric Airborne Meteorological Data Reporting
TKE	turbulent kinetic energy
USGS	US Geological Service
UTC	Coordinated Universal Time
WGS84	World Geodetic System 1984
WPS	WRF Preprocessing System
WREN_RT	Weather Running Estimate – Nowcast Real Time
WRF	Weather Research and Forecasting model
WRFEE	WRF End-to-End

WRF-ARW	Advanced Research version of the Weather Research and Forecasting model

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIR ARL IMAL HRA RECORDS MGMT RDRL DCL TECH LIB
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
2 (PDF)	ARL RDRL CIE M B REEN L DAWSON