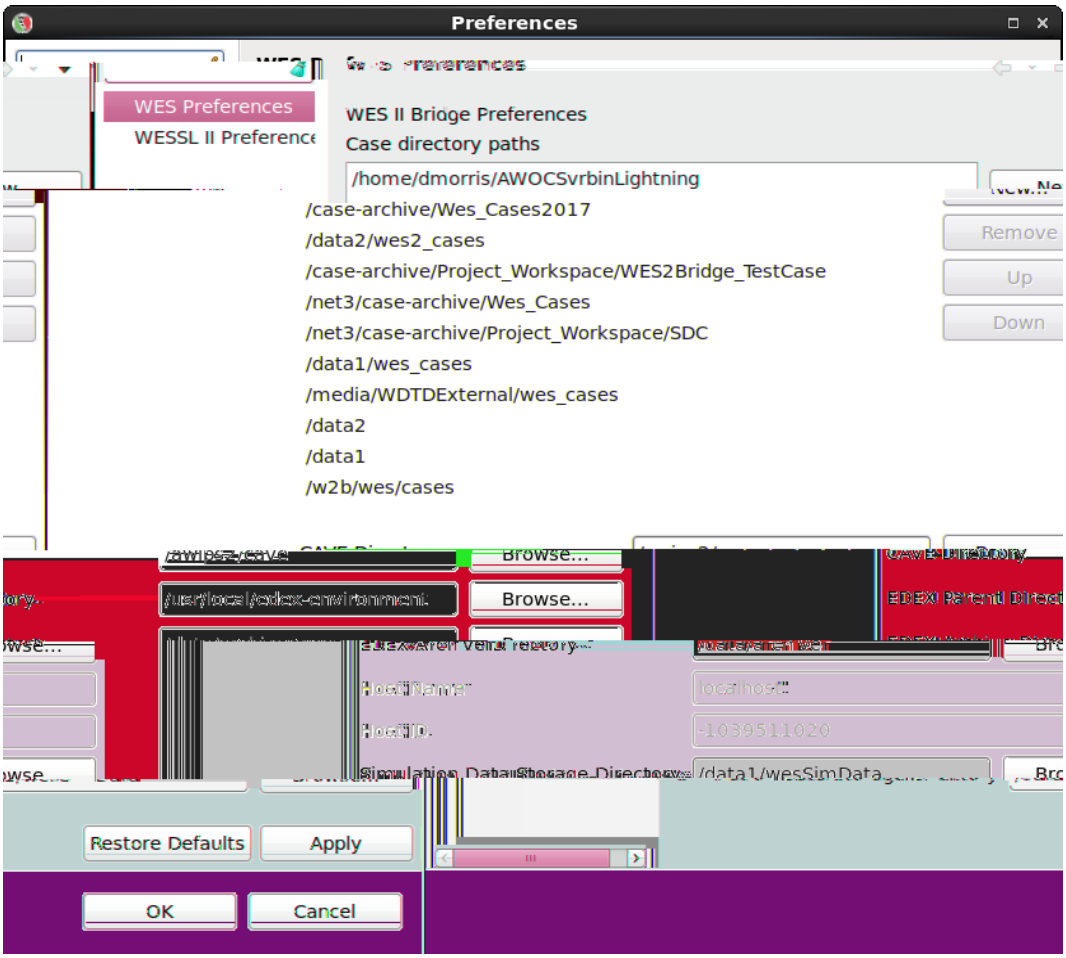
Release Notes

WES-2 Bridge Version 17.1.1

October 2017

1. **Centralized WES-2 Bridge Case Path Preferences .**
   In previous versions, each user had a preferences file that contained the locations where WES-2 Bridge would look for data cases. In 17.1.1, it looks for a file named gov.noaa.wdtb.wes.preferences.prefs in /w2b/data/.metadata/.plugins/org.eclipse.core.runtime/.settings

   It is still edited using the WES Preferences GUI accessed with the Window ►Preferences menu.



2. **User Account Creation and Postgres Database Authentication**

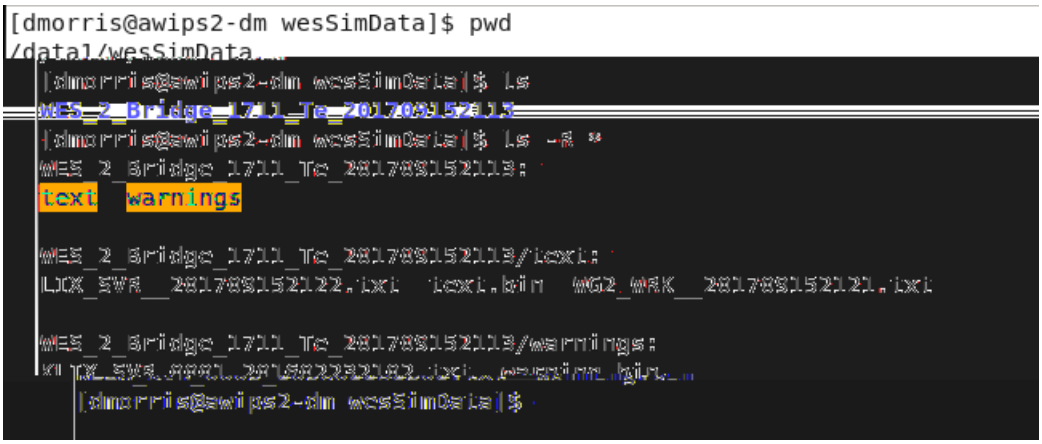
Changes in the AWIPS Build of 17.1.1 required WES-2 Bridge to implement individual user authentication for the Postgres database. This is done by creating user key and certificate files in the user's home directory. In order to facilitate this, there is a new script called **/w2b/util/newWesUser.csh.** When you need to add a new user, this script will create the new user account as well as create all the files needed to authenticate this user with the database.

For existing users, the AWIPS-2 installation that accompanies WES-2 Bridge automatically creates files needed based on the accounts and home directories already on the WES-2 Bridge machine.

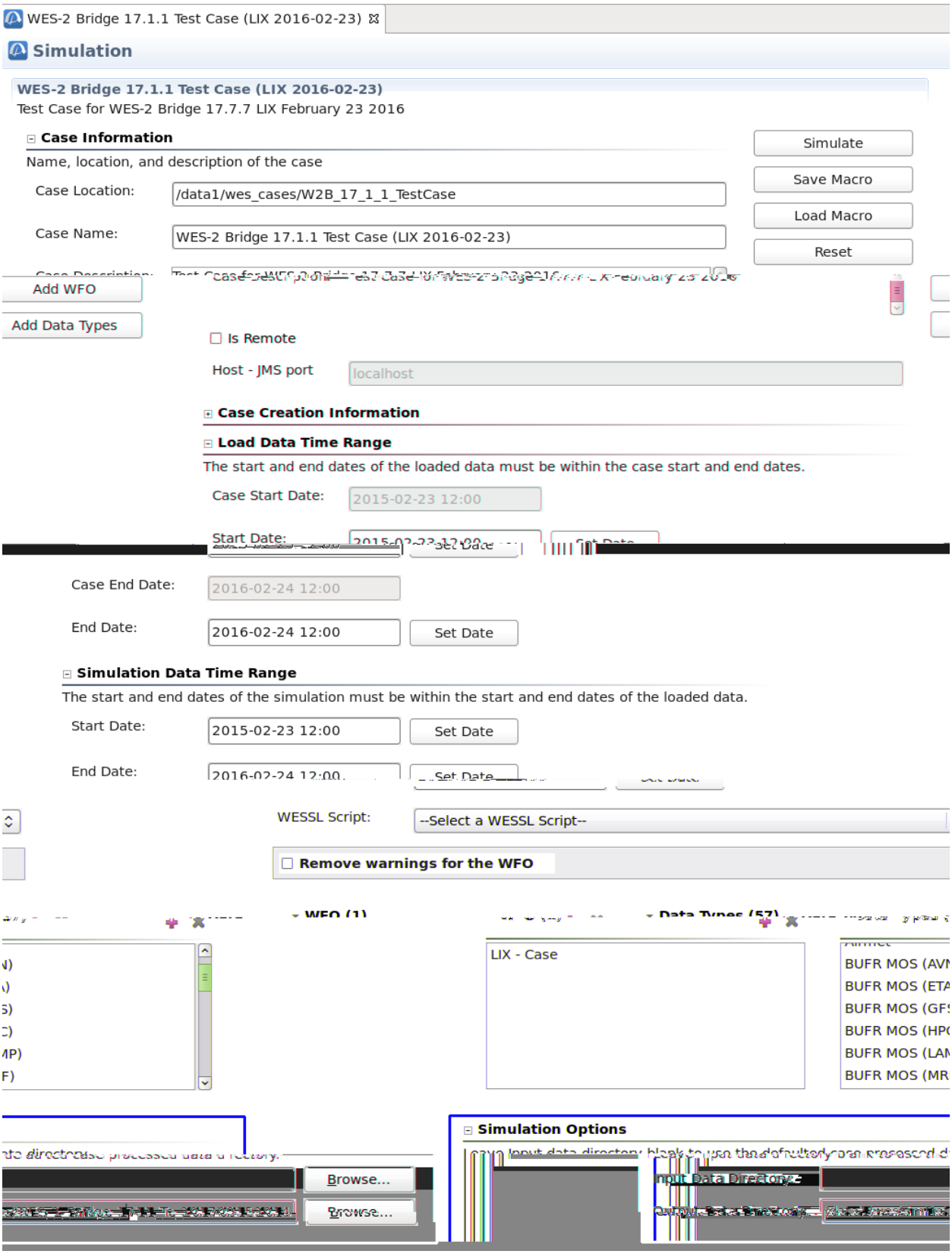
3. **Simulation Input and Output Directories for Trainee Warnings and Forecast Products**

Beginning in Build 17.1.1, warnings and forecast text products generated during a simulation are saved and are more easily viewable. WES-2 Bridge had saved warnings previously but in a binary format; AWIPS also saved warnings in /data_store but with an unintuitive date/time location. However, with Build 17.1.1, when a simulation is finished, the WES-2 Bridge software saves both text and binary versions of text products and warnings in an output directory. This directory is specified in the Simulation Editor, where an input directory can also be specified. These are located at the bottom of the screen. These directories also can contain the output Fcst and/or Official databases from a simulation that ran GFE (see below).

As stated above, the output directory is specified in the Simulation Editor. If the user does not specify a directory, the WES-2 Bridge creates an intelligent default based on an output directory of /data1/wesSimData, followed by an identifier that comes from the first 20 characters of the name of the case (in the example below, "WES_2_Bridge_1711_Te" from the WES-2 Bridge 17.1.1 test case) along with the system date and time of when the simulation was run (in YYYYMMDDHHMM format as in 201709152014). In this manner every simulation will save off the warnings and forecasts created during that simulation. In the screenshot below, the output directory for a simulation is /data1/wesSimData/WES_2_Bridge_1777_Te_201709152113 (the simulation was run on September 15, 2017 at 21:13). The output from this particular simulation was a Severe Thunderstorm Warning which was saved both in its text format and in the warning format. Both have .txt files which are readable by any text viewer and .bin files which can be reloaded into AWIPS by WES-2 Bridge. The .bin files contain the extra database metadata required for proper timing and indexing by both WES-2 Bridge and AWIPS.



The input directory provides a way to initialize a simulation with a pre-created warning or forecast product. The simulation facilitator would create a warning and/or forecast and then save it off. Then those particular products will appear at the beginning of a simulation, if the simulation's input directory is pointed to the place where the warning or forecast was saved off. If this location is saved as part of a macro, the warnings or the forecasts located in this directory will be used each time the particular simulation is run using that macro.


WES-2 Bridge 17.1.1 Test Case (LIX 2016-02-23)
Simulation
WES-2 Bridge 17.1.1 Test Case (LIX 2016-02-23)
Test Case for WES-2 Bridge 17.7.7 LIX February 23 2016
Case Information
Name, location, and description of the case
Case Location:
/data1/wes_cases/W2B_17_1_1_TestCase
Case Name:
WES-2 Bridge 17.1.1 Test Case (LIX 2016-02-23)
Simulate
Save Macro
Load Macro
Reset
Add WFO
Add Data Types
Is Remote
Host - JMS port
localhost
Case Creation Information
Load Data Time Range
The start and end dates of the loaded data must be within the case start and end dates.
Case Start Date:
2015-02-23 12:00
Case End Date:
2016-02-24 12:00
End Date:
2016-02-24 12:00
Set Date
Simulation Data Time Range
The start and end dates of the simulation must be within the start and end dates of the loaded data.
Start Date:
2015-02-23 12:00
Set Date
End Date:
WESSL Script:
--Select a WESSL Script--
Remove warnings for the WFO
LIX - Case
BUFR MOS (AV
BUFR MOS (ETA
BUFR MOS (GFS
BUFR MOS (HP
BUFR MOS (LAM
BUFR MOS (MR
Browse...
Browse...
Simulation Options

4. **Graphical Forecast Editor Support**

WES-2 Bridge supports basic GFE functionality. After editing Fcst grids, WES-2 Bridge can save the Fcst and Official databases as well as any products generated using Formatters. The Fcst and Official databases can be loaded back either to initialize a new simulation or to review the database for evaluation purposes.

The GFE functionality works together with the input and output directories. When the simulation is finished, the Fcst and Official GFE databases are saved in the Output directory. If a Fcst or Official database is in the Input Directory, WES-2 Bridge will load those two databases back into AWIPS. These can be used to either initialize a new simulation, or to review the grids that were edited during a previous simulation.

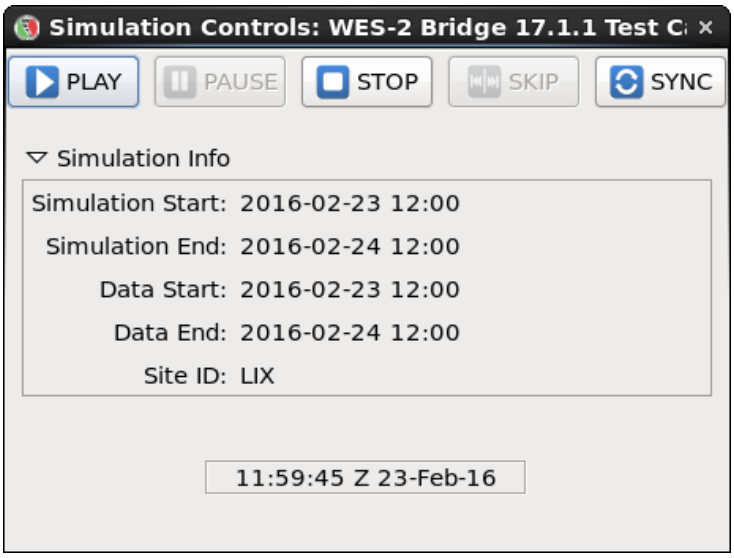
Note: Sometimes the output Fcst database can be generated by the awips user with 755 permissions. WES-2 Bridge needs the permissions to be 775. If the Fcst database has the wrong permissions, there can be problems with initializing a simulation with a previously saved Fcst database. If this happens, someone with root priviledges will need to change permissions similar to this:

`chmod –R g+w /usr/local/edex-environment/EDEX_XX/edex/data/hdf5/gfe/LLL/Fcst`,

where XX is the instance number and LLL is the site's WFO ID.

There can be some issues in using GFE with WES. The ifpServer has been observed to occasionally delete some grids which can result in some missing data. This appears to be enforcing a constraint on the number of available versions of a particular model. This can be a bit tricky for an archived case. In addition, if certain SmartTools and procedures developed for GFE assume some particular version of SmartInits or model data, then it's possible or likely that using them with some archived GFE can be problematic. To help address part of these issues, WES-2 Bridge uses a special serverConfig.py file so that FFE will not purge models and not run smartInits.

Due to having to reinitialize the GFE server to begin a simulation, there are some changes in how simulations begin as the timing of when CAVE starts up during a simulation. When a simulation starts with GFE, WES-2 Bridge has to activate GFE. Also during this time, it is common for the WES-2 Bridge Simulation Controls to display before CAVE starts. Once you successfully launch CAVE, pressing the Play button syncs the time in CAVE and the starts the simulation.



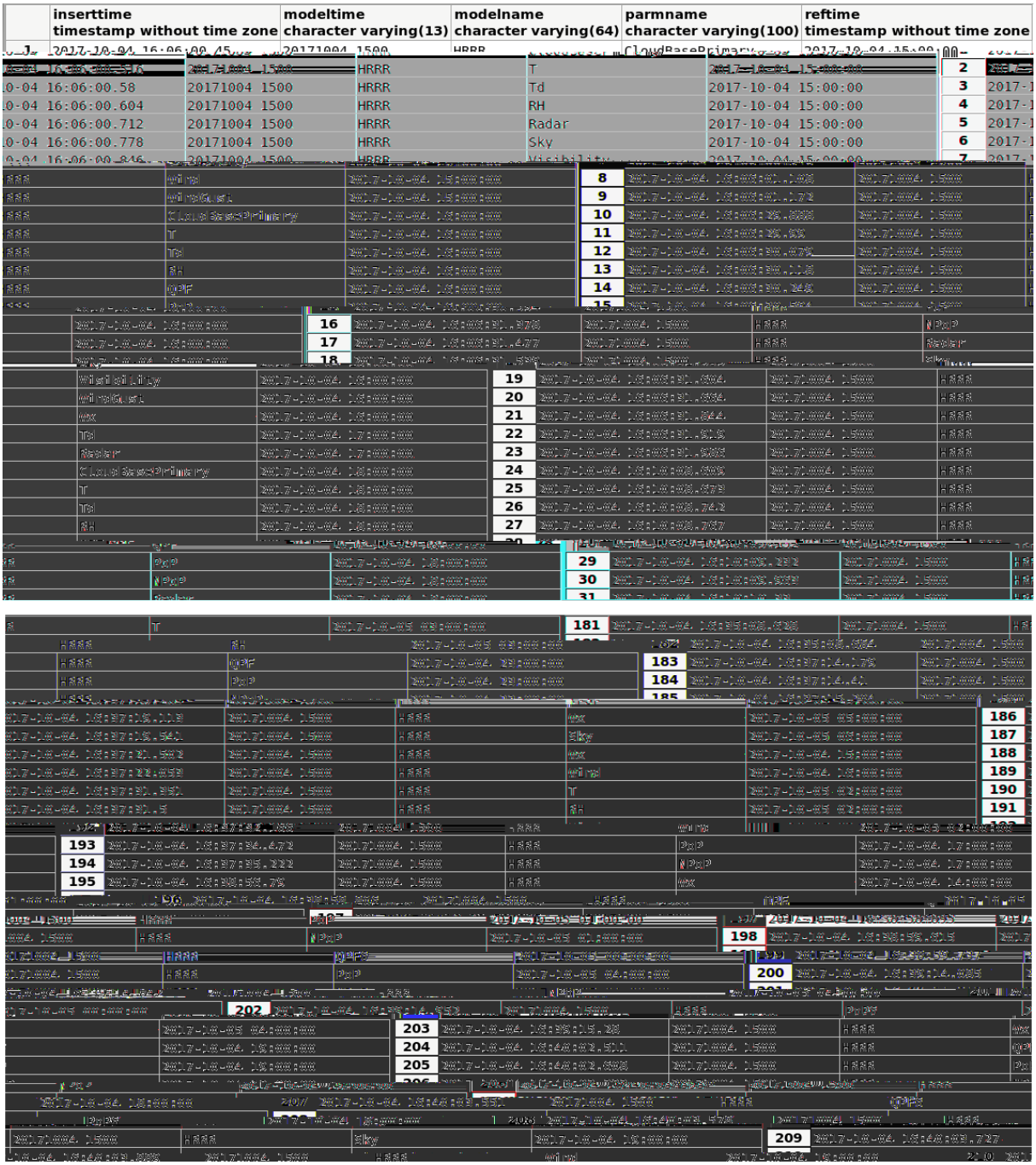
There is a final best practice that deserves mention. It is a good practice to close out of a GFE perspective prior to closing the CAVE window. This is because CAVE remembers its last state on a per-user basis. If the next or case review that is attempted doesn't start GFE correctly, then it's possible for CAVE to crash on startup. There are some behind-the-scenes efforts to correct for this issue, but closing out GFE prior to existing CAVE avoids the issue entirely.

Note on Grid Availability in GFE during a simulation:

In this 17.1.1 version of WES-2 Bridge, grids will not update during a simulation. That is, the only grids available during a simulation are the ones that would have been available at the beginning of the simulation. For example, in the WES-2 Bridge test case, the HRRR for 1700 UTC would be scheduled to become available at 1826 UTC (see the inserttime values in the figure). However, the notification mechanism between WES-2 Bridge and GFE currently fails to make the grids available. In this situation, the 17Z HRRR may appear as a choice in the Weather Element Browser or in the "Copy Available Grids From…" dialog boxes, but attempting to load the model will not produce any usable grids. The work around for this is to start a new simulation any time after 1826 UTC and then the 17Z HRRR would be available. Because it's possible to propagate a Fcst database forward from another simulation, this issue likely becomes an inconvenience; WDTD is working to fix this issue in an upcoming build.

	inserttime timestamp without time zone	modeltime character varying(13)	modelname character varying(64)
1	2016-02-23 14:26:00	20160223 1300	HRRR
2	2016-02-23 15:26:00	20160223 1400	HRRR
3	2016-02-23 16:55:00	20160223 1500	HRRR
4	2016-02-23 17:20:00	20160223 1600	HRRR
5	2016-02-23 18:26:00	20160223 1700	HRRR
6	2016-02-23 19:26:00	20160223 1800	HRRR
7	2016-02-23 20:26:00	20160223 1900	HRRR
8	2016-02-23 21:26:00	20160223 2000	HRRR
9	2016-02-23 22:26:00	20160223 2100	HRRR
10	2016-02-23 23:26:00	20160223 2200	HRRR

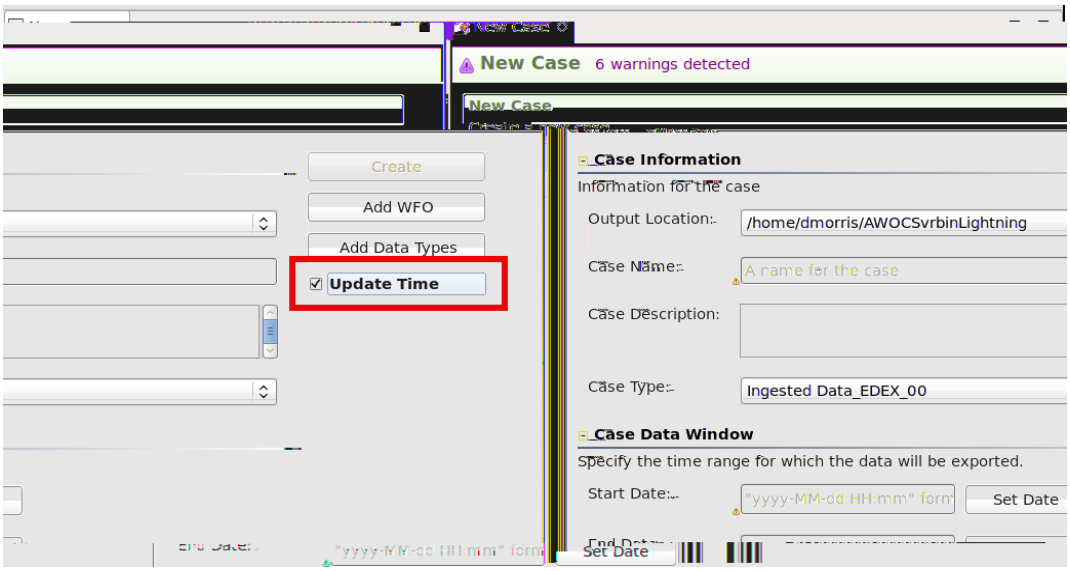
Also note that in reality, a certain model in GFE becomes available over a span of time rather than at a given instant. In the figure below, the various grids that comprised the 15Z HRRR for October 4, 2017, became available in GFE between 1606 UTC and 1640 UTC (there is a record for each grid based on parameter and valid time; valid time is the reftime in this example so that in line 25, this record represents the 3-hour forecast temperature grid from the 15Z HRRR, valid at 1800 UTC, so that each record corresponds to a grid box in the Grid Manager of GFE).



5. **Behind the Scenes Improvements**
    a. Localization checking. WES-2 Bridge checks for localizations that are compatible with Build 17.1.1 and attempts to move directories to proper locations if it finds an incompatible older localization.
    b. WESSL-2 LSR importer improvements and stability
    c. Additional Utility Scripts (new user creation, updating localization from AWIPS, updating case localizations)

6. **Gridded Data Delays for Reprocessed Raw Data (gridDelay.xml)**

To better support GFE, WES-2 Bridge implemented a method of applying more realistic delays to gridded data that have been reprocessed from Raw Data. These delays will be applied by using the New Case option and ensuring the "Update Time" checkbox is checked.



The delays are applied on a model-by-model basis using a configurable file located at /w2b/wes/config/gridDelay.xml. Users may want to modify this file in order to use reprocessed data that includes local or experimental models.

The gridDelay.xml file is broken up into two sections

- gridDelayRule entries
- gridCategory entries

The gridDelayRule section defines specific fields from the grid and grid_info database tables that are used to determine a delay for a specific model. Many models are disseminated and processed in order of their forecast hour (for example, GFS and NAM forecast hour 03 is available well before forecast hour 60). Other grids from models become available at almost the same time. Still others have delays based on a list of parameters. Each rule then defines a set of database constraints.

The gridCategory section defines a set of models (a "category") that obey a certain rule with the same set of delays. Each gridCategory defines a datasetid which is a list of model names as defined in the grid_info metadata database table. Then the rule is applied by matching one or more keys to each database constraint defined in the gridDelayRule corresponding to the given gridCategory. Each set of one or more keys then defines a delay value which is the number of seconds to be added to the reftime field (the reference time of the grid) to determine the delay (as stored in the inserttime field of the model).

The section of the gridDelay.xml file that has the HRRR, LAPS, and MRMS data illustrates how the gridCategory and gridDelayRule entries work together to define the delays.

The HRRR model (see the HRRR gridCategory) uses the "forecastTime" gridRule. The forecastTime gridDelayRule has a metadataMap section which defines the database field "forecasttime" as the key to match to determine the grid delay value.

```
<metadataMap>
        <mapping key="forecasttime">
                <constraint constraintType = "EQUALS"/>
        </mapping>
</metadataMap>
```

The gridCategory that contains the HRRR gives the forecast hour (0,1,2,3, etc) and the corresponding delays in seconds. Therefore the WES-2 Bridge software finds all database records where the datasetid matches "HRRR" and then applies a delay of 3273 seconds (54 minutes and 33 seconds) for the 0 hour HRRR forecast, and a delay of 3374 seconds (56 minutes 14 seconds) for the 1 hour forecast, and so on. These values were obtained by averaging metadata from several actual runs from an operational dataset. This means, for example, that the 1 hour forecast from the 12Z run would become available in a WES-2 Bridge simulation one second after 12:56:14 (because WES-2 Bridge runs on a 15 second heartbeat).

```
<gridCategory id="HRRR">
        <description>HRRR                </description>
        <gridRule>forecastTime</gridRule>
        <datasetid>
                <constraint constraintType = "EQUALS" constraintValue="HRRR"/>
        </datasetid>

        <delay key="0">3273</delay>
        <delay key="1">3374</delay>
        <delay key="2">3470</delay>
        <delay key="3">3565</delay>
        <delay key="4">3675</delay>
        <delay key="5">3772</delay>
        <delay key="6">3850</delay>
        <delay key="7">3961</delay>
        <delay key="8">4040</delay>
        <delay key="9">4148</delay>
        <delay key="10">4240</delay>
        <delay key="11">4349</delay>
        <delay key="12">4444</delay>
        <delay key="13">4539</delay>
        <delay key="14">4615</delay>
        <delay key="15">4756</delay>
        <delay key="16">4849</delay>
        <delay key="17">4929</delay>
        <delay key="18">4991</delay>

</gridCategory>
```

The LAPS gridCategory references the “name” gridDelayRule, as shown below.

```
<gridDelayRule id="name">
       <description>
                </description>
                <metadataMap/>

           </gridDelayRule>
```

```
 <gridCategory id="LAPS">

        <description>LAPS</description>

        <gridRule>name</gridRule>

        <datasetid>
                <constraint constraintType = "EQUALS" constraintValue="LAPS"/>
        </datasetid>

        <delay>
                <value>1427</value>
        </delay>
 </gridCategory>
```

This is the simplest rule because a single delay (in this case 1427 seconds, or 23 minutes 47 seconds, is applied to every LAPS grid. This is because LAPS is generated locally at each WFO on a PX machine via a cron job, rather than being centrally produced and disseminated. Thus, the delays are constant for the given grid, and there is no database metadata required other than the datasetid to identify the model name, and therefore to obtain the required delay.

MRMS grids are analyses, rather than forecasts. It turns out that certain sets of products in the MRMS suite of products are generated on different schedules. For example, some that come from the basic reflectivity cube have much shorter delays than some of the QPE products, and there are products that fall in between. To handle this type of situation, a "parmList" gridDelayRule was developed.

```
<gridDelayRule id="parmList">
        <description>
                This works for part of ETA218 and ETA242
                </description>

                <metadataMap>
                        <mapping key="parameter_abbreviation">
                                <constraint constraintType = "SIMILAR_TO"/>
                        </mapping>
                </metadataMap>

        </gridDelayRule>
```

The parmList rule applies delays based only on the parameter_abbreviation field in the grid_info table. The MRMS application of the parmList rule is shown in the MRMS gridCategory:

```
<gridCategory id="MRMS">

        <description> This is MultiRadar MultiSensor.
        </description>

        <gridRule>parmList</gridRule>

        <datasetid>
                <constraint constraintType = "IN" constraintValue="MRMS_1000,MRMS_0500"/>
        </datasetid>

        <!--Special QPE Products -->
        <delay key="GaugeCorr%">5757</delay>
        <delay key="MountainMapper%">5263</delay>
        <delay key="WarmRain%">4730</delay>

        <!--Lightning  Density Products -->
        <delay key="LightningDensity%">171</delay>

        <!--Lightning  Probability Products -->
        <delay key="LightningProb%">239</delay>

        <!--Reflectivity Volume Products (MESH/VIL/QC, etc) -->
        <delay key="MESH%,MergedReflectivity%,MRMSVIL,VII,MergedBase%">105</delay>

        <!--Precip 1 HR QPE-->
        <delay key="PrecipRate,PrecipType,RadarOnlyQPE01H">148</delay>

        <!--QPEs-->
        <delay key="RadarOnlyQPE03H,RadarOnlyQPE06H,RadarOnlyQPE12H,RadarOnlyQPE24H,RadarOnlyQPE48H,RadarOnlyQPE72H">215</delay>

        <!--Basic Reflectivity Products-->
        <delay key="EchoTop%,H50%,H60%,Height%,LLComposite%,Reflectivity%,RadarQuality%,Seamless%">143</delay>

        <!--Rotation Tracks-->
        <delay key="RotationTrack%">102</delay>

        <!--need to add FLASH -->

</gridCategory>
```

Based on these definitions, the MESH products have 105 second (1 minute 45 second) delay, Echo Tops have 143 second delay and most QPE products have a 215 delay. However, some QPE products (like Gauge Corrected QPE) have much longer delays (like 5757 seconds, or 1 hour 35 minute and 57 seconds).

One final example illustrates a more complicated rule that uses more than one database field and constraint to determine the delay. This example uses the “forecastTimeRefHHCategory” gridRule. This was developed for the RAP (13km, also known as RUC130) as well as potentially other models whose arrival times over the SBN depend on the forecast hour and also the model run time. It takes longer for the RAP to be generated and disseminated for the 00Z and 12Z runs than the other hourly rules (perhaps due to competition of computational resources and bandwidth for the 00Z and 12 runs of other models).

The forecastTimeRefHHCategory gridDelayRule

```
<gridDelayRule id="forecastTimeRefHHCategory">
        <description>
                This rule is for grids (like RUC) whose delays are based on the forecast hour but have different delays
                for different forecast hours (00/12 Z are later than other forecast times)

        </description>
        <metadataMap>
                <mapping key="reftime">
                        <constraint constraintType = "IN" constraintTimeFormat="HH" />
                </mapping>

                <mapping key="forecasttime">
                        <constraint constraintType = "EQUALS"/>
                </mapping>
        </metadataMap>

</gridDelayRule>
```

uses both the refTime and the forecastTime in the grid database table as constraints. The refTime has a timeFormat of “HH” applied to it so that a two-digit value for the model run time is generated (as in “00”, “01”, “02”, etc). The constraintType of “IN” means that the database set notation is used to differentiate the 00 and 12Z runs as one set and the remaining runs as another set as shown in these excerpts of the corresponding gridCategory entry:

```xml
<gridCategory id="RUC130">

        <description>13km RAP/RUC</description>
        <gridRule>forecastTimeRefHHCategory</gridRule>

        <datasetid>
                <constraint constraintType = "EQUALS" constraintValue="RUC130"/>
        </datasetid>

        <delay>
                <key id="tocharREFHH">00,12</key>
                <key id="fcsttime">0</key>
                <value>5227</value>
        </delay>
        <delay>
                <key id="tocharREFHH">00,12</key>
                <key id="fcsttime">1</key>
                <value>5042</value>
        </delay>
        <delay>
                <key id="tocharREFHH">00,12</key>
                <key id="fcsttime">2</key>
                <value>5096</value>
        </delay>
        <delay>
                <key id="tocharREFHH">00,12</key>
                <key id="fcsttime">3</key>
                <value>5120</value>
        </delay>
        <delay>
                <key id="tocharREFHH">00,12</key>
                <key id="fcsttime">4</key>
                <value>5154</value>
        </delay>
        <delay>
                <key id="tocharREFHH">00,12</key>
                <key id="fcsttime">5</key>
                <value>5187</value>
        </delay>
        <delay>
                <key id="tocharREFHH">00,12</key>
                <key id="fcsttime">6</key>
                <value>5208</value>
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
        <delay>
                <key id="tocharREFHH">00,12</key>
                <key id="fcsttime">21</key>
                <value>5591</value>
        </delay>
        <delay>
                <key id="tocharREFHH">01,02,03,04,05,06,07,08,09,10,11,13,14,15,16,17,18,19,20,21,22,23</key>
                <key id="fcsttime">0</key>
                <value>3322</value>
        </delay>
        <delay>
                <key id="tocharREFHH">01,02,03,04,05,06,07,08,09,10,11,13,14,15,16,17,18,19,20,21,22,23</key>
                <key id="fcsttime">1</key>
                <value>3134</value>
        </delay>
```

Consequently, the one-hour forecast of the RAP13 model has a 5042 second delay applied for the 00Z and 12Z runds and a 3134 second delay for the remaining hourly runs. Again, these values were determined by computing average delays from a realtime database from a sample dataset.

The bottom of this file has an experimental section for new models and is the suggested location to place any local edits.

Here is a final note about the gridDelay rule definitions. There are a fixed set of contraintTypes, and here is a listing of them: EQUALS, NOT_EQUALS, GREATER_THAN, GREATER_THAN_EQUALS, LESS_THAN, LESS_THAN_EQUALS, BETWEEN, IN, LIKE, ILIKE, ISNULL, ISNOTNULL, NOT_IN, SIMILAR_TO and NOT_SIMILAR_TO The SIMILAR_TO and NOT_SIMILAR_TO constraints were added in WES-2 Bridge; the others are part of the AWIPS baseline code. The SIMILAR_TO and NOT_SIMILAR_TO constraints allow some wildcard characters in some of the keys (see the MRMS example above).