WESTERN REGION TECHNICAL ATTACHMENT
NO. 97-23
JULY 8, 1997

PROCEDURES FOR ENSURING RELIABLE DATA FLOW

Keith W. Meier and Jonathan Van Ausdall - NWSO Billings, MT

Overview

Forecast offices within the Western Region have been receiving gridded model data, digital satellite imagery, and other forms of model/observed data via the Western Region's Wide Area Network (WAN) for nearly two years. The data are being distributed using the Local Data Manager (LDM) software package developed by Unidata several years ago.

Although this data feed is very dependable, there are occasions when problems at a local site or the regional server contribute to missing data. Rather than accept missing data, a series of procedures was developed to allow the forecaster to proactively alleviate problems created by missing data. These procedures were developed so that the forecaster would not have to remember how to fix particular problems or possess substantial knowledge of Unix, but simply consult a flow chart and run the indicated choice from a pop-up menu by answering a few simple questions.

The series of procedures detailed below do not require the forecaster to know separate passwords or manually edit files, but simply run particular scripts to complete the desired action to fix missing data problems. In addition, these procedures also provide safeguards against local system problems that may contribute to data ingest problems.

System Problems

On occasion, local system problems may kill processes that are critical for the ingest of data (LDM) and the generation of graphics for use with Ntrans (NAG). To alleviate this, two separate safeguards have been put in place: 1) Start LDM and NAG after reboot, and 2) periodically check for LDM/NAG processes and restart if necessary.

Starting LDM and NAG after a system reboot is handled by entries in the localrc file (Attachment 1) contained in the /usr/local/bin directory, as documented by Croft (1996). The entries in the localrc file replicate the interactive process of reinitializing and starting the LDM and NAG processes. Although this procedure works very well and in most

instances, occasionally the LDM and NAG do not get restarted after a system reboot. An "INCLUDE" statement to call the localrc file must be included in the rc file (located in /usr/bin under HPUX 9.x and /usr/sbin under HPUX 10.x). To ensure that LDM and NAG are restarted, a script (Attachment 2) is run in the cron every 30 minutes (as user Gempak) to determine if these processes are running and to restart them if they are not running. This ensures that the LDM and NAG processes do not remain inactive for a prolonged period.

Determining Necessary Action

A flowchart (Fig. 1) was developed to assist the forecasters in determining the appropriate action for various situations and is accessible through the office intranet. For instance, if the data may be on the local system but Ntrans graphics are not available, the forecaster would choose the pop-up menu to "Generate Graphics." Or, if the Eta model raw data were missing for the zero through 18 hour forecasts, the forecaster would first go to the WR Data Status homepage and check to see if the data was on the WR Server. If so, then the forecaster would choose the pop-up menu to "Retrieve Data and Generate Graphics". However, if the data was not on the WR Server the forecaster could also check to see if the data was on the OSO Server via the WR Data Status Homepage.

Missing Data and Data Processing Problems

With the large volume of gridded model data obtained via the WR WAN and AFOS data via the AFOS Protocol Translator (APT), it is not surprising that some data are missed or not properly processed. However, this is not necessarily a satisfying reason for the operational forecaster. In order to ensure the forecaster has some control over the data received, several procedures have been implemented to assist the forecaster in monitoring the local data feed and to actively solve data problems on the workstations.

Monitoring Gridded Data Ingest and Processing

The forecaster must be able to quickly determine which raw gridded data has been received and resides on the local system, as well as determine which data has been processed into graphics for Ntrans. This is accomplished through the use of a script which determines which raw gridded model files reside on the local system (Barker, 1996) and also identifies the number of files waiting to be processed. This script (Attachment 3) is run from a pop-up menu and refreshes every two minutes until terminated by the user.

An example of the output of this script appears as Attachment 4. The listing provides the model, cycle and forecast times for which raw gridded files exist on the local system, which allows the forecaster to quickly determine if any files are missing from the data set. The listing also provides the model, cycle and forecast times that have been processed into graphics, which allows the

forecaster to determine the progress of the processing. The complete listing will refresh itself every two minutes, if left running.

Generating Graphics

In the event the graphics for a specified model(s) were not generated from the raw data or were generated out of order (i.e. the raw data are on the local system), a script (Attachment 5) may be run from a pop-up menu to delete any existing meta files and regenerate meta files for the specified model(s). This is accomplished through a series of questions that the user answers from which the script spawns a series of separate processes. The initial script (Attachment 5) executes a C program (Attachment 6) which runs a C-shell script (Attachment 7) as a specified user, in this case as Gempak. The advantage of this method is the username and password of user Gempak is compiled within the C program and thus does not require the user to know the password or pose a security problem. The initial script (Attachment 5) gathers input variables, such as the cycle and model(s) for which graphics should be generated. The C program (Attachment 6) simply initiates a second script with the privileges of user Gempak. This second script (Attachment 7) stops NAG, rebuilds the qf.q file with the requested model/cycle GRIB filenames, and restarts NAG which begins processing the appropriate GRIB files.

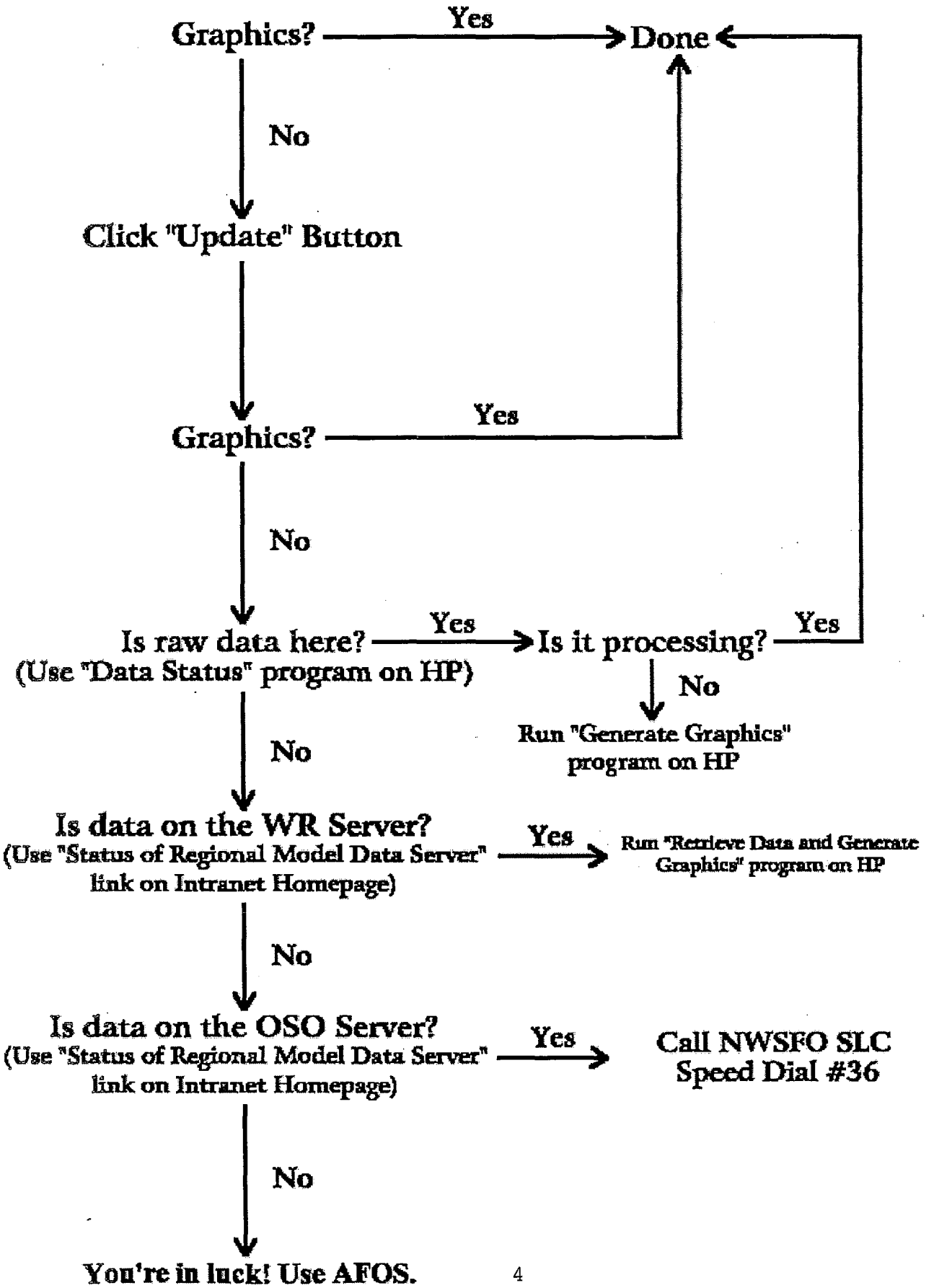
Get Raw Data and Generate Graphics

Occasionally, local problems may prevent some gridded model files from being received. Rather than live without this data, the forecaster can retrieve the raw gridded data files and generate the graphics for use with Ntrans, through the use of an interactive script (Attachment 8) that prompts the forecaster for the data of interest. Based on the input from the forecaster, this script will initiate the retrieval of the raw gridded data files from the WR Server and generate the model graphics for Ntrans. This is accomplished by a similar process as described above for Generating Graphics (Attachments 9 and 10).

Conclusion

The steps outlined above allow the forecaster to ensure viewable data (either through Ntrans or GARP) in situations when a data problem can be fixed locally. These local problems can be things like: data missed when a system was shutdown/rebooted; out of order graphics; LDM/NAG stopped for various reasons. The major advantages of this methodology is that the forecasters can fix many data problems very simply and all of these procedures are accomplished by users with a variety of computer skills (from little to many) by answering a few simple questions through scripts accessed from pop-up menus. This is all accomplished without having the forecasters log-in as the Gempak user (or local applications supervisor), which provides some security against inadvertently deleting important files.

Fig. 1


Graphics?
Yes
Done
No
Click "Update" Button
Graphics?
Yes
No
Is raw data here?
(Use "Data Status" program on HP)
Yes
Is it processing?
Yes
No
Run "Generate Graphics" program on HP
No
Is data on the WR Server?
(Use "Status of Regional Model Data Server" link on Intranet Homepage)
Yes
Run "Retrieve Data and Generate Graphics" program on HP
No
Is data on the OSO Server?
(Use "Status of Regional Model Data Server" link on Intranet Homepage)
Yes
Call NWSFO SLC Speed Dial #36
No
You're in luck! Use AFOS.

Attachment #1 -localrc

```
#!/bin/ksh
# localrc: execute local system commands
#
## Start up LDM
if [ -x /users/gempak/ldm/bin/ldmadmin ] ; then
        echo -n "rpc.ldmd "
        /bin/rm -f /tmp/ldmadmin.lockout
        /bin/su - gempak -c '/users/gempak/ldm/bin/ldmadmin delqueue
                /users/gempak/ldm/bin/ldmadmin mkqueue
                /users/gempak/ldm/bin/ldmadmin newlog
                /users/gempak/ldm/bin/ldmadmin start
                rm -f /users/gempak/ldm/data/nag.lock
                /bin/nohup /users/gempak/ldm/bin/nag start&'
fi
```

==========

Attachment #2 -ldmchk.csh

```
#!/bin/csh
#
# Set some variables for the script

cd ~gempak
set tag = `date -u "+%m/%d/%y %H:%M GMT"`
set BIN = ~/ldm/bin
set LDM = ~/ldm/data

# Check Status of the LDM

ldmping -q -i 0 >& /dev/null
if ($status == 0 ) then
  echo 'It looks like LDM is running at '$tag >> /users/gempak/logs/ldmchk.log
else
  echo 'It looks like LDM is dead at '$tag >> /users/gempak/logs/ldmchk.log
```

```
  echo '  Starting it...'>>/users/gempak/logs/ldmchk.log
  ldmadmin start >> /users/gempak/logs/ldmchk.log
endif

# Check If NAG is running

/bin/ps -ef | grep -v grep | grep ./nag_watch.pl > nagcheck.txt
if ( -z nagcheck.txt ) then
  ${BIN}/nag stop
  rm ${LDM}/nag.lock
  ${BIN}/nag start
else
  echo "nag running" > /users/gempak/logs/ldmchk.log
endif
```

==========

Attachment #3 -gridstat

```
#! /bin/csh
#
# set environmental variables
#
set e10file = (f00 f03 f06 f09 f12 f15 f18 f21 f24 f27 f30 f33)
set rucfile = (YxQAx ZxQBx YxQBx ZxQEx YxQCx)
set etafile = (YxQAx YxQBx YxQCx YxQDx YxQEx YxQFx YxQGx YxQHx YxQIx)
set rglfile = (YxQAx YxQBx YxQCx YxQDx YxQEx YxQFx YxQGx YxQHx YxQIx)
set avnfile = (YxQAx YxQBx YxQCx YxQDx YxQEx YxQFx YxQGx YxQHx YxQIx
ZxQMx YxQJx ZxQNx YxQKx)
set msofile = (f00 f03 f06 f09 f12 f15 f18 f21 f24 f27 f30 f33)
set ms2file = (f00 f03 f06 f09 f12 f15 f18 f21 f24 f27 f30 f33)
set mrffile = (YxAAx YxACx YxAEx YxAGx YxAIx YxAJx YxAKx YxALx YxAMx YxAIx
YxAQx YxASx YxAUx YxAWx YxAYx)
set ruchr = (f00 f03 f06 f09 f12)
set ngmhr = (f00 f06 f12 f18 f24 f30 f36 f42 f48)
set avnhr = (f00 f06 f12 f18 f24 f30 f36 f42 f48 f54 f60 f66 f72)
set mrfhr = (f00 f12 f24 f36 f48 f60 f72 f84 f96 f120 f144 f168 f192 f216 f240)
set yeaz = `date -u +%y`
set monz = `date -u +%m`
set dayz = `date -u +%d`
set houz = `date -u +%H`
```

```
set process = $LDMHOME/process
set msotemp = $MODEL/msotemp
set ms2temp = $MODEL/ms2temp
set e10temp = $MODEL/e10temp
#
# Check to see if server is up
#
while ('1' == '1')
echo 'Status of Gridded Data Ingest'
echo '-----------------------------------'
set val=`ping 192.133.29.170 64 1 | grep -c '1 packets received'`
if ($val == 1) then
echo 'Internet to SLC data server is UP'
else
echo 'Internet to SLC data server is DOWN'
endif
echo '-----------------------------------'
#
# set hour for RUC Model
#
if (${houz}>-1) set rucz='00'
if (${houz}>2) set rucz='03'
if (${houz}>5) set rucz='06'
if (${houz}>8) set rucz='09'
if (${houz}>11) set rucz='12'
if (${houz}>14) set rucz='15'
if (${houz}>17) set rucz='18'
if (${houz}>20) set rucz='21'
#
# Check to see if RUC is in.  If it is, an 'X' will be assigned to
# the variable $modstr; otherwise, a '.' will be assigned.
#
set moddate=`ls -1 $RAW/us008_gf086_*| tail -n1| cut -c 30-37`
set modstr=''
foreach file ($rucfile)
if (-e $RAW/us008_gf086_${moddate}_${file}) then
  set modstr=${modstr}'X'
else
  set modstr=${modstr}'.'
endif
end
#
# Set month, day, and hour
#
```

```
set mon=`echo $moddate | cut -c 3-4`
set day=`echo $moddate | cut -c 5-6`
set hou=`echo $moddate | cut -c 7-8`
echo ${mon}'/'${day}' '${hou}'Z RUC - '${modstr}
#
# Check status of ETA Model
#
set moddate=`ls -1 $RAW/us008_gf089_*| tail -n1| cut -c 30-37`
set modstr=''
foreach file ($etafile)
if (-e $RAW/us008_gf089_${moddate}_${file}) then
  set modstr=${modstr}'X'
else
  set modstr=${modstr}'.'
endif
end
set mon=`echo $moddate | cut -c 3-4`
set day=`echo $moddate | cut -c 5-6`
set hou=`echo $moddate | cut -c 7-8`
echo ${mon}'/'${day}' '${hou}'Z ETA - '${modstr}
#
# Check status of NGM Model
#
set moddate=`ls -1 $RAW/us008_gf039_*| tail -n1| cut -c 30-37`
set modstr=''
foreach file ($rglfile)
if (-e $RAW/us008_gf039_${moddate}_${file}) then
  set modstr=${modstr}'X'
else
  set modstr=${modstr}'.'
endif
end
set mon=`echo $moddate | cut -c 3-4`
set day=`echo $moddate | cut -c 5-6`
set hou=`echo $moddate | cut -c 7-8`
echo ${mon}'/'${day}' '${hou}'Z NGM - '${modstr}
#
# Check status of AVN Model
#
set moddate=`ls -1 $RAW/us008_gf077_*| tail -n1| cut -c 30-37`
set modstr=''
foreach file ($avnfile)
if (${file}== YxQAx) then
if (-e $RAW/us008_gf081_${moddate}_${file}) then
```

```
  set modstr=${modstr}'X'
else
  set modstr=${modstr}'.'
endif
else
if (-e $RAW/us008_gf077_${moddate}_${file}) then
  set modstr=${modstr}'X'
else
  set modstr=${modstr}'.'
endif
endif
end
set mon=`echo $moddate | cut -c 3-4`
set day=`echo $moddate | cut -c 5-6`
set hou=`echo $moddate | cut -c 7-8`
echo ${mon}'/'${day}' '${hou}'Z AVN - '${modstr}
#
# Check status of MRF Model
#
set moddate=`ls -1 $RAW/us008_gf078_*00_*| tail -n1| cut -c 30-37`
set modstr=''
foreach file ($mrffile)
if (-e $RAW/us008_gf078_${moddate}_${file}) then
  set modstr=${modstr}'X'
else
  set modstr=${modstr}'.'
endif
end
set mon=`echo $moddate | cut -c 3-4`
set day=`echo $moddate | cut -c 5-6`
set hou=`echo $moddate | cut -c 7-8`
echo ${mon}'/'${day}' '${hou}'Z MRF - '${modstr}
#
# Check status of 29 km Mesoeta Model (40 km grid)
#
set moddate=`ls -1 $msotemp/mso_*| tail -n1| cut -c 34-41`
set modstr=''
foreach file ($msofile)
if (-e $msotemp/mso_${moddate}_${file}.gem) then
  set modstr=${modstr}'X'
else
  set modstr=${modstr}'.'
endif
end
```

```
set mon=`echo $moddate | cut -c 3-4`
set day=`echo $moddate | cut -c 5-6`
set hou=`echo $moddate | cut -c 7-8`
echo ${mon}'/'${day}' '${hou}'Z MSO - '${modstr}
#
# Check status of 29km Mesoeta Model (20 km grid)
#
set moddate=`ls -1 $ms2temp/ms2_*| tail -n1| cut -c 34-41`
set modstr=''
foreach file ($ms2file)
if (-e $ms2temp/ms2_${moddate}_${file}.gem) then
  set modstr=${modstr}'X'
else
  set modstr=${modstr}'.'
endif
end
set mon=`echo $moddate | cut -c 3-4`
set day=`echo $moddate | cut -c 5-6`
set hou=`echo $moddate | cut -c 7-8`
echo ${mon}'/'${day}' '${hou}'Z MS2 - '${modstr}
#
# Check status of 10 km Mesoeta Model
#
set moddate=`ls -1 $e10temp/e10_*| tail -n1| cut -c 34-41`
set modstr=''
foreach file ($e10file)
if (-e $e10temp/e10_${moddate}_${file}.gem) then
  set modstr=${modstr}'X'
else
  set modstr=${modstr}'.'
endif
end
set mon=`echo $moddate | cut -c 3-4`
set day=`echo $moddate | cut -c 5-6`
set hou=`echo $moddate | cut -c 7-8`
echo ${mon}'/'${day}' '${hou}'Z E10 - '${modstr}
#
echo ' '
echo '---------------------------------'
echo '   Summary of Processed Models   '
echo '---------------------------------'
#
# See how much of RUC has been processed
#
```

```
set moddate=`ls -1 $process/ruc/RUC* | tail -n1| cut -c 35-42`
set modstr=''
foreach file ($ruchr)
if (-e $process/ruc/RUC_${moddate}_${file}) then
  set modstr=${modstr}'X'
else
  set modstr=${modstr}'.'
endif
end
set mon=`echo $moddate | cut -c 3-4`
set day=`echo $moddate | cut -c 5-6`
set hou=`echo $moddate | cut -c 7-8`
echo ${mon}'/'${day}' '${hou}'Z RUC - '${modstr}
#
# See how much of ETA has been processed
#
set moddate=`ls -1 $process/eta/ETA* | tail -n1| cut -c 35-42`
set modstr=''
foreach file ($ngmhr)
if (-e $process/eta/ETA_${moddate}_${file}) then
  set modstr=${modstr}'X'
else
  set modstr=${modstr}'.'
endif
end
set mon=`echo $moddate | cut -c 3-4`
set day=`echo $moddate | cut -c 5-6`
set hou=`echo $moddate | cut -c 7-8`
echo ${mon}'/'${day}' '${hou}'Z ETA - '${modstr}
#
# See how much of NGM has been processed
#
set moddate=`ls -1 $process/ngm/NGM* | tail -n1| cut -c 35-42`
set modstr=''
foreach file ($ngmhr)
if (-e $process/ngm/NGM_${moddate}_${file}) then
  set modstr=${modstr}'X'
else
  set modstr=${modstr}'.'
endif
end
set mon=`echo $moddate | cut -c 3-4`
set day=`echo $moddate | cut -c 5-6`
set hou=`echo $moddate | cut -c 7-8`
```

```
echo ${mon}'/'${day}' '${hou}'Z NGM - '${modstr}
#
# See how much of AVN has been processed
#
set moddate=`ls -1 $process/avn/AVN* | tail -n1| cut -c 35-42`
set modstr=''
foreach file ($avnhr)
if (-e $process/avn/AVN_${moddate}_${file}) then
  set modstr=${modstr}'X'
else
  set modstr=${modstr}'.'
endif
end
set mon=`echo $moddate | cut -c 3-4`
set day=`echo $moddate | cut -c 5-6`
set hou=`echo $moddate | cut -c 7-8`
echo ${mon}'/'${day}' '${hou}'Z AVN - '${modstr}
#
# See how much of MRF has been processed
#
set moddate=`ls -1 $process/mrf/MRF* | tail -n1| cut -c 35-42`
set modstr=''
foreach file ($mrfhr)
if (-e $process/mrf/MRF_${moddate}_${file}) then
  set modstr=${modstr}'X'
else
  set modstr=${modstr}'.'
endif
end
set mon=`echo $moddate | cut -c 3-4`
set day=`echo $moddate | cut -c 5-6`
set hou=`echo $moddate | cut -c 7-8`
echo ${mon}'/'${day}' '${hou}'Z MRF - '${modstr}
#
# See how much of 29 km Mesoeta (40 km grid) has been processed
#
set moddate=`ls -1 $process/mso/MSO* | tail -n1| cut -c 35-42`
set modstr=''
foreach file ($msofile)
if (-e $process/mso/MSO_${moddate}_${file}) then
  set modstr=${modstr}'X'
else
  set modstr=${modstr}'.'
endif
```

```
end
set mon=`echo $moddate | cut -c 3-4`
set day=`echo $moddate | cut -c 5-6`
set hou=`echo $moddate | cut -c 7-8`
echo ${mon}'/'${day}' '${hou}'Z MSO - '${modstr}
#
# See how much of 29 km Mesoeta  (20 km grid) has been processed
#
set moddate=`ls -1 $process/ms2/MS2* | tail -n1| cut -c 35-42`
set modstr=''
foreach file ($msofile)
if (-e $process/ms2/MS2_${moddate}_${file}) then
  set modstr=${modstr}'X'
else
  set modstr=${modstr}'.'
endif
end
set mon=`echo $moddate | cut -c 3-4`
set day=`echo $moddate | cut -c 5-6`
set hou=`echo $moddate | cut -c 7-8`
echo ${mon}'/'${day}' '${hou}'Z MS2 - '${modstr}
#
# See how much of 10 km Mesoeta has been processed
#
set moddate=`ls -1 $process/e10/E10* | tail -n1| cut -c 35-42`
set modstr=''
foreach file ($msofile)
if (-e $process/e10/E10_${moddate}_${file}) then
  set modstr=${modstr}'X'
else
  set modstr=${modstr}'.'
endif
end
set mon=`echo $moddate | cut -c 3-4`
set day=`echo $moddate | cut -c 5-6`
set hou=`echo $moddate | cut -c 7-8`
echo ${mon}'/'${day}' '${hou}'Z E10 - '${modstr}
#
# Check number of graphics waiting to be processed in qf.q file
# and also see if nag.lock is there, which means graphics are
# processing
#
set gphproc=`wc -l $LDMHOME/data/qf.q | cut -d' ' -f 1`
echo ' '
```

```
echo ' '
echo '----Generating GEMPAK graphics-----'
if (-e $LDMHOME/data/nag.lock) then
  echo 'Currently making graphics'
else
  echo 'Not making graphics'
endif
echo 'with '${gphproc}' times waiting.'
echo ' '
echo '---------------------------------'
echo '      Updates every 2 minutes     '
echo '---------------------------------'
echo ' '
echo ' Ctrl-C to exit'
echo ' '
#
# Wait 2 minutes before running program again.
#
sleep 120
end
```

==========

Attachment #4 -screen output from gridstat

```
Status of Gridded Data Ingest
---------------------------------
Internet to SLC data server is UP
---------------------------------
04/10 15Z RUC - XXXXX
04/10 12Z ETA - XXXXXXXXX
04/10 12Z NGM - XXXXXXXXX
04/10 12Z AVN - XXXXXXXXXXXXX
04/10 00Z MRF - XXXXXXXXXXXXXXX
04/10 15Z MSO - XXXXXXXXXXXX
04/10 15Z MS2 - XXXXXXXXXXX.
04/10 03Z E10 - XXXXXXXXXXXX
---------------------------------
   Summary of Processed Models
---------------------------------
04/10 15Z RUC - XXXXX
04/10 12Z ETA - XXXXXXXXX
04/10 12Z NGM - XXXXXXXXX
```

```
04/10 12Z AVN - XXXXXXXXXXXXX
04/10 00Z MRF - XXXXXXXXXXXXXXX
04/10 15Z MSO - XXXXXXXXXXXX
04/10 15Z MS2 - XXXXXXXXXXX.
04/10 03Z E10 - XXXXXXXXXXXX
----Generating GEMPAK graphics-----
Currently making graphics
with 0 times waiting.
---------------------------------
    Updates every 2 minutes
---------------------------------
 Ctrl-C to exit
```

==========

Attachment #5 -Generate Graphics (metaparam.csh)

```
#! /bin/csh
#
# This script will rebuild metafiles for
# N-TRANS if problems occur
#
# Check to see if user is on (put hostname of your HP here).  If not, kick
# them out of program.
#
#
set HOSTCHECK = "`hostname`"
if ( "$HOSTCHECK" != "huey") then
echo "You can only run this program from (put hostname of HP here)."
sleep 10
exit
endif
#
# Make sure program runs in correct directory
#
set HOME = $NAWIPS/exe/scripts/localapps
cd ${HOME}
#
# ask what model user wants to rebuild
#
echo -n " What model cycle do you want to rebuild (00 or 12)? "
set cycle=$<
echo $cycle > cycle
```

```
@ msocycle = $cycle + 3
if ($msocycle == 3) then
    set msocycle = '03'
endif
echo $msocycle > msocycle
#
echo " "
echo " Which of the following models do you want to rebuild?"
echo " (Answer y or n)"
echo " "
echo -n " ETA? "
set etaresponse=$<
echo $etaresponse > etares
echo ""
echo -n " AVN? "
set avnresponse=$<
echo $avnresponse > avnres
echo ""
echo -n " MESOETA? "
set msoresponse=$<
echo $msoresponse > msores
echo ""
echo -n " MRF? "
set mrfresponse=$<
echo $mrfresponse > mrfres
echo ""
echo -n " NGM? "
set ngmresponse=$<
echo $ngmresponse > ngmres
#
ldmreset
```

==========

Attachment #6 -Generate Graphics (ldmreset.c)

```
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>

char *host[] = {"" };
```

```
char *user = "user";
char *passwd = "password";
char *cmd = "rebuildmeta.csh";

void main()

      {
    int sd;
    struct servent *servent;
    FILE *fp;
    char ch;

    servent = getservbyname("exec", "tcp");
    sd = rexec(host, servent->s_port, user, passwd, cmd, 0);
     fp = fdopen(sd, "r");
     while ((ch = getc(fp)) != EOF)
      putchar(ch);
     exit(0);

      }
```

==========

Attachment #7 -Generate Graphics (rebuildmeta.csh)

```
#! /bin/csh
#
# This script will rebuild metafiles for
# N-TRANS if problems occur
#
# Set variables for year, month, and date
#
set HOME = $NAWIPS/exe/scripts/localapps
cd ${HOME}
set YEAR = `date +%y`
set MONTH = `date +%m`
set MONTH1 = `date +%b`
set DAY = `date +%e`
set CYCLE = `cat cycle`
set MSOCYCLE = `cat msocycle`
set ETARES = `cat etares`
set AVNRES = `cat avnres`
set MSORES = `cat msores`
set MRFRES = `cat mrfres`
```

```
set NGMRES = `cat ngmres`
#
# Set variables for grib files
#
set etafile = (YxQAx YxQBx YxQCx YxQDx YxQEx YxQFx YxQGx YxQHx YxQIx)
set avnfile = (YxQBx YxQCx YxQDx YxQEx YxQFx YxQGx YxQHx YxQIx ZxQMx
YxQJx ZxQNx YxQKx)
set msofile = (f00 f03 f06 f09 f12 f15 f18 f21 f24 f27 f30 f33)
set mrffile = (YxAAx YxACx YxAEx YxAGx YxAIx YxAJx YxAKx YxALx YxAMx YxAOx
YxAQx YxASx YxAUx YxAWx YxAYx)
#
# Stop nag
#
$LDMHOME/ldm-5.0.2/bin/nag stop
#
# Based on responses...add files to new qf.q file
#
if ($ETARES == "Y" || $ETARES == "y") then
        cd $NTRANS_META/eta
        rm eta*${DAY}_$CYCLE*
        cd $LDMHOME
            foreach file ($etafile)
        nag_add.pl us008_gf089_${YEAR}${MONTH}${DAY}${CYCLE}_${file}
        end
echo "
echo 'The' $CYCLE'Z' $MONTH1 $DAY 'run of the ETA is being regenerated.'
endif
#
if ($AVNRES == "Y" || $AVNRES == "y") then
        cd $NTRANS_META/avn
        rm avn*${DAY}_$CYCLE*
        cd $LDMHOME
        nag_add.pl us008_gf081_${YEAR}${MONTH}${DAY}${CYCLE}_YxQAx
        foreach file ($avnfile)
        nag_add.pl us008_gf077_${YEAR}${MONTH}${DAY}${CYCLE}_${file}
        end
echo "
echo 'The' $CYCLE'Z' $MONTH1 $DAY 'run of the AVN is being regenerated.'
endif
#
if ($MSORES == "Y" || $MSORES == "y") then
        cd $NTRANS_META/mso
        rm mso*${DAY}_$MSOCYCLE*
        cd $LDMHOME
```

```
        foreach file ($msofile)
        nag_add.pl mso_${YEAR}${MONTH}${DAY}${MSOCYCLE}_${file}.gem
        end
echo ''
echo 'The' $MSOCYCLE'Z' $MONTH1 $DAY 'run of the MESOETA is being
regenerated.'
endif
#
if ($MRFRES == "Y" || $MRFRES == "y") then
        cd $NTRANS_META/mrf
        rm mrf*${DAY}_$CYCLE*
        cd $LDMHOME
        foreach file ($mrffile)
        nag_add.pl us008_gf078_${YEAR}${MONTH}${DAY}${CYCLE}_${file}
        end
echo ''
echo 'The' $CYCLE'Z' $MONTH1 $DAY 'run of the MRF is being regenerated.'
endif
#
if ($NGMRES == "Y" || $NGMRES == "y") then
        cd $NTRANS_META/ngm
        rm ngm*${DAY}_$CYCLE*
        cd $LDMHOME
        foreach file ($etafile)
        nag_add.pl us008_gf039_${YEAR}${MONTH}${DAY}${CYCLE}_${file}
        end
echo ''
echo 'The' $CYCLE'Z' $MONTH1 $DAY 'run of the NGM is being regenerated.'
endif
#
# Restart NAG and exit from GEMPAK
#
$LDMHOME/ldm-5.0.2/bin/nag start
exit
```

==========

Attachment #8 -Retrieve Data and Generate Graphics (ftpmetaparam.csh)

```
#! /bin/csh
#
# This script will rebuild metafiles for
# N-TRANS if problems occur
```

```
#
# Check to see if user is on (put hostname of HP here).  If not, kick
# them out of program
#
set HOSTCHECK = "`hostname`"
if ( "$HOSTCHECK" != "(put hostname of HP here)") then
echo "You can only run this program from (put hostname of HP here)."
sleep 10
exit
endif
#
# Make sure program runs from correct directory.
#
set HOME = $NAWIPS/exe/scripts/localapps
cd ${HOME}
#
# ask what model user wants to rebuild
#
echo -n " What model cycle do you want to rebuild (00 or 12)? "
set cycle=$<
echo $cycle > cycle
@ msocycle = $cycle + 3
if ($msocycle == 3) then
    set msocycle = '03'
endif
echo $msocycle > msocycle
#
echo " "
echo " Which of the following models do you want to rebuild?"
echo " (Answer y or n)"
echo " "
echo -n " ETA? "
set etaresponse=$<
echo $etaresponse > etares
echo ""
echo -n " AVN? "
set avnresponse=$<
echo $avnresponse > avnres
echo ""
echo -n " MESOETA? "
set msoresponse=$<
echo $msoresponse > msores
echo ""
echo -n " MRF? "
```

```
set mrfresponse=$<
echo $mrfresponse > mrfres
echo ""
echo -n " NGM? "
set ngmresponse=$<
echo $ngmresponse > ngmres
#
ldmresetftp
```

==========

Attachment #9 -Retrieve Data and Generate Graphics (ldmresetftp.c)

```
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>

char *host[] = {"" };
char *user = "user";
char *passwd = "password";
char *cmd = "ftpbuildmeta.csh";

void main()

    {
  int sd;
  struct servent *servent;
  FILE *fp;
  char ch;

  servent = getservbyname("exec", "tcp");
  sd = rexec(host, servent->s_port, user, passwd, cmd, 0);
   fp = fdopen(sd, "r");
   while ((ch = getc(fp)) != EOF)
    putchar(ch);
   exit(0);

    }
```

==========

Attachment #10 -Retrieve Data and Generate Graphics (ftpbuildmeta.csh)

```
#! /bin/csh
#
# This script will ftp files from WR Server and rebuild
# metafiles for N-TRANS if problems occur
#
# Set variables for year, month, and date -u
#
set HOME = /users/nawips/exe/scripts/localapps
cd ${HOME}
set YEAR = `date -u +%y`
set MONTH = `date -u +%m`
set MONTH1 = `date -u +%b`
set DAY1 = `date -u +%e`
set CYCLE = `cat cycle`
set MSOCYCLE = `cat msocycle`
set ETARES = `cat etares`
set AVNRES = `cat avnres`
set MSORES = `cat msores`
set MRFRES = `cat mrfres`
set NGMRES = `cat ngmres`
#
# Set variables for grib files
#
set etafile = (YxQAx YxQBx YxQCx YxQDx YxQEx YxQFx YxQGx YxQHx YxQIx)
set avnfile = (YxQBx YxQCx YxQDx YxQEx YxQFx YxQGx YxQHx YxQIx ZxQMx
YxQJx ZxQNx YxQKx)
set msofile = (f00 f03 f06 f09 f12 f15 f18 f21 f24 f27 f30 f33)
set mrffile = (YxAAx YxACx YxAEx YxAGx YxAIx YxAJx YxAKx YxALx YxAMx YxAOx
YxAQx YxASx YxAUx YxAWx YxAYx)
#
# Stop nag
#
$LDMHOME/ldm-5.0.2/bin/nag stop
#
cd ${HOME}
#
# Check to see if $DAY1 is less than 10. If so, add a '0' before $DAY
#
if ( $DAY1 < 10 ) then
set DAY = '0'$DAY1
else
set DAY = $DAY1
```

```
endif
#
# Based on responses...get data from WR Server and
# add files to new qf.q
#
if ($ETARES == "Y" || $ETARES == "y") then
echo 'mget *089*'$DAY$CYCLE'*' > etajob
cat ftpgrib.hdr etajob ftp.ftr > getmodel.job
getmodel
        cd $NTRANS_META/eta
        rm eta*${DAY}_$CYCLE*
        cd $LDMHOME
            foreach file ($etafile)
        nag_add.pl us008_gf089_${YEAR}${MONTH}${DAY}${CYCLE}_${file}
        end
echo ''
echo 'The' $CYCLE'Z' $MONTH1 $DAY 'run of the ETA is being regenerated.'
endif
#
cd ${HOME}
#
if ($AVNRES == "Y" || $AVNRES == "y") then
echo 'mget *081*'${DAY}${CYCLE}'*' > avnjob
echo 'mget *077*'${DAY}${CYCLE}'*' >> avnjob
cat ftpgrib.hdr avnjob ftp.ftr > getmodel.job
getmodel
        cd $NTRANS_META/avn
        rm avn*${DAY}_$CYCLE*
        cd $LDMHOME
        nag_add.pl us008_gf081_${YEAR}${MONTH}${DAY}${CYCLE}_YxQAx
            foreach file ($avnfile)
        nag_add.pl us008_gf077_${YEAR}${MONTH}${DAY}${CYCLE}_${file}
        end
echo ''
echo 'The' $CYCLE'Z' $MONTH1 $DAY 'run of the AVN is being regenerated.'
endif
#
cd ${HOME}
#
if ($MSORES == "Y" || $MSORES == "y") then
echo 'mget mso*'${DAY}${MSOCYCLE}'*' > msojob
cat ftpgem.hdr msojob ftp.ftr > getmodel.job
getmodel
        cd $NTRANS_META/mso
        rm mso*${DAY}_$MSOCYCLE*
```

```
        cd $LDMHOME
        foreach file ($msofile)
        nag_add.pl mso_${YEAR}${MONTH}${DAY}${MSOCYCLE}_${file}.gem
        end
echo ''
echo 'The' $MSOCYCLE'Z' $MONTH1 $DAY 'run of the MESOETA is being
regenerated.'
endif
#
cd ${HOME}
#
if ($MRFRES == "Y" || $MRFRES == "y") then
echo 'mget *078*'${DAY}${CYCLE}'*' > mrfjob
cat ftpgrib.hdr mrfjob ftp.ftr > getmodel.job
getmodel
        cd $NTRANS_META/mrf
        rm mrf*${DAY}_$CYCLE*
        cd $LDMHOME
        foreach file ($mrffile)
        nag_add.pl us008_gf078_${YEAR}${MONTH}${DAY}${CYCLE}_${file}
        end
echo ''
echo 'The' $CYCLE'Z' $MONTH1 $DAY 'run of the MRF is being regenerated.'
endif
#
cd ${HOME}
#
if ($NGMRES == "Y" || $NGMRES == "y") then
echo 'mget *039*'$DAY$CYCLE'*' > ngmjob
cat ftpgrib.hdr ngmjob ftp.ftr > getmodel.job
getmodel
        cd $NTRANS_META/ngm
        rm ngm*${DAY}_$CYCLE*
        cd $LDMHOME
        foreach file ($etafile)
        nag_add.pl us008_gf039_${YEAR}${MONTH}${DAY}${CYCLE}_${file}
        end
echo ''
echo 'The' $CYCLE'Z' $MONTH1 $DAY 'run of the NGM is being regenerated.'
endif
#
# Restart NAG and exit from GEMPAK
#
$LDMHOME/ldm-5.0.2/bin/nag start
exit
```