Wide-field Infrared Survey Explorer (WISE)

Data Processing Operations Procedures

Version [2.0]

21-January-2010

Prepared by: Ron Beck

Infrared Processing and Analysis Center
California Institute of Technology

WSDC D-C005

Approved By:

Ned Wright, WISE Principal Investigator

Donald Royer, WISE Mission Operations Center Manager

Roc Cutri, WISE Science Data Center Manager

[Other Appropriate Names], WISE Science Data Center [Title]

[Other Appropriate Names], WISE Science Data Center [Title]

Revision History

Date	Version	Author	Description
10/02/09	0.1	Ron Beck	Initial Draft
10/28/09	1.0	Ron Beck	
01/21/10	2.0	Ron Beck	

Table of Contents

1 INTRODUCTION

1.1 Document Scope

This document will explain the procedures for Data Processing Operations. Once the raw data has been ingested, we can run the scan pipeline processing on it. We expect to receive about 32 scans per day containing around 250 frames worth of raw images in four deliveries. The scan pipeline processing runs the frame pipeline on all frames and then will run a number of modules on the entire scan. Once all the scans for a region of the sky have been processed, we can run the one degree square coadds that cover that region including the source lists. When the coadds and source lists are created, we will have scheduled deliveries of the scans, coadds and lists to IRSA so the data can be made public. We are responsible for the raw data archive which means multiple backups and sending copies offsite. We need to manage resources - disk space, cpus, condor (the pipeline distribution system) and troubleshoot network issues.

1.2 Applicable Documents

http://wisewiki.ipac.caltech.edu/index.php/Nodes_command
http://wisewiki.ipac.caltech.edu/index.php/Node_clean_command
http://wisewiki.ipac.caltech.edu/index.php/Scan_summary_command
http://wisewiki.ipac.caltech.edu/index.php/Compile_mframe_logs_command
http://wisewiki.ipac.caltech.edu/index.php/Qlook_summary_command
http://wisewiki.ipac.caltech.edu/index.php/Broken_links_command
http://wisewiki.ipac.caltech.edu/index.php/Add2scans_command
http://wisewiki.ipac.caltech.edu/index.php/Run_scans_command
http://wisewiki.ipac.caltech.edu/index.php/Showme_running_command
http://www.cs.wisc.edu/condor/manual/v6.2/8_Command_Reference.html#command-reference

1.3 Acronyms

IPAC - Infrared Processing and Analysis Center, California Institute of Technolo
gy
MOS - Mission Operations System
WSDC - WISE Science Data Center (IPAC)
WSDS - WISE Science Data System

2 Scan Pipeline Procedure

2.1 run_scans Command

Command /home/beck/bin/run_scans is the scan distribution script to be run on wcnode01. This script will run continuously in operations. In dir /wise/fops/operations, there is a file "scan". run_scans will continuously (every 10 seconds) check this file for scans to run. The scan file can take two different forms. There can be entries with just the scan identifier or a scan identifier with the wsspipe command to be run.

Here is an example of both flavors ...

```
beck@caustic;rhe4():operations[0]% cat scan
01248a
01248a pass "wsspipe -ind . -cluster -preclean all -steps @all,+dynacal"
beck@caustic;rhe4():operations[0]%
```

Any entries with just the scan identifier will be run with the following exec.

```
wsspipe -v -cluster -preclean all -mkdir -ind . -frnums -1
```

When the run_scans script sees the pass parameter, it knows to run the attached wsspipe command verbatim instead of the vanilla exec.

The run_scans script will monitor the condor job queue using the condor_status command. When the number of jobs returned from this command dips below 400, the script will look to start another scan from the scan file. This is an attempt to keep all the node slots busy continuously.

run_scans has an optional parameter which is the max number of scans to run at once and defaults to 8.

2.2 run_scans Miscellaneous files

The run_scans script creates or updates numerous files. These files are located in the /wise/fops/operations dir. File exec.status keeps track of the scan start and stop times. Here is a sample.

```
beck@caustic;rhe4():operations[0]% tail exec.status
Jul 13 16:37:39 01248a start
Jul 13 16:45:37 01248a complete
Jul 16 11:35:28 01248a start
Jul 16 11:59:30 01248a complete
Jul 16 13:11:51 01248a start
Jul 16 13:42:17 01248a complete
Jul 16 15:19:24 01250a start
Jul 16 15:28:54 01250a complete
beck@caustic;rhe4():operations[0]%
```

This comes in handy if you want to know when a scan was run or how many times.

There is a gostop file where you can control whether more scans are started from the scan file. 0 means don't start any more, 1 means let em rip. This is queried every 10 seconds by run_scans.

2.3 Scan Summary Script

When a scan completes, the run_scans script fires off another script that goes through all the log files for the scan including all the frame pipe logs and outputs them to files in the /wise/fops/operations/problems dir. the files take the form SCAN.MM-D-HH:MM:SS.PROBS where PROBS is either ok or errors.

Below is a directory listing with scan summary files.

```
beck@caustic;rhe4():problems[0]% pwd
/wise/fops/operations/problems
beck@caustic;rhe4():problems[0]% \ls -1 | head
00433a.06-30-14:55:14.errors
00434a.06-30-15:48:44.errors
00434a.07-1-08:12:05.ok
00435a.06-30-15:49:13.errors
00435a.07-1-08:13:16.ok
00436a.06-30-15:55:21.errors
00436a.07-1-08:18:39.ok
00437a.06-30-19:46:30.ok
00438a.06-30-19:40:56.ok
00439a.06-30-19:51:41.ok
beck@caustic;rhe4():problems[0]%
```

The *.ok scans return '0' code for all pipeline modules whereas the *.errors files had some non-zero codes returned.

Below is a sample ok listing.

```
beck@caustic;rhe4():problems[1]% m 12345a.06-13-18:37:18.ok
FRAME      START          ELAP STAT  SIG CODE HOST  PROGRAM
     09/06/14_01:24:44  12:13    0    0    0   caustic WSSPipe
 001 09/06/14_01:24:57  05:18    0    0    0   10 WSFPipe
 001 09/06/14_01:35:53  00:04    0    0    0   20 WSFPipePost
 002 09/06/14_01:24:57  05:23    0    0    0   10 WSFPipe
 002 09/06/14_01:35:53  00:10    0    0    0   04 WSFPipePost
 003 09/06/14_01:24:57  10:20    0    0    0   02 WSFPipe
 003 09/06/14_01:35:54  00:09    0    0    0   11 WSFPipePost
 004 09/06/14_01:24:57  05:21    0    0    0   11 WSFPipe
 004 09/06/14_01:35:55  00:08    0    0    0   12 WSFPipePost
 005 09/06/14_01:24:57  04:15    0    0    0   20 WSFPipe
 005 09/06/14_01:35:55  00:11    0    0    0   08 WSFPipePost
 007 09/06/14_01:24:57  05:25    0    0    0   10 WSFPipe
 007 09/06/14_01:35:55  00:10    0    0    0   08 WSFPipePost
 008 09/06/14_01:24:57  10:31    0    0    0   02 WSFPipe
 008 09/06/14_01:35:54  00:10    0    0    0   04 WSFPipePost
 009 09/06/14_01:24:57  05:19    0    0    0   11 WSFPipe
 009 09/06/14_01:35:55  00:10    0    0    0   08 WSFPipePost
 010 09/06/14_01:24:57  04:14    0    0    0   20 WSFPipe
 010 09/06/14_01:35:55  00:10    0    0    0   08 WSFPipePost
 011 09/06/14_01:24:57  05:28    0    0    0   10 WSFPipe
 011 09/06/14_01:35:54  00:07    0    0    0   30 WSFPipePost
 012 09/06/14_01:24:57  10:34    0    0    0   02 WSFPipe
 012 09/06/14_01:35:54  00:09    0    0    0   20 WSFPipePost
     09/06/14_01:35:33  00:00    0    0    0   caustic Spawn_rotmeta
     09/06/14_01:35:33  00:13    0    0    0   caustic Spawn_wssflag
     09/06/14_01:35:33  00:13    0    0    0   caustic WSSFlag
     09/06/14_01:35:34  00:00    0    0    0   caustic Spawn_dumptbl
     09/06/14_01:35:48  00:02    0    0    0   30 WSSPCal
     09/06/14_01:35:49  00:00    0    0    0   30 Spawn_spcal
     09/06/14_01:35:51  00:17    0    0    0   caustic Spawn_wsspipepost
     09/06/14_01:35:51  00:17    0    0    0   caustic WSSPipePost
     09/06/14_01:36:19  00:27    0    0    0   10 Spawn_scansync
     09/06/14_01:36:19  00:37    0    0    0   10 ScanQA
     09/06/14_01:36:46  00:01    0    0    0   10 Spawn_rotmeta
     09/06/14_01:36:47  00:03    0    0    0   10 Spawn_rotmeta
     09/06/14_01:36:51  00:01    0    0    0   10 Spawn_rotmeta
     09/06/14_01:36:52  00:01    0    0    0   10 Spawn_rotmeta
     09/06/14_01:36:53  00:01    0    0    0   10 Spawn_rotmeta
     09/06/14_01:36:54  00:01    0    0    0   10 Spawn_rotmeta
     09/06/14_01:36:55  00:00    0    0    0   10 Spawn_rotmeta
     09/06/14_01:36:55  00:01    0    0    0   10 Spawn_dumptbl
```

The summary is sorted by the module start time and lists the return codes and elapsed wall clock time by module. An error file will have non-zero return code to be followed by the offending error messages.

2.4 run_scans Summary Output

The window where run_scans is running updates every 10 seconds and monitors the running scans by sifting through the WSSPipe.log file. The display shows the running scans, the currently running module and the scan elapsed time. If the scan is in the WSFPIPE step, the run_scans script breaks that down by total frames, running, complete and failed frames. Below is a sample of the screen.

```
                              scan    elapsed   step
 of    251 frames total       01248a  00:21:10  SFPIPE          Jul 17 14:40:38
       217 frames running
        33 frames finished ok
>>>      1 frames failed

sleeping 10 ...
```

2.5 showme_running Script

Additionally, run_scans outputs a exec.update file that looks similar to the above screen sample every 10 seconds. This works in conjunction with the showme_running script that anyone can run and see the scans progress just like the ops people.

Here is a sample output from showme_running ...

```
                              scan    elapsed   step
 of    251 frames total       01248a  00:22:14  SFPIPE          Jul 17 14:41:42
         7 frames running
       243 frames finished ok
>>>      1 frames failed
```

2.6 wmspipe file

Once a scan completes, an entry of the scan id is appended to file /wise/fops/operations/wmspipe. This will be used later on when running the wmspipe command.

3 Multi Scan Pipeline Procedure

Once the scan frame pipelines complete, we can run the multi scan pipeline. This pipeline will run on an entire scan and applies photometric calibration and artifact identification to the lists and images.

3.1 Multi Scan Pipeline

Below is a sample command line for running the multi scan pipeline on scan 01707a.

```
 wmspipe -v -cluster -dataroot /wise/tops -scans 01707a \
-out_base /wise/tops/scans/7a/01707a
```

The wmspipe command submits a number of processes to the nodes that run pretty quickly on a dry cluster 2 - 3 minutes per scan.

3.2 run_wmspipe command

The file /wise/fops/operations/wmspipe contains scan ids that have been run through the scan frame pipelines. These scans are now available to be run through the wmspipe command. The scan id is loaded into the wmspipe file regardless of return code from the scan frame pipeline so a little discretion is needed here when using this command. We want to make sure the scans are ready for wmspipe. Additionally, we need to run the scans in acsending scan order. The run_wmspipe command will run the scans in the /wise/fops/operations/wmspipe file in ascending order, just make sure the scan frame pipelines don't get run out of order or wait before running wmspipe. run_wmspipe will grab a scan id from the wmspipe file and run it while removing it from the wmspipe file.

3.3 run_wmspipe command usage

```
beck@wcnode35;rhe4(int.v3.2):~[0]% /home/beck/bin/run_wmspipe
command run_wmspipe: template
               where template is template filename
```

Here is an example of the template file.

```
beck@wcnode35;rhe4(int.v3.2):~[0]% cat /wise/fops/operations/wmspipe.template
wmspipe -v -cluster -dataroot /wise/tops -scans SCAN -out_base /wise/tops/scans/PRE/SCAN
beck@wcnode35;rhe4(int.v3.2):~[0]%
```

The run_wmspipe command once started will run continuously until the /wise/fops/operations/wmspipe file is exhausted. it will take each scan id entry from the /wise/fops/operations/wmspipe file in ascending scan order. duplicate scan ids are eliminated. The script basically, substitutes for the SCAN and PRE in the template file where PRE is the last 2 characters of the SCAN id, runs it and displays the return code.

3.4 run_wmspipe log file

The same output displayed to the screen for this command also is captured in log file /wise/fops/operations/logs/wmspipe.log. Below is an example.

```
Nov 13 22:28:46 wmspipe -v -cluster -dataroot /wise/tops -scans 01164a -out_base
/wise/tops/scans/4a/01164a
  Nov 13 22:32:32 successful wmspipe for scan 01164a ...
Nov 13 22:32:32 wmspipe -v -cluster -dataroot /wise/tops -scans 01165a -out_base
/wise/tops/scans/5a/01165a
  Nov 13 22:33:59 successful wmspipe for scan 01165a ...
  found no scans to wmspipe ...
```

Notice from the log file where the command line is the same as the template file just substituting for SCAN and PRE.

4 Coadd Pipeline Procedure

Once the scans and the multiscan pipeline have been run and QA have blessed the scans, we can now create the 1.5 degree coadds. The procedure consists of the wmcpipe command which given scan and position criteria will create and execute the wmfpipe or the multi frame pipeline.

4.1 wmcpipe Command

Below is the wmcpipe command used to create and run the coadds for the 30 orbit simulation scans 01707a-01765a.

```
wmcpipe -v -tail tops_300off -outb tops_300off -workloc local \
-data_root /wise/tops -preclean all -copts cmdfile=1 \
-elonspec date=20100113T031534,20100115T014203,elonoff=-90 -getfix \
-:getfix '-f scan=01707a-01765a,pstat=0'
```

In this example, the criteria given is the scan range which is all the scans for the 30 orbit sim, the date that the deliveries cover and the "elonoff=-90" means only do one side of the sky. This command generated 122 wmfpipe commands submitted to the cluster. This is however only one side of the sky. This same command was run again for the other side of the sky using the "elonoff=90" parm.

The wmcpipe command tracks the progress of the wmfpipes, logging the return codes in the *-coadds.tbl in the working directory that the command is run. File /wise/fops/coadds/coadds.tbl contains all coadds that have been run.

4.2 run_coadds Command

The run_coadds command uses a template file for creating the wmcpipe commands and executing it. It takes as input the starting and ending scan ids and a switch for just displaying the created command, starting one of the sides or both. The created command is executed in background and the WMCPipe*log files need to be monitored for completion. The working directory is created in /wise/fops/coadds/dMMM where MMM is WISE mission day number. Below is an example template file.

```
 beck@caustic;rhe4():templates[0]% cat coadd_1
 wmcpipe -v -tail DAY_OF_MISSION_1 -preclean all -elonspec
date=STARTUTC,ENDUTC,elonoff=90 -getfix -:getfix '-f scan=STARTSCAN-ENDSCAN,pstat=0'
 beck@caustic;rhe4():templates[0]%
```

The capitalized words are substituted for by the script. STARTSCAN and ENDSCAN are command line parms and the rest are calculated by run_coadds.

4.3 run_coadds command usage

```
beck@caustic;rhe4(ops):~/bin[0]% /home/beck/bin/run_coadds
 command run_coadds: STARTSCAN STOPSCAN RUNIT
             where   RUNIT is n/y/1/2
                             n - just show command
                             y - run it
                             1 - run side 1
                             2 - run side 2
beck@caustic;rhe4(ops):~/bin[0]%
```

The example below just displays the created command using "n" for the RUNIT parm.

```
wiseops@wcnode01;rhe4(ops):~[0]% /home/beck/bin/run_coadds 00934b 00964a n
 command: wmcpipe -v -tail d031_1 -preclean all -elonspec
date=20100114T104551,20100115T103719,elonoff=90 -getfix -:getfix '-f scan=00934b-
00964a,pstat=0'
 command: wmcpipe -v -tail d031_2 -preclean all -elonspec
date=20100114T104551,20100115T103719,elonoff=-90 -getfix -:getfix '-f scan=00934b-
00964a,pstat=0'
wiseops@wcnode01;rhe4(ops):~[0]%
```

This example is the same as above only using a "1" for RUNIT which starts the side 1 run in background.

```
wiseops@wcnode01;rhe4(ops):~[0]% /home/beck/bin/run_coadds 00934b 00964a 1
 starting command: wmcpipe -v -tail d031_1 -preclean all -elonspec
date=20100114T104551,20100115T103719,elonoff=90 -getfix -:getfix '-f scan=00934b-
00964a,pstat=0'
 coadds started - check logs in /wise/fops/coadds/d031 for progress ...
wiseops@wcnode01;rhe4(ops):~[0]%
```

The scan range is all the scans run for the previous night except for the last scan which we assume is a partial scan which will be completed with the next delivery. Note that the script tells you where the output logs are going. There will be 122 coadds started per side. The WMCPipe*.log files need to be queried to determine the coadds progress.

5 IRSA Delivery Procedure

Twice weekly we will get a request for IRSA database deliveries. This entails the source lists generated by the frame pipelines and their images. We will also deliver the coadded images and their source lists. This is accomplished with the warchpipe script.

5.1 WARCHPIPE Script

The warchpipe output data currently goes to /wise/data/irsa_in directory. There will be a loadX subdir created where the intermediate files are made. The final products will be created in the irsa_in dir. Below is a sample command.

```
warchpipe -out_dir /wise/data/irsa_in -load 5 -steps @std -log
/wise/data/irsa_in/warchpipe.5
```

Below is a sample of the resulting created files for a scan.

```
beck@caustic;rhe4():load5[0]% \ls -al !$
\ls -al *02333b*
-rwxrwxr-x  2 evans wise       1707 Jan  6 15:10 chksum.1b.out.02333b
-rwxrwxr-x  2 evans wise       1707 Jan  6 15:10
chksum.1b.out_100106_230102_smNj.save_02333b
-rwxrwxr-x  2 evans wise     117983 Jan  6 14:53 dbprep.1b.out.02333b
-rwxrwxr-x  2 evans wise     117983 Jan  6 14:53
dbprep.1b.out_100106_222606_uEHN.save_02333b
-rwxrwxr-x  2 evans wise       9985 Jan  6 14:59 imgprep.1b.out.02333b
-rwxrwxr-x  2 evans wise       9985 Jan  6 14:59
imgprep.1b.out_100106_225629_lHim.save_02333b
-rw-rw-r--  1 evans wise      16967 Jan  6 14:53 wise_i1ba_frm_4_5.bartbl.02333b
-rw-rw-r--  1 evans wise     486682 Jan  6 14:53 wise_i1ba_mch_4_5.bartbl.02333b
-rw-rw-r--  1 evans wise      13156 Jan  6 14:53 wise_i1bc_frm_4_5.bartbl.02333b
-rw-rw-r--  1 evans wise     123311 Jan  6 14:53 wise_i1bc_mch_4_5.bartbl.02333b
-rw-rw-r--  1 evans wise        279 Jan  6 14:53 wise_i1bc_scn_4_5.bartbl.02333b
-rw-rw-r--  1 evans wise         61 Jan  6 14:56 wise_i1bl_lod_4_5.bartbl.02333b
-rw-rw-r--  1 evans wise     423016 Jan  6 15:10 wise_i1bl_xfr_4_5.txt.02333b
-rw-rw-r--  1 evans wise     933746 Jan  6 14:59 wise_i1bm_frm_4_5.bartbl.02333b
-rw-rw-r--  1 evans wise     550039 Jan  6 14:53 wise_i1bs_frm_4_5.bartbl.02333b
-rw-rw-r--  1 evans wise 1420531384 Jan  6 14:53 wise_i1bs_psd_4_5.bartbl.02333b
-rw-rw-r--  1 evans wise   10561920 Jan  6 14:53 wise_i1bs_psd_4_5.sptind.txt.02333b
beck@caustic;rhe4():load5[0]% pwd
/wise/data/irsa_in/load5
beck@caustic;rhe4():load5[0]%
```

The *.bartbl* files are database load ready pipe-delimited files. There are several output files from the different modules that operate on the scan. The *xfr*txt* file contains information for copying the 1b images and checksumming. The *psd*sptind* file is a spacial index file needed by IRSA.

Here is a sample of the coadd created files.

```
beck@caustic;rhe4():load5[0]% ls -al *1825m399_d002_2*
-rwxrwxr-x  2 evans wise     1767 Jan  6 15:02 chksum.3o.out.1825m399_d002_2
-rwxrwxr-x  2 evans wise     1767 Jan  6 15:02
chksum.3o.out_100106_230103_OBpk.save_1825m399_d002_2
-rwxrwxr-x  2 evans wise     4916 Jan  6 14:28 dbprep.3o.out.1825m399_d002_2
-rwxrwxr-x  2 evans wise     4916 Jan  6 14:28
dbprep.3o.out_100106_222531_Sy89.save_1825m399_d002_2
-rwxrwxr-x  2 evans wise     4383 Jan  6 14:57 imgprep.3o.out.1825m399_d002_2
```

```
-rwxrwxr-x  2 evans wise     4383 Jan  6 14:57
imgprep.3o.out_100106_225630_vuIS.save_1825m399_d002_2
-rw-rw-r--  1 evans wise       70 Jan  6 14:56 wise_i3ol_lod_4_5.bartbl.1825m399_d002_2
-rw-rw-r--  1 evans wise     1728 Jan  6 15:02 wise_i3ol_xfr_4_5.txt.1825m399_d002_2
-rw-rw-r--  1 evans wise     2062 Jan  6 14:57 wise_i3om_cdd_4_5.bartbl.1825m399_d002_2
-rw-rw-r--  1 evans wise     1968 Jan  6 14:27 wise_i3os_cdd_4_5.bartbl.1825m399_d002_2
-rw-rw-r--  1 evans wise 40425793 Jan  6 14:27 wise_i3os_psd_4_5.bartbl.1825m399_d002_2
-rw-rw-r--  1 evans wise   324630 Jan  6 14:27
wise_i3os_psd_4_5.sptind.txt.1825m399_d002_2
beck@caustic;rhe4():load5[0]% pwd
/wise/data/irsa_in/load5
beck@caustic;rhe4():load5[0]%
```

They are similar in nature to the scans files only not as many.

The primary product from warchpipe is the load manifest file. This file contains all the concatonated scan and coadd files for the entire delivery. Below is an example for load5.

```
beck@caustic;rhe4():irsa_in[0]%  cat manifest.5.10010622
/wise/data/irsa_in/wise_i1ba_frm_4_5.bartbl   b791e1d3f5d47b397fbbf35d114dc7fd
/wise/data/irsa_in/wise_i1ba_mch_4_5.bartbl   5eba535fd9577b113801bcb5178eaa77
/wise/data/irsa_in/wise_i1bc_frm_4_5.bartbl   d6bbf7f9ca44e40a7a4e43cbe70adc0d
/wise/data/irsa_in/wise_i1bc_mch_4_5.bartbl   f5aa67556574e8d31f23e4fad341c93f
/wise/data/irsa_in/wise_i1bc_scn_4_5.bartbl   652fde9f0aa03f94547732e8736fa948
/wise/data/irsa_in/wise_i1bl_lod_4_5.bartbl   a8e9018ee94e51ef3942d0e7f28af678
/wise/data/irsa_in/wise_i1bm_frm_4_5.bartbl   a9df7d91c3ff49027b8fefd4ef4f971e
/wise/data/irsa_in/wise_i1bs_frm_4_5.bartbl   713f91fee5632f4da0b53e8a870f2d74
/wise/data/irsa_in/wise_i1bs_psd_4_5.bartbl   3ffb63f0b8c9c290e65688ccf1039be0
/wise/data/irsa_in/wise_i1bl_xfr_4_5.txt   c19d39ad37dfd687773339647c874680
/wise/data/irsa_in/wise_i1bs_psd_4_5.sptind.txt   d8735c653967e88b61a1ed88a08cab0a
/wise/data/irsa_in/wise_i3ol_lod_4_5.bartbl   64dcfd98b1a6b293c1a071af9ad63521
/wise/data/irsa_in/wise_i3om_cdd_4_5.bartbl   baf35242f618e87f887b36c09966153e
/wise/data/irsa_in/wise_i3os_cdd_4_5.bartbl   73e750b3bc248cda3515939febdb4d43
/wise/data/irsa_in/wise_i3os_psd_4_5.bartbl   25fd74f7ee57a1fa98f760bc9072f645
/wise/data/irsa_in/wise_i3ol_xfr_4_5.txt   b4d3ae3a920b90d9bf9123c59e19a8c9
/wise/data/irsa_in/wise_i3os_psd_4_5.sptind.txt   ad6a24d1d8a93528ae9f1c25dc3f2033
beck@caustic;rhe4():irsa_in[0]% pwd
/wise/data/irsa_in
beck@caustic;rhe4():irsa_in[0]%
```

This is a list of all the files IRSA should pull over and their checksums. Once complete this file is emailed to the irsa-wise-notify@lists.ipac.caltech.edu to alert them that a load is ready for them to pickup.

Problems need to be addressed with Tracey Evans.

6 WISE Raw Data Backup Procedure

We are the sole repository for the WISE raw data until we ship off a copy to NSSDC. We need to backup to tape the raw MOS and HRP data we receive daily.

6.1 raw_backup Script

Log on to machine nyx using the wisesw account. Execute command /home/wisesw/bin/raw_backup.

```
 > ./raw_backup
 command usage: raw_backup YYDOY DRIVE TAPE

                where YYDOY is the 2 digit year and julian doy dir to backup
                      DRIVE is 0 or 1 for the dat drives on nyx to use
                      TAPE  is the outer tape lable for the directory
```

The raw_backup script will backup the raw data located in /wise/fops/ingest/delivs/YYDOY. All subdirectories in this dir will be backed up one per tape file. The DRIVE parm expects either 0 or 1 for the tape drives located on the nyx machine. This machine is used because the ingest/delivs dir is local to nyx. The directory containing the tape names, files and contents is in /home/wisesw/raw_data_tape_log. There is an entry for each file on all tapes and the columns are tape name, file number, dir backed up, size, tape drive name and write date.

Once we get a successful backup of the YYDOY dir, we need to make 2 copies of the tapes, one for a local copy, and one to be sent offsite to long term storage. The latter to be arranged by the IPAC library people. This can be accomplished using a dd command from the nyx drive 0 to 1.

7 Pipeline Machines Maintenance

7.1 nodes Script

The nodes command is handy when you want to run the same command on all node machines. The IFILE parm is optional and would contain a list of machines to run on. If no IFILE parm is supplied, file /wise/fops/operations/nodes will be used. You have to create a public key prior to running this command unless you want to be prompted for a password for each machine. Instructions below.

```
beck@caustic;rhe4(ops):~[0]% nodes

 command usage: nodes COMMANDS BACKGROUND IFILE

         where  COMMANDS is list of commands separated by ';' to
                run on all pipeline nodes
                BACKGROUND is y or n to put the command in background
                IFILE is file containing list of machines to run
                commands on (optional and defaults to wcnode01-32)
```

Here is an example.

```
beck@caustic;rhe4(ops):~[1]% nodes "uptime" n
 ssh wcnode01 uptime ...
 09:13:58 up 20 days, 18:03,  1 user,  load average: 0.28, 0.21, 0.12
 ssh wcnode02 uptime ...
 09:13:58 up 20 days, 17:57,  1 user,  load average: 0.00, 0.01, 0.00
 ssh wcnode03 uptime ...
 09:13:58 up 20 days, 17:57,  0 users,  load average: 0.03, 0.02, 0.00
 ssh wcnode04 uptime ...
 09:13:59 up 20 days, 17:58,  0 users,  load average: 0.00, 0.02, 0.00
 ssh wcnode05 uptime ...
 09:13:59 up 20 days, 17:58,  0 users,  load average: 0.00, 0.00, 0.00
 ssh wcnode06 uptime ...
 09:14:00 up 20 days, 17:58,  0 users,  load average: 0.00, 0.00, 0.00
 ssh wcnode07 uptime ...
 09:14:01 up 20 days, 17:58,  0 users,  load average: 0.01, 0.01, 0.00
 ssh wcnode08 uptime ...
 09:14:01 up 20 days, 17:58,  0 users,  load average: 0.01, 0.01, 0.00
```

```
 ssh wcnode09 uptime ...
 09:14:02 up 20 days, 17:57,  0 users,  load average: 0.00, 0.00, 0.00
 ssh wcnode10 uptime ...
 09:14:03 up 20 days, 17:58,  1 user,  load average: 0.72, 0.88, 0.81
beck@caustic;rhe4(ops):~[0]%
```

7.2 Creating a Public Key

Instructions for creating a public key.

```
sirius:.ssh tungn$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/tungn/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/tungn/.ssh/id_rsa.
Your public key has been saved in /Users/tungn/.ssh/id_rsa.pub.
```

copy the id_rsa.pub to the remote machine $HOME/.ssh/authorized_keys

In this case, the remote machine would be caustic.

7.3 Node Machines Disk Usage

The scan pipelines create work directories on the node machines in the /local/wise directories fops, rtb and tops. These work directories contain intermediate pipeline processed files which would be useful for debug purposes and are linked to the /wise/[fops|rtb|tops]/scans dirs. There are two scripts to help us maintain enough space in these directories to continuously keep scan pipelines running.

7.3.1 broken_links Script

Command broken_links is to be run on the wcnode machines and will check the dirs on the wcnode machine versus the work link name in /wise/[fops|rtb|tops/scans/*/*/fr/*. Output displays discrepancies with the option of deleting the broken link dirs thereby saving the node /local space.

```
beck@caustic;rhe4(ops):~[0]% /home/beck/bin/broken_links
 command usage: broken_links DELETE
         where  DELETE is y or n to delete or just display
beck@caustic;rhe4(ops):~[0]%
```

Below is an example run on wcnode10.

```
/home/beck/bin/broken_links n
```

Creates following output.

```
/compute/wcnode13/wise/fops/scans/5a/00435a/fr/051/work ...
/compute/wcnode21/wise/fops/scans/5a/00435a/fr/100/work ...
/compute/wcnode02/wise/fops/scans/5a/00435a/fr/002/work ...
/compute/wcnode11/wise/fops/scans/5a/00435a/fr/020/work ...
/compute/wcnode26/wise/fops/scans/5a/00445a/fr/144/work ...
/compute/wcnode05/wise/fops/scans/5a/00445a/fr/117/work ...
/compute/wcnode16/wise/fops/scans/5a/00445a/fr/059/work ...
/compute/wcnode14/wise/fops/scans/5a/00445a/fr/237/work ...
/local/wise/fops/scans/5a/00445a/fr/090/work link broken - dir does not exist ...
/compute/wcnode11/wise/fops/scans/2a/00442a/fr/159/work ...
/local/wise/fops/scans/2a/00442a/fr/147/work link broken - dir does not exist ...
```

All the lines that start with /compute are the link names that are in the /wise/[fops|rtb|tops]/scans/*/*/fr/* dirs or where the frame work dir actually is for that frame. The "link broken" lines mean that there is a dir on wcnode10 with no corresponding link in /wise/[fops|rtb|tops]/scans at all and therefore orphaned and a prime candidate for deletion.

7.3.2 node_clean Script

Command node_clean runs continuously on the wcnode machines 2 - 32 making sure we do not run out of pipeline work space on the nodes. Log files are located in /wise/fops/operations/node_clean dir.

```
command usage: node_clean DISKPER CLEANPER SLEEP

        where  DISKPER is minimum percent used on /local to start deletes
               CLEANPER is percentage to clean down to
               SLEEP is minutes before checking
```

notes: this command will awaken every SLEEP minutes and check the /local disk used percentage versus DISKPER parm. if percent used is greater than DISKPER, all /local/wise/[fops|rtb|tops]/scans work dir names are gathered by modified date. the oldest work dirs are deleted until /local used percentage is below CLEANPER and then back to sleep. the pointer to the /local node dir in /wise/fops/scans is also deleted first checking that it indeed is pointing to the /local work directory.

node_clean also cleans up all condor log files older than 3 days that are located in the /local directory.

7.4 Node Machines Messages Files

The wcnode machines messages files are located in /var/log. The messages files are rotated weekly with the messages file being current and messages.1, messages.2 and so on being older versions. The messages file can be read with the "dmesg" command as in the example below.

```
beck@wcnode10;rhe4():log[0]% dmesg messages | tail -15
microcode: No new microdata for cpu 6
microcode: No new microdata for cpu 7
IA-32 Microcode Update Driver v1.14 unregistered
Linux Kernel Card Services
  options:  [pci] [cardbus] [pm]
ip_tables: (C) 2000-2002 Netfilter core team
ip_tables: (C) 2000-2002 Netfilter core team
MSI INIT SUCCESS
bnx2: eth0: using MSI
bnx2: eth0 NIC Copper Link is Up, 1000 Mbps full duplex
ip_tables: (C) 2000-2002 Netfilter core team
i2c /dev entries driver
Installing knfsd (copyright (C) 1996 okir@monad.swb.de).
flatcal[28746]: segfault at 0000000000771958 rip 000000000042f200 rsp 0000007fbfff9468
error 4
Losing some ticks... checking if CPU frequency changed.
beck@wcnode10;rhe4():log[0]%
```

This can clue us in should the node machine start having problems such as a disk going bad, disk out of space, memory problems etc.

8 Condor commands

Condor is the software package that we use for distributing background jobs to the node machines. The condor master runs on wcnode01 while the machines wcnode02 - 32 do the actual processing as jobs are farmed out to them by the master. The condor master and clients start up automatically when rebooted and otherwise run continuously.

8.1 condor_status Command

The condor_status command displays all the nodes status and a summary of claimed and unclaimed nodes. See below.

```
beck@caustic;rhe4(ops):~[0]% condor_status

Name               OpSys      Arch   State     Activity LoadAv Mem   ActvtyTime

slot1@wcnode02.ipa LINUX      X86_64 Unclaimed Idle     0.000  4020  0+08:57:43
slot2@wcnode02.ipa LINUX      X86_64 Unclaimed Idle     0.000  4020  0+06:38:56
slot3@wcnode02.ipa LINUX      X86_64 Claimed   Busy     0.000  4020  0+10:46:07
slot4@wcnode02.ipa LINUX      X86_64 Unclaimed Idle     0.000  4020  0+09:50:28
slot5@wcnode02.ipa LINUX      X86_64 Claimed   Busy     0.000  4020  0+10:46:09
slot6@wcnode02.ipa LINUX      X86_64 Unclaimed Idle     0.080  4020  0+01:14:24
slot7@wcnode02.ipa LINUX      X86_64 Unclaimed Idle     0.000  4020  0+14:59:32
slot8@wcnode02.ipa LINUX      X86_64 Unclaimed Idle     0.000  4020  0+13:18:06
slot1@wcnode03.ipa LINUX      X86_64 Unclaimed Idle     0.000  4020  0+10:37:16
slot2@wcnode03.ipa LINUX      X86_64 Claimed   Busy     0.000  4020  0+10:46:56
slot3@wcnode03.ipa LINUX      X86_64 Claimed   Busy     0.000  4020  0+10:46:57
slot4@wcnode03.ipa LINUX      X86_64 Unclaimed Idle     0.000  4020  0+14:12:37
slot5@wcnode03.ipa LINUX      X86_64 Unclaimed Idle     0.000  4020  0+15:03:53
slot6@wcnode03.ipa LINUX      X86_64 Unclaimed Idle     0.000  4020  0+15:02:43
slot7@wcnode03.ipa LINUX      X86_64 Unclaimed Idle     0.000  4020  0+15:04:30
slot8@wcnode03.ipa LINUX      X86_64 Unclaimed Idle     0.040  4020  0+00:55:03
slot1@wcnode04.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+07:31:21
slot2@wcnode04.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:05:22
slot3@wcnode04.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:05:58
slot4@wcnode04.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:06:30
slot5@wcnode04.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:17:26
slot6@wcnode04.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:14:32
slot7@wcnode04.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:38:54
slot8@wcnode04.ipa LINUX      X86_64 Unclaimed Idle     0.030  1501  0+00:55:03
slot1@wcnode05.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:05:45
slot2@wcnode05.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:05:51
slot3@wcnode05.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:14:19
slot4@wcnode05.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:13:17
slot5@wcnode05.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:39:04
slot6@wcnode05.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+17:09:42
slot7@wcnode05.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+17:08:08
slot8@wcnode05.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+00:55:04
slot1@wcnode06.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:16:16
slot2@wcnode06.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:17:47
slot3@wcnode06.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:36:55
slot4@wcnode06.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:46:15
slot5@wcnode06.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+17:13:15
slot6@wcnode06.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+17:08:05
slot7@wcnode06.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+00:55:10
slot8@wcnode06.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+19:05:23
slot1@wcnode07.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:38:24
slot2@wcnode07.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:45:59
slot3@wcnode07.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+17:09:57
```

```
 slot4@wcnode07.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+19:04:57
 slot5@wcnode07.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+19:05:04
 slot6@wcnode07.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+19:11:14
 slot7@wcnode07.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+20:29:32
 slot8@wcnode07.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+00:55:03
 slot1@wcnode08.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+05:16:48
 slot2@wcnode08.ipa LINUX      X86_64 Claimed   Busy     0.000  1501  0+10:45:41
 slot3@wcnode08.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+11:44:04
 slot4@wcnode08.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:04:39
 slot5@wcnode08.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:01:14
 slot6@wcnode08.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:04:03
 slot7@wcnode08.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:17:21
 slot8@wcnode08.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+00:55:03
 slot1@wcnode09.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:03:48
 slot2@wcnode09.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+08:21:49
 slot3@wcnode09.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:06:47
 slot4@wcnode09.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:04:23
 slot5@wcnode09.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:19:25
 slot6@wcnode09.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:15:07
 slot7@wcnode09.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:39:58
 slot8@wcnode09.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+00:55:03
 slot1@wcnode10.ipa LINUX      X86_64 Claimed   Busy     0.060  1501  0+10:45:09
 slot2@wcnode10.ipa LINUX      X86_64 Claimed   Busy     0.000  1501  0+10:45:10
 slot3@wcnode10.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+00:32:01
 slot4@wcnode10.ipa LINUX      X86_64 Claimed   Busy     0.000  1501  0+10:45:16
 slot5@wcnode10.ipa LINUX      X86_64 Claimed   Busy     0.000  1501  0+10:45:17
 slot6@wcnode10.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+02:35:22
 slot7@wcnode10.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+05:42:25
 slot8@wcnode10.ipa LINUX      X86_64 Unclaimed Idle     0.700  1501  0+00:55:03

                     Total Owner Claimed Unclaimed Matched Preempting Backfill

        X86_64/LINUX   248     0      23       225       0          0        0

               Total   248     0      23       225       0          0        0
 beck@caustic;rhe4(ops):~[0]%
```

Another variation of the command is adding the -submitters parm. This will display the jobs submitted by user and their status.

```
 beck@wcnode01;rhe4(dev):09264[0]% condor_status -submitters

 Name                  Machine      Running IdleJobs HeldJobs

 tedlungu@ipac.caltec caustic.ip       184        0        0
 beck@ipac.caltech.ed wcnode01.i        96     1020        0

                              RunningJobs           IdleJobs           HeldJobs

 beck@ipac.caltech.ed                 96               1020                  0
 tedlungu@ipac.caltec                184                  0                  0

                Total                280               1020                  0
```

Adding the -total parm will display just the totals by user.

```
 beck@wcnode01;rhe4(dev):09264[0]% condor_status -submitters -total

                            RunningJobs           IdleJobs           HeldJobs

 beck@ipac.caltech.ed                96               1020                  0
 tedlungu@ipac.caltec               184                  0                  0

                Total               280               1020                  0
```

8.2 condor_q Command

The condor_q command displays the actual jobs status and owners.

```
beck@caustic;rhe4(ops):~[0]% condor_q -g


-- Schedd: wcnode11.ipac.caltech.edu : <134.4.142.213:32774>
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
  54.0   bauer           8/4  23:17   0+10:52:04 R  0   1220.7 procGlob
  54.1   bauer           8/4  23:17   0+10:52:04 R  0   732.4 procGlob
  54.4   bauer           8/4  23:17   0+10:52:07 R  0   1220.7 procGlob
  54.6   bauer           8/4  23:17   0+10:52:07 R  0   1220.7 procGlob
  54.9   bauer           8/4  23:17   0+10:52:07 R  0   1953.1 procGlob
  54.10  bauer           8/4  23:17   0+10:52:07 R  0   976.6 procGlob
  54.13  bauer           8/4  23:17   0+10:52:07 R  0   1464.8 procGlob
  54.14  bauer           8/4  23:17   0+10:52:07 R  0   976.6 procGlob
  54.15  bauer           8/4  23:17   0+10:52:07 R  0   2685.5 procGlob
  54.17  bauer           8/4  23:17   0+10:52:07 R  0   317.4 procGlob
  54.20  bauer           8/4  23:17   0+10:52:07 R  0   244.1 procGlob
  54.21  bauer           8/4  23:17   0+10:52:07 R  0   1464.8 procGlob
  54.23  bauer           8/4  23:17   0+10:52:07 R  0   1464.8 procGlob
  54.24  bauer           8/4  23:17   0+10:52:07 R  0   1464.8 procGlob
  54.25  bauer           8/4  23:17   0+10:52:07 R  0   732.4 procGlob
  54.26  bauer           8/4  23:17   0+10:52:07 R  0   1220.7 procGlob
  54.27  bauer           8/4  23:17   0+10:52:07 R  0   7324.2 procGlob
  54.29  bauer           8/4  23:17   0+10:52:07 R  0   1709.0 procGlob
  54.30  bauer           8/4  23:17   0+10:52:07 R  0   1464.8 procGlob
  54.31  bauer           8/4  23:17   0+10:52:07 R  0   244.1 procGlob
  54.32  bauer           8/4  23:17   0+10:52:07 R  0   1709.0 procGlob
  54.33  bauer           8/4  23:17   0+10:52:07 R  0   1709.0 procGlob
22 jobs; 0 idle, 22 running, 0 held

-- Schedd: caustic.ipac.caltech.edu : <134.4.141.175:32785>
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
84506.0   bauer           8/4  17:40   0+16:28:14 R  0   122.1 hashGlobs -i 30orb

1 jobs; 0 idle, 1 running, 0 held
beck@caustic;rhe4(ops):~[0]%
```

8.3 condor_on Command

From the condor master machine wcnode01, you can run command condor_on NODE where NODE is the wcnode machine name you want to start running condor on. a condor_status command will confirm that the node slots for the machine have been started.

8.4 condor_off Command

From the condor master machine wcnode01, you can run command condor_off NODE where NODE is the wcnode machine name you want to stop running condor on. a

condor_status command will confirm that the node slots for the machine have been stopped. Should there be processes running on the node slots when the command is issued, they will be allowed to finish before shutting down.

8.5 condor_rm Command

From the condor master machine wcnode01, you can run command condor_rm USER where USER is the user name associated with the processes that you want to kill. This will kill all running processes instantly and remove any jobs that have yet to start. Jobs can also be deleted by cluster number and cluster.proc. These are numbers assigned by condor when you submit your job. A condor_q or condor_status -submitters will confirm that the jobs are gone.

Condor is a very sophisticated software package with lots of bells and whistles that we will not need. We will be using it to basically control running of the batch pipelines. The following link explains all of the available commands.

http://www.cs.wisc.edu/condor/manual/v6.2/8_Command_Reference.html#command-reference

9 Survey Progress Reports

We need to create table files containing position information on scans that have been run. Ned Wright at UCLA uses these tables to plan the survey strategy. These table files are copied to phoebe:/Volumes/gnd01/down/Tasks/Survey daily once the scans have been completed and Ned emailed.

9.1 run_ned_report command

The run_ned_report command runs on a user supplied list of scans. Typically, these scans will be the scans completed from the overnight deliveries. The table files are currently created in /wise/data/beck before being transferred over to phoebe. The last table created needs to be renamed to table "a" because the next set of tables created will start with the last table file. We don't want to append to existing files but rather create a new "b" file. below is a listing of the current table files in phoebe:/Volumes/gnd01/down/Tasks/Survey.

```
[phoebe:down/Tasks/Survey] beck% ls -al
total 12816
drwxrwxr-x + 13 tim      wise      442 Jan 18 09:59 .
drwxr-xr-x + 12 wachter  wise      408 Jan 14 10:36 ..
-rw-r--r-- +  1 beck     wise    36212 Jan 18 09:59 survey_progress.20100110.tbl
-rw-r--r-- +  1 beck     wise    35628 Jan 18 09:59 survey_progress.20100111.tbl
-rw-r--r-- +  1 beck     wise   817750 Jan 18 09:59 survey_progress.20100112.tbl
-rw-r--r-- +  1 beck     wise  1030326 Jan 18 09:59 survey_progress.20100113.tbl
-rw-r--r-- +  1 beck     wise   430412 Jan 18 09:59 survey_progress.20100114a.tbl
-rw-r--r-- +  1 beck     wise   603276 Jan 18 09:59 survey_progress.20100114b.tbl
-rw-r--r-- +  1 beck     wise   414790 Jan 18 09:59 survey_progress.20100115a.tbl
-rw-r--r-- +  1 beck     wise   603130 Jan 18 09:59 survey_progress.20100115b.tbl
-rw-r--r-- +  1 beck     wise  1058650 Jan 18 09:59 survey_progress.20100116.tbl
-rw-r--r-- +  1 beck     wise  1024736 Jan 18 09:59 survey_progress.20100117.tbl
-rw-r--r-- +  1 beck     wise   482562 Jan 18 09:59 survey_progress.20100118a.tbl
[phoebe:down/Tasks/Survey] beck% pwd
/Volumes/gnd01/down/Tasks/Survey
[phoebe:down/Tasks/Survey] beck%
```

9.2 run_ned_report Command Usage

```
beck@caustic;rhe4(ops):~/bin[0]% /home/beck/bin/run_ned_report
 command run_ned_report: SCANS
beck@caustic;rhe4(ops):~/bin[0]%
```

The run_ned_report script uses a template file that looks like this.

```
beck@caustic;rhe4(ops):templates[0]% cat ned_report
 getfix -cols @std,-glon,-glat,-elon,-elat -frtime STARTUTC,ENDUTC | egrep -v '(^\\|^\|)'
>> /wise/data/beck/survey_progress.RUNDATE.tbl
beck@caustic;rhe4(ops):templates[0]%
```

The script calculates the capitalized words from the input scans. Below is an example run.

```
beck@caustic;rhe4(ops):beck[0]% /home/beck/bin/run_ned_report $x
 starting command:  getfix -cols @std,-glon,-glat,-elon,-elat -frtime
20100118T114210,20100118T121716 | egrep -v '(^\\|^\|)' >>
/wise/data/beck/survey_progress.20100118.tbl
 starting command:  getfix -cols @std,-glon,-glat,-elon,-elat -frtime
20100118T123939,20100118T131140 | egrep -v '(^\\|^\|)' >>
/wise/data/beck/survey_progress.20100118.tbl
 .
 .
 .
 starting command:  getfix -cols @std,-glon,-glat,-elon,-elat -frtime
20100119T095608,20100119T102910 | egrep -v '(^\\|^\|)' >>
/wise/data/beck/survey_progress.20100119.tbl
 starting command:  getfix -cols @std,-glon,-glat,-elon,-elat -frtime
20100119T105207,20100119T112538 | egrep -v '(^\\|^\|)' >>
/wise/data/beck/survey_progress.20100119.tbl
beck@caustic;rhe4(ops):beck[0]%
```

The resulting files in /wise/data/beck need to be renamed.

```
beck@caustic;rhe4(ops):beck[141]% ls -lt | head -3
total 5128638
-rw-rw-r--   1 beck    wise        480549 Jan 19 11:25 survey_progress.20100119.tbl
-rw-rw-r--   1 beck    wise        594534 Jan 19 11:24 survey_progress.20100118.tbl
beck@caustic;rhe4(ops):beck[141]%
```

Move survey_progress.20100118.tbl to survey_progress.20100118b.tbl because there is an "a" file out there and rename survey_progress.20100119.tbl to survey_progress.20100119a.tbl because tomorrow's delivery will create another survey_progress.20100119.tbl file. The files are then copied to phoebe and an email sent to Ned Wright.

10 Manual Backups

We need to run manual backups from the 4 servers nyx, themis, myson and erebus for the l0, scans, and ingest/delivs dirs located under /wise/fops. This would be a hardship on the ISG automated backups because it would take too long. This procedure consists of three steps. Creating the backup directory list per machine, copying and running that list on the different servers and checking the output and updating the history files.

10.1 setup_nsr command

The setup_nsr command will query files in /wise/fops/operations/hist/fops and look for scans, l0 and ql dirs that have not been backed up. here is an example of one of the hist files.

```
beck@caustic;rhe4():fops[0]% cat 00982a
10016T044513 ingestpipe ops.v3.2.2  0 Jan 16 00:15:53 Jan 16 01:20:05
             scanframe ops.v3.2.2  3 Jan 16 03:15:19 Jan 16 03:39:23
               wmspipe ops.v3.2.2  0 Jan 16 12:36:57 Jan 16 12:37:55
          l0    nsrsave         na  0 Jan 16 16:19:45 Jan 17 01:27:46
   scanframe    nsrsave         na  0 Jan 16 16:20:21 Jan 16 23:47:07
beck@caustic;rhe4():fops[0]% pwd
/wise/fops/operations/hist/fops
beck@caustic;rhe4():fops[0]%
```

Each of the processes that we run against a scan will be logged in this type of file. In this example you can see that this scan - 00982a also the filename was ingested as part of delivery 10016T044513. This scan was run through scanframe and wmspipe pipelines. The scan and l0 data have also been backed up with the manual backup. There are start and stop dates along with the return codes so we can track the history of a scan. There are also files created for MOS and TLM deliveries. See below.

```
beck@caustic;rhe4():fops[0]% cat 10011M031014
10011M031014 ingestpipe ops.v3.2.2  0 Jan 10 21:03:06 Jan 10 21:04:05
  ingestpipe    nsrsave         na  0 Jan 14 15:38:13 Jan 14 19:48:59
beck@caustic;rhe4():fops[0]% pwd
/wise/fops/operations/hist/fops
beck@caustic;rhe4():fops[0]%
```

The setup_nsr command checks these files to see what needs to be backed up and creates a file with the directories that need to nsrsaved.

10.2 setup_nsr Command Usage

```
beck@caustic;rhe4():fops[0]% /home/beck/bin/setup_nsr
 command setup_nsr: OPS
             where  OPS is tops or fops
beck@caustic;rhe4():fops[0]%
```

Currently this command is only setup for backing up fops data so the OPS parm will be fops. Below is an example.

```
beck@caustic;rhe4():fops[0]% /home/beck/bin/setup_nsr fops
 created /tmp/nyx ...
 created /tmp/myson ...
 created /tmp/themis ...
 created /tmp/erebus ...
beck@caustic;rhe4():fops[0]%
```

Displaying the /tmp/nyx reveals the directories on server nyx that need backing up.

```
beck@caustic;rhe4():fops[0]% cat /tmp/nyx
/export/ops-11/wise/fops/ingest/delivs/10018/10018M015428
/export/ops-11/wise/fops/ingest/delivs/10018/10018M035738
/export/ops-11/wise/fops/ingest/delivs/10018/10018M052645
/export/ops-11/wise/fops/ingest/delivs/10018/10018M114000
/export/ops-11/wise/fops/ingest/delivs/10018/10018M125308
/export/ops-11/wise/fops/ingest/delivs/10018/10018M181521
/export/ops-11/wise/fops/ingest/delivs/10018/10018T024629
/export/ops-11/wise/fops/ingest/delivs/10018/10018T042531
/export/ops-11/wise/fops/ingest/delivs/10018/10018T104342
/export/ops-11/wise/fops/ingest/delivs/10018/10018T122127
/export/ops-11/wise/fops/ingest/delivs/10019/10019M041351
```

```
/export/ops-11/wise/fops/ingest/delivs/10019/10019M044401
/export/ops-11/wise/fops/ingest/delivs/10019/10019M112913
/export/ops-11/wise/fops/ingest/delivs/10019/10019M124021
/export/ops-11/wise/fops/ingest/delivs/10019/10019M181536
/export/ops-11/wise/fops/ingest/delivs/10019/10019T023446
/export/ops-11/wise/fops/ingest/delivs/10019/10019T041251
/export/ops-11/wise/fops/ingest/delivs/10019/10019T103253
/export/ops-11/wise/fops/ingest/delivs/10019/10019T120908
/export/ops-11/wise/fops/l0/0a/01060a
/export/ops-11/wise/fops/l0/0a/01070a
/export/ops-11/wise/fops/l0/0a/01080a
/export/ops-11/wise/fops/l0/6b/01086b
/export/ops-11/wise/fops/l0/8b/01058b
/export/ops-11/wise/fops/l0/8b/01078b
/export/ops-11/wise/fops/scans/1a/01061a
/export/ops-11/wise/fops/scans/1a/01081a
/export/ops-11/wise/fops/scans/3b/01073b
/export/ops-11/wise/fops/scans/5a/01065a
/export/ops-11/wise/fops/scans/5a/01085a
/export/ops-11/wise/fops/scans/7a/01057a
/export/ops-11/wise/fops/scans/7a/01077a
/export/ops-11/wise/fops/scans/9b/01069b
beck@caustic;rhe4():fops[0]%
```

Note that any scans, MOS or TLM deliveries that have a current nsrsave entry in their file will not be selected for backup. I say current because should a scan be rerun after an nsrsave has been run on it, it would again be eligable for backup. Same goes for l0 data should a delivery be reingested. The run_scans, run_wmspipe and run_ingest scripts all create entries in the hist files. Once the setup_nsr files are created, they are copied to their respective machines home directory which is /export/home/wisesw. Once there the run_backups script running on the four servers takes over.

10.3 run_backups Command

The run_backups command runs continuously on each of the four server machines nyx, themis, myson and erebus. This script looks in the /export/home/wisesw directory for it's hostname named file. Once finding this file will begin an instance of the nsrsave command backing up the directories contained in the file.

10.4 run_backups Command Usage

```
> /export/home/wisesw/bin/run_backups
 command usage: run_backups SLEEP

         where  SLEEP is number of seconds to sleep
```

This command should always be running in the background currently using a SLEEP parm of 300 seconds or 5 minutes. It will check every 5 minutes for it's hostname named file of directories to backup. When this file is found, it is renamed appending the current date and the nsrsave command is started. Output from the nsrsave will go to file HOSTNAME.MMDDYY.out. See below example on server nyx.

```
> ls nyx.010910*
nyx.010910      nyx.010910.out
> pwd
/export/home/wisesw
>
```

The run_backups script found the nyx file copied over from the setup_nsr script, renamed it to nyx.010910, started the nsrsave command with output going to the nyx.010910.out file. Once the nsrsave command completes, it will append the nyx.010910 with start and stop timestamps and a nsrsave summary line as below.

```
> tail nyx.010910
/export/ops-11/wise/fops/scans/7a/00737a
/export/ops-11/wise/fops/scans/7a/00757a
/export/ops-11/wise/fops/scans/7a/00777a
/export/ops-11/wise/fops/scans/9b/00689b
/export/ops-11/wise/fops/scans/9b/00709b
/export/ops-11/wise/fops/scans/9b/00749b
/export/ops-11/wise/fops/scans/9b/00769b
#Jan 9 09:02:31
#save: /export/ops-11/wise/fops/  187 GB 03:28:05 831682 files
#Jan 9 12:30:40
>
```

10.5 run_backups Log File

The run_backups script creates a log file in /export/home/wisesw/nsr.log. Below is an example of the log.

```
tail -20 nsr.log
Jan 11 21:28:59: found backup to do ...
 /usr/bin/nsr/save -i -s kelley -y 01/11/2016 -b "WISE stage" -I
/export/home/wisesw/nyx.011110 1> /export/home/wisesw/nyx.011110.out 2>&1

save: /export/ops-11/wise/fops/  120 GB 02:12:39 373565 files
Jan 11 23:41:43: backup complete ...
Jan 14 15:38:13: found backup to do ...
 /usr/bin/nsr/save -i -s kelley -y 01/14/2016 -b "WISE stage" -I
/export/home/wisesw/nyx.011410 1> /export/home/wisesw/nyx.011410.out 2>&1

save: /export/ops-11/wise/fops/  248 GB 04:10:44 734043 files
Jan 14 19:48:59: backup complete ...
Jan 16 16:20:08: found backup to do ...
 /usr/bin/nsr/save -i -s kelley -y 01/16/2016 -b "WISE stage" -I
/export/home/wisesw/nyx.011610 1> /export/home/wisesw/nyx.011610.out 2>&1

save: /export/ops-11/wise/fops/  251 GB 07:50:06 733230 files
Jan 17 00:10:16: backup complete ...
Jan 18 16:31:08: found backup to do ...
 /usr/bin/nsr/save -i -s kelley -y 01/18/2016 -b "WISE stage" -I
/export/home/wisesw/nyx.011810 1> /export/home/wisesw/nyx.011810.out 2>&1

save: /export/ops-11/wise/fops/  255 GB 03:45:59 751062 files
Jan 18 20:17:08: backup complete ...
>
```

When the nsrsave command completes, the HOSTNAME.MMDDYY files are then copied back to /wise/fops/operations/nsrsave for input to script update_nsr.

10.6 update_nsr Script

When the nsrsaves complete and the HOSTNAME.MMDDYY files are copied back to /wise/fops/operations/nsrsave, the update_nsr script can be run. The update_nsr script reads in these files and updates the scans, MOS and TLM files located in /wise/fops/operations/hist/fops with the nsrsave scans, l0 and ingest/delivs backup times.

10.7 update_nsr Command Usage

```
beck@caustic;rhe4(ops):~/bin[0]% /home/beck/bin/update_nsr
 command setup_nsr: BACKUPFILE
              where  BACKUPFILE is the nsr backup file
beck@caustic;rhe4(ops):~/bin[0]%
```

This command will read the BACKUPFILE copied over from the server, find the start and stop times at the end of the file and create an entry in each appropriate scan, MOS and TLM file in /wise/fops/operations/hist/fops.