3. IRIS Diagnostic Utilities

IRIS supplies a number of commands that can help with system management and troubleshooting.

3.1 sigmet_env Command

This program tests many things which could cause problems with an IRIS installation. If you suspect problems, please run this (logged in as the normal IRIS user), and check the results. The following are checked:

- Checks that all the IRIS operators and observers are in the /etc/users file.
- Checks that all IRIS environment variables which point to directories are defined and the directory exists, and the directory can be read and written as required.
- Checks for some obvious bad file names in the saved menu directory.
- Checks that IRIS executable files which require the "set–UID–on–execute" bit set are setup with the correct UID.
- Checks the RDA (RVP8/RCP8) environment as well..

Here is an example, with a bad file name of ".TSC".

```
$ sigmet_env
Checking IRIS_OPERATORS list...
Checking IRIS_OBSERVERS list...
Checking installation directories...
Checking configuration directories...
Checking data directories...
Checking file names in IRIS_MENU...
Bad menu filename: '/usr/sigmet/config/menu/.TSC'
Checking root file ownerships...

==========================================
  Errors Detected -- Please Check Printout
==========================================
```

3.2 ps_iris Command

The **ps_iris** command lists all the currently active IRIS, antenna, and utility processes, including information about their owner UID, PID, time start time, and total CPU time.

Use this command to determine what IRIS processes are running on the system and when they were started. You can use the PID as an argument to the **kill** command if you need to stop a process.

```
$ ps_iris
IRIS Activity on 'humid.sigmet.com' at: Wed Dec  4 16:38:50 EST 1996
Detached Processes:
        UID   PID  PPID  C    STIME TTY      TIME COMMAND
   operator 26705     1  0 15:34:56 ttyp6    0:00 ingfio IRIS_INGFIO
   operator 26709     1  0 15:34:57 ttyp6    0:00 server IRIS_SERVER
   operator 26713     1  0 15:34:59 ttyp6    0:00 output IRIS_OUTPUT13
   operator 26718 26706  0 15:36:27 ttyp6    0:00 network IRIS_NETWORK
   operator 26714 26711  0 15:34:59 ttyp6    0:00 window IRIS_WINDOW1
   operator 26710     1  0 15:34:58 ttyp6    0:00 watchdog IRIS_WATCHDOG
   operator 26708     1  0 15:34:57 ttyp6    0:00 reingest IRIS_REINGEST
   operator 26706     1  0 15:34:56 ttyp6    0:00 network IRIS_NETWORK
   operator 26712     1  0 15:34:59 ttyp6    0:00 output IRIS_OUTPUT02
   operator 26711     1  0 15:34:58 ttyp6    0:00 output IRIS_OUTPUT01
   operator 26704     1  0 15:34:55 ttyp6    0:00 ingest IRIS_INGEST

Antenna Processes:
        UID   PID  PPID  C    STIME TTY      TIME COMMAND
   operator 26422     1  0 14:35:11 ttyp6    0:00 ant_xmt IRIS_ANT_XMT
   operator 26418     1  0 14:35:10 ttyp6    0:00 ant_rcv IRIS_ANT_RCV

Stand-alone Utilities:
        UID   PID  PPID  C    STIME TTY      TIME COMMAND
       alan 26717 28786 15 15:35:50 ttyqb    0:02 iris
       alan 26894 26717 15 15:41:14 ttyqb    0:02 iris_clnt_recv 7 1097655
```

3.3 restart_iris Command

The **restart_iris** command runs through all the IRIS host processes and checks to see if they are still running. If any have stopped (usually due to a fatal error), it restarts them. Because of the automatic restarting feature, this utility is useful mainly for restarting processes after manually killing them for debugging purposes.

It takes the following command line options

–colors	Reload color setup information
–restart	Restart all IRIS processes (default, if no other args)
–resurrect	Only restart from fatal internal errors
–rescan	Rescan file system for new PRODUCT files
–silent	Do not print informational messages

Note that the IRIS automatic restarting feature runs "restart_iris –resurrect –silent". Use of the –rescan option is for diagnostic purposes only. Once you have a working configuration, please run **qiris** and **siris** to make sure everything is OK.

```
$ restart_iris
Restarting all IRIS processes...
    IRIS_INGEST OK.
    IRIS_INGFIO OK.
   IRIS_RTD_XMT OK.
   IRIS_NETWORK 0K.
   IRIS_NORDRAD OK.
   IRIS_PRODUCT OK.
  IRIS_REINGEST OK.
    IRIS_SERVER OK.
  IRIS_WATCHDOG OK.
  IRIS_OUTPUT01 Restarted.
  IRIS_OUTPUT02 OK.
  IRIS_OUTPUT03 OK.
  IRIS_OUTPUT04 OK.
  IRIS_ARCHIVE1 OK.
  IRIS_ARCHIVE2 OK.
  IRIS_OUTPUT07 OK.
  IRIS_OUTPUT08 OK.
```

3.4 show_iris Command

The **show_iris** command gives information about the IRIS process — when it was started, the present state of semaphores and event flags, plus the current inventory of in-use products.

The **show_iris**command also provides some usefule command line options for in–use bits. Type **show_iris –help** to learn more.

```
$ show_iris
IRIS Activity on 'hot' at: 09:52:52 17 SEP 1999

IRIS V7.11 was started at 16:19:38 16 SEP 1999 by 'joe'.
  Manual startup from TTY:'/dev/tty' ; Restarts:1

Features License: 00004001–000101–WAHRMA–01–Y9ANHF
Products License: 000007FF–000101–WAHRMA–03–WFW4KR

Present states of Semaphores...
    Process Control: FREE  (ID: 5833)         Process Modes: FREE  (ID: 5826)
      Task Schedule: FREE  (ID: 5834)      Product Schedule: FREE  (ID: 5831)
   Ingest Directory: FREE  (ID: 5831)     Product Directory: FREE  (ID: 5849)
       Device Table: FREE  (ID: 5852)     Mode Switch Table: FREE  (ID: 5834)
  Archive Directory: FREE  (ID: 5834)             Error Log: FREE  (ID: 5834)

Present states of Event Flags...
          RTDISP: CLEAR                INGEST: CLEAR
          INGFIO: CLEAR       INGFIO Mapping: SET
  INGFIO Waiting: SET                WATCHDOG: CLEAR
         PRODUCT: CLEAR               REINGEST: CLEAR
         NETWORK: CLEAR                NORDRAD: CLEAR
   Global Mapped: SET

Event Flags SET for Output Processes:  7 8

Event Flags SET for Network Child Processes:  1 2 3 4 5 6 7 8 9 10 11 12 13 14
15 16

Checking INGEST inventory for in–use files:
  Total files checked: 92, total in use: 0.

Checking PRODUCT inventory for in–use files:
  Total files checked: 260, total in use: 0.

            ======= Product Inventory Contents =======
      Prod Type      Count  Size(Mb)   Kept Count  Kept Size
   ---------------   -----  --------   ----------  ----------
              PPI       0      0.00          0        0.00
              RHI       0      0.00          0        0.00
            CAPPI     159     92.06         21       60.92
    Cross Section       0      0.00          0        0.00
        Echo Tops       0      0.00          0        0.00
         Tracking       0      0.00          0        0.00
  Hourly Rainfall       0      0.00          0        0.00
 N Hours Rainfall       0      0.00          0        0.00
  Vol. Vel. Proc.       0      0.00          0        0.00
 Vert.Int. Liquid       0      0.00          0        0.00
```

Wind Shear	0	0.00	0	0.00
Warning	1	0.01	1	0.01
Real Time PPI	0	0.00	0	0.00
Real Time RHI	0	0.00	0	0.00
Raw Data	73	52.35	73	52.35
Max with panels	0	0.00	0	0.00
User Map	0	0.00	0	0.00
User Section	0	0.00	0	0.00
User Other	0	0.00	0	0.00
Status	25	0.06	0	0.00
Shear Line	0	0.00	0	0.00
Horizontal Wind	0	0.00	0	0.00
Beam Pattern	0	0.00	0	0.00
Text	0	0.00	0	0.00
Forecast	0	0.00	0	0.00
Multi–Doppler	2	15.36	2	15.36
Image	0	0.00	0	0.00
Composite	0	0.00	0	0.00
LLWAS	0	0.00	0	0.00
	260	159.84 Mb	97	128.65 Mb

3.5 structmap Command

The **structmap** command displays the format of IRIS structures. This is useful for programmers writing applications that access IRIS data. You must install IRIS with the **–headers** option to make **structmap** available on your system.

structmap can take several options, producing different results. To display the list of options, enter the command without options or parameters:

```
$ structmap

Command Line Options:
   <struct name> : Display internal contents of IRIS structure(s)
  –include <dir> : Override default 'include' directory name
         –nopack : Force no packing of structure elements
           –scan : Produce list of all defined structures
      –scanlocal : Like 'scan', but do local directory only
        –noflags : Suppress error flags in output
      –recursive : Descend into substructures
           –data : Show numeric data read from std.input
    –dimension N : Use with '–data' for N–dimensional printout

E.G.,  "structmap 'structmap –scan'"  displays everything
```

When you invoke **structmap** with the name of a structure, it displays the name of the include file where the structure is defined and a description of each element of the structure — its offset from the beginning of the structure, its size, the number of times it occurs, its data type, and name. For example, **structmap** returns the following information about the **tape_header_record** structure:

```
$ structmap tape_header_record

tape_header_record /usr/sigmet/iris/include/output.h
         0      12       1   struct structure_header hdr
        12      16      16   char stape_id[]
        28      16      16   char sitename[]
        44      12       1   struct ymds_time init_time
        56       2       1   SINT2 idrive
        58       2       2   char ipad58x2[]
        60       8       8   char sversion[]
        68     252     252   char ipad_end[]
       320
```

structmap shows that the structure is defined in **/iris/include/output.h** and contains the following elements:

- **hdr** is a structure of type **structure_header**, taking up the first 12 bytes.
- **stape_id** and **sitename** are arrays of 16 characters each, at offsets 12 and 28.

- **`init_time`** is a **`ymds_time`** structure, taking up 12 bytes starting at offset 44.
- **`idrive`** is a long integer at offset 56.
- **`ipad58x2`**, **`sversion`**, and **`ipad_end`** are arrays of 2, 8 and 252 characters, at offsets 58, 60, and 68, respectively.

The total size of the structure is 320 bytes.

When you invoke **`structmap`** with the **`–scan`** option, it lists the names of all the structure defined by IRIS.

```
$ structmap –scan
ant_manual_setup
bitex_field_def
bitex_top_def
cappi_psi_struct
.
.
.
```

You can also use the **`–scan`** option to recursively call **`structmap`** and display the format of all the structures in the system. The following command takes this output and redirects it to a file:

```
$ structmap ‘structmap –scan‘ > allstructs.out
```