The Automated Satellite Data Processing System

Core Libraries

Naval Research Laboratory, Stennis Space Center, MS
Paul Martinolich
Theresa Scardino

The Automated Satellite Data Processing System: Core Libraries

by Naval Research Laboratory, Stennis Space Center, MS, Paul Martinolich, and Theresa Scardino

Publication date 18 November 2009

Table of Contents

I. User's Guide 1
- 1. Introduction to the Automated Processing System Concept 2
  - Understanding the APS 2
- 2. Getting Started 3
  - What You Need to Know to Get Started 3
  - System Requirements 3
  - Installation 3
    - APS Setup Window 3
    - Environment Variables 3
    - Database Variables 4
    - Processing Parameters 4
    - File Parameters 4
- 3. Using the APS 5
  - Defining Your Interest 5
  - Waters Within the Gulf of Mexico 5
  - Controlling the Projection Characteristics of the Output 6
    - Setting The Parameters 6
  - Creating the Map 7
    - Run the Program 7
    - Writing the Areas Script 10
  - Selecting Our Products 10
  - Selecting Browse Products 10
  - Testing Our Script 11
- 4. Automation 13
  - How The APS Works 13
    - Directory Structure 13
    - Locking Mechanism 13
    - Script Monitoring 14
  - Adding an Area 14
  - Processing Data 15
  - APS Directory Structure 15
- 5. APS Site Configuration 17
  - Use of apsPreProcess and apsPostProcess 17
  - Interfacing with a TeraScan System 17
  - Adding a 30m bathymetry line to images 17
  - ASCII Dumps of Products 18
  - Create high-resolution images for Real-Time Display 18
  - Define ranges of sst products 19
  - Changing the Logo and text string on the browse image 20
  - Extracting Data at a Point of Interest 20
  - Extracting Time Series at Region(s) of Interest 22
  - Reading APS products in SeaDAS 23
  - Setting up a PostgreSQL database 23
  - Clusters (using Ruby) 24
  - Sun Grid Engine 25
  - Data Extraction from L2 files 26

II. Data Format - Version 3.2 28
- 6. The APS Format 29
  - Standard APS File Attributes 29
- 7. Level 3 Regional Data Products 31
  - Naming Convention (Newer) 31

Naming Convention (Older) 32
File Attributes 33
Special Data Sets 34
Product Data Sets 36
8. Level-4 Regional Data Products 39
File Attributes 39
Special Data Sets 39
Product Data Sets 40
III. Command Line Reference 41
aps 42
daylight 47
filefmt 48
gregor 51
hdf 52
maps 55
apsScripts 65
apsDownloadMailFilter 71
apsDownloadManager 73
apsDownloadSearch 75
FileTransferDB 78
9. Release Notes 82
Release v3.8.2.3-72-g7b866d 82
Release v3.8.2 82
Release v3.8.1 82
Release v3.8 84

List of Figures

2.1. APS Setup Window .....
3.1. APS Map Example .....
4.1. The Execution Process .....

List of Tables

5.1. Default Point Files 21
5.2. Areas directory structure for apsSGE.sh 25
6.1. File Attributes 29
6.2. FileType Values 29
6.3. Time Attributes 30
6.4. Sensor Attributes 30
7.1. Input Parameters Attributes 33
7.2. Navigation Attributes 33
7.3. Input Geographical Coverage Attributes 34
7.4. Geographical Location Datasets 35
7.5. Data Quality Datasets 35
7.6. l2_flags Bit Flag Values 35
7.7. Product Data Set Attributes 37
8.1. Input Parameters Attributes 39
15. Spheroids 60
9.1. Processing Version 83
9.2. Processing Version 85

List of Examples

1. 77

Part I. User's Guide

The chapters in Part I form a User's Guide which describes the Automated Processing System.

The Automated Processing System (aps) is a collection of UNIX programs and shell scripts designed to generate map-projected image data bases of satellite derived products from a large flow of raw satellite input data in an automated fashion. Individual scenes are sequentially processed from the raw digital counts (Level-1) using standard parameters to a radiometrically and geometrically corrected (Level-3) product within several minutes. It further processes the data into several different temporal (daily, 8-day, monthly, yearly, and latest pixel) composites or averages (Level-4). These products are stored in the Hierarchical Data Format (HDF) with aps specific attributes. Additionally, it automatically generates quick-look “browse” images in JPEG format and may populate an SQL database using PostgreSQL.

The aps was designed for the Naval Research Laboratory Remote Sensing Applications Branch at Stennis Space Center, MS (now the Ocean Sciences Branch) to handle the continuous stream of satellite data. Originally, the system was designed to produce sea surface temperature maps from data collected by the Advanced Very High Resolution Radiometer (AVHRR). Data from the AVHRR is received daily (up to six passes per day). The aps provides for near real-time processing with the option of reprocessing historical data. The aps has since been upgraded to process ocean color satellite data from MERIS, MODIS (Terra and Aqua), OCM, and SeaWiFS.

The aps uses a simple monitoring technique, which has been found to be fairly reliable. The main driver regularly polls a specified input directory for incoming data and, for each found, executes what are known as `areas` scripts on the file in a working directory. After each area script has been run on the file, it is moved to an output directory. This method uses the directory as the queueing system for data to be processed. The areas scripts do the actual construction of the desired results (i.e. the data bases).

The 3.8.2.3-72-g7b866d version represents the processing algorithms employed at the Naval Research Laboratory as of 18 November 2009 for the SeaWiFS, MERIS, MODIS, OCM, and AVHRR sensors. The system has been developed on CenTOS 5.3 (i386, x86_64)). Secondary platforms include CenTOS 4.4 (i386), Mac OS X 10.5 (PowerPC and Intel), and Solaris 2.10 (i386).

Chapter 1. Introduction to the Automated Processing System Concept

The Automated Processing System (APS) is a collection of programs designed to allow scientists to generate co-registered image databases of geophysical parameters derived from remotely sensed data. The system is characterized by: *extension* and *automation*.

Extension is the use of small programs, each designed to address a specific task, and a shell to 'glue' them together. This idea is similar to the UNIX operating system and its many programs like **cat**, **tr**, **basename**, etc. and allows the user to augment the system with their own features and programs.

Automation is the technique of making a system operate without human effort or decision. For the APS, it is achieved by setting up a directory structure and using a script to monitor the directories for new input data. As new data is made available to the system, it is processed - all without the user's intervention.

The APS does not contain any GUIs or visulization programs. Therefore, to address these characterizations, all user input *must* be provided to the program upon *start*.

Understanding the APS

The APS is not a shell language or generic batch processor. It is a collection of programs used to process satellite data. Specifically, the system contains programs for:

- atmospheric correction
- geometric correction
- image processing
- compositing

Each of these programs provides the scientist with the tools to manipulate and study satellite data.

Additionally, the system includes programs that allow the above scientific programs to be used in an unattended way. These include functions to:

- query data files for information
- monitor the system for new data

Chapter 2. Getting Started

What You Need to Know to Get Started

Before starting to use the APS, you should be familiar with your UNIX environment, BASH shell programming, and remote sensing, especially with regard to computer processing of satellite data.

System Requirements

The system requirements are difficult to guage. The amount of memory needed is dependent upon the amount and type of satellite data you wish to process. The larger the area, the larger the requirements. For example, the entire Atlantic Ocean will require more processing power than the Mississippi Bight. Plus, the type of data being processed will determine how robust the system should be.

As only a guideline, the Naval Research Laboratory processes approximately 15 regions of MODIS, MERIS, AVHRR, SEAWIFS, and OCM real-time data on a quad processor. APS was developed on CentOS 4.4 (AMD Athlon and Opteron). Porting efforts have been made for Mac OS X 10.5 (Leopard) on the PowerPC and Intel, and Solaris 2.10 (i386).

Installation

> **Note**
> If APS has already been installed, please skip to the next section.

Once the system requirements have been met, you should be ready to install APS. APS is distributed different ways. Some users may simply FTP the tarfile while others use an installation CD. No matter how you've received it, copy the tarfile into your `home` directory. Extract the file using the `tar -xvf` command, and then begin the program.

To begin the program, run the `aps.rb` executable. It is located in the `/aps_3.8.2.3-72-g7b866d/bin/` directory. When the APS interface starts you will first see the Setup window. APS uses a default setting for everything included in this window. However, your installation may have some differences that need to be addressed before running APS. Here we will review the APS Setup window and look at what each field means.

APS Setup Window

APS requires a specific directory structure setup to run correctly. The top-level directory is `aps_v3.8.2.3-72-g7b866d`. Under this directory are the `in, work, out, bin, data, err, ftp, areas,` and `src` subdirectories. Since every directory is used by the system, the default values are generated. However, the Setup window allows the user to change that according to his/her unique system setup. This becomes very important when the user is not the owner of the installation executable. Usually, the default varaibles and parameters are acceptable. Simply review the window, make the appropriate changes and select `Save` from the `File` menu.

Environment Variables

The environment variables keep track of the directory locations. These variables are defined based on the user's perspective, which one installation might have multiple users.

APS_DIR: Top-level directory of APS.
APS_ETC: Site configuration files.
APS_BIN: Location of APS binaries.
APS_LIB: Location of APS libraries and scripts.
APS_AREAS: A user owned directory that stores the executable scripts for APS.
APS_IN: APS watches this directory for incoming data. This directory is important for automation purposes.
APS_WORK: While APS is processing it will use this directory to work in.
APS_OUT: The completed products will be in this directory.
APS_ERROR: If an error is thrown, this directory will contain the log messages.
APS_DATA: The data for APS.
APS_VAR: Variable data.
APS_VAR_DATA: Dynamic data that requires many updates, like calibration files.
APS_LOCK: A common directory where most APS lock files will be found.
APS_LOG: A common directory that holds the log files.

Database Variables

APS_DATA_BASE: Allows browsing of the rs directory for data.
APS_L1_DATA_BASE thru APS_L5_DATA_BASE: Output data locations with levels of 1-5 data, respectively.
APS_IMAG_BASE: Allows browsing of the browse directory for data.
APS_L1_IMAG_BASE thru APS_L5)IMAG_BASE : Output data locations with levels of 1-5 data, respectively.

Processing Parameters

Verbosity: The default is set to '1', which means 'on'. This tells the system to report more information out as it runs and be verbose.
Number of Threads: The default is set to '1'. Typically, this number is set to the number of CPUs on your system due to the multi-threading capability of the CPU.

File Parameters

Log File: The location of the log file.
PID File: The location of the PID (Process Identification) file.
Process File: The location ofthe process file.
Pre-Processing Script: The location of the apsPreProcess.sh file.
Post-Processing Script: The location of the apsPostProcess.sh file.

Chapter 3. Using the APS

To get started using the APS, we will write an areas script in this chapter. This script will be installed into the APS in Chapter 4, *Automation*. The reason for some code constructs will become clear in that chapter.

In this section, we will create a BASH shell script for the creation of chlorophyll a image maps for waters within the Gulf of Mexico. It is assumed that the APS has been installed at your site. If the system has not been installed, see the section called “Installation” . Furthurmore, it is assumed that APS was installed in `/home/aps/aps_v3.8.2.3-72-g7b866d` and that the variable `APS_DIR` is set to `/home/aps/aps_v3.8.2.3-72-g7b866d`.

Defining Your Interest

Before writing a script, we must decide what it is we want to create. In this example, the objective is to create an image data base of chlorophyll a concentration using two different algorithms for waters within the Gulf of Mexico. We wish to compare the OC2 algorithm with the OC4 algorithm in both open ocean and coastal waters. We plan to use the data from the SeaWiFS sensor.

Waters Within the Gulf of Mexico

First, we will define our region of interest. For this example, it will be defined as the waters of the Gulf of Mexico. Using a USGS or National Geographic map of the Gulf of Mexico, we decide to define this region as the area within the following geographical boundaries:

- The upper left corner shall be 31 N and 98 Wn
- The lower right corner shall be 17 N and 80 W

Because we plan to do spatial and temporal variability studies of the loop current in the Gulf of Mexico (and maybe make a nice movie loop of several days worth of images), we need each scene to be registered to a standard map projection. We use the term *image map* to indicate a specific map projected image. An *image map* is the combination of a map projection and an output image. Each pixel's relationship to its neighbors is defined by the projection used. In this example, we want our image to be a full-resolution image of chlorophyll a represented as a Mercator projection. The Mercator projection as one of thirty available map projections provided by the General Cartographic Transformation Package. The GCTP package was obtained from the United States Geological Survey.

One group of outputs generated by the Level-2 APS programs is collectively termed a control points structure. This structure allows a program to determine the geographical coordinates of any pixel in the original scene by tying down specific points in the satellite scene (sample,line) to geographical coordinates (longitude,latitude). These points are commonly called Ground Control Points (GCP). In the APS, the structure forms a regular grid across the image. That is, the for each column and row in the input image, a known latitude and longitude pair is given. Thus, the column or pixel locations as well as the row or line locations are given as 1-dimensional arrays. The pixel and line locations are stored in separate arrays. The latitudes and longitudes at the intersections of the row and columns are stored in 2-dimenional arrays. The latitudes and longitudes are also stored in separate arrays. This information is used by the geometric registration program.

The APS includes a geometric registration program known as **imgMap**. This program moves the pixels or data from the input file to newer locations in the output file. This process is known as geometric registration or “warping”. The translation from input pixel to output pixel is defined by the *image map* and the control points structure. The results of the pixel movement will depend up the accuracy of the input control points

structure. Additionally, the parameters unique to the selected map projection will effect the accuracy in scale of the resulting image. For example, the Mercator map projection will exaggerate land mass as we reach the poles. That is, Greenland will appear much larger in size than it actually is.

Controlling the Projection Characteristics of the Output

The *image map* parameters determine the projection of the output image as well as of the imaginary map of the world of which it is a part. Although the image is projected onto that map, the input parameters allow the user to control where the window will be positioned over the map as well as the scale of the map.

Setting The Parameters

The first two required parameters will associate a longitude and latitude (LL_1) from the world to a particular location on the output image (XY_1). The XY_1 parameter refers to the image coordinates (sample and line location) of a point in the image whereas LL_1 refers to the geocoordinates (longitude and latitude) of a point on the world map. The points will be associated so that the image point (XY_1) overlays the map point (LL_1). It may be convenient to choose LL_1 to be the center of the image by assigning XY_1 to the first coordinate :number of samples/2, and the second coordinate: number of lines/2. Alternately, the user may wish that a certain landmark appear at a certain location in the image. In such a case, the landmark's geocoordinates would be entered for LL_1 and the desired image location specified by XY_1.

The next two parameters will determine the scale of the *image map*. The first parameter is a second point on the world map (LL_2) and the second parameter is a distance (in pixels) away from the first (DELTA). The use of LL_2 and DELTA in conjunction with LL_1 and XY_1 controls the scale of the map and, hence, also controls how much of the mapped image appears within the image. LL_2 represents the geocoordinates of another point on the world map and DELTA represents the separation in pixels between that point and the XY_1 image location. A positive DELTA represents a horizontal separation, whereas a negative value represents a vertical separation.

Note that this second point need not be within the image and that the absolute value of DELTA may be larger than the image width or height. For given parameters, a larger absolute DELTA will decrease the geographical area covered by the window (enlarge the map or *increase* the scale); a smaller absolute value will increase this area (contract the map or *decrease* the scale). The direction of the second point relative to the first - that is, where they both fall on the world map - is determined solely by the projection.

Although DELTA represents the separation in either the horizontal and vertical direction (not the absolute separation), the points for LL_1 and LL_2 must be chosen such that they have both a horizontal *and* vertical separation on the imaginary map. Therefore, some *a priori* knowledge of where these points will fall on that map is required when choosing these parameter values.

A convenient way to determine the value for LL_2 and DELTA is to use the geocoordinates of another landmark for LL_2 and enter the desired separation between LL_1 and LL_2 for DELTA. Another convenient way to determine these values is to determine the scale for the map at LL_1. (In certain cases, depending on the projection, the scale will vary greatly even within the image area.) That is, the user decides how many image pixels (DELTA) should separate a longitudinal or latitudinal degree and assign LL_2 accordingly. For example, if the scale at LL_1 is to be one latitudinal degree per 100 pixels and LL_1 is 10 degrees latitude and 38 degrees longitude, LL_2 would be 9 and 38 degrees and DELTA would be -100 (assuming that north is on the top of this projection).

The example below may help understand this idea. The window is represented by the red box with the points labeled to show XY_1, LL_1, and LL_2. The arrows show the DELTA.

Creating the Map

In this example, we align the North West corner (LL_1) with the upper left corner of the image (XY_1). Thus, LL_1 will have a value of 98 W and 31 N and XY_1 will have a value of sample 1, line 1. (For images produced by **imgMap** the origin of the image is located in the upper left. In other words, we count our lines down from the top and our samples (or columns) from the left. So, line 1 is at the top of the image; line n is at the bottom. Pixel 1 is on the left edge of the image; pixel n is on the right edge.)

To define the second point (which does not have to exist in the output image), we need to set the DELTA component. This component is the horizontal or vertical separation in pixels on the projected image between the LL_1 (the NW corner) and LL_2. This parameter, along with the latitude and longitude of the second point, define the resolution of the output image. To maintain a 0.01 degrees/pixel resolution in the longitudinal direction, we set DELTA to 1799. If pixel 1 corresponds to 98 W and pixel 1800 corresponds to 80 W, then there are 1800 - 1, or 1799, pixels between them.

For the time being we will assume that the scale in the latitudinal direction is the same as in the longitudinal direction, though for the Mercator projection this is certainly *not* true. Since we want to place LL_2 near 80 W and 17 N, we will start by making a temporary map with 1400 lines because the Northern and Southern latitude differ by 14 degrees and we have a scale of 0.01 degrees/pixel.

Once the map has been defined we will use the **toll** command to get a better approximation. Remember that LL_1, XY_1, LL_2, and DELTA define the image to the imaginary map with LL_1 and XY_1 being tied together. The size of the window (our output image) over the imaginary map can be different in size

Run the Program

The program **maps** is used to create the *image map*. We give the mapped image a name and store the parameters in an HDF file for later use. We call this file *image map*. Note that in the example below, we set the central meridian and latitude of true scale to the middle of our projection and that we use the WGS 84 datum (12). See maps(1) for a complete list of possible map projections and datums.

So we begin by running the **maps** program and attempt to create our first *image map*:

```
$ cd
$ pwd
/home/aps/
$ maps
maps>create temp
Full Name? [ temp ]
Code (3-letter)? [ temp ]
Map Projection (? for list)? [ Mercator ]
Size of Map (w h)? 1800 1400
Pt1 of Map (lon lat)? -98.0 31.0
Pt1 of Map (x y)? [ 1 1 ]
Pt2 of Map (lon lat)? -80.0 17.0
Delta (pixels)? [ 1799 ]
Aspect ratio? [ 1.0 ]
Datum (? for list)? [ WGS 84 ]
SMajor? [ 6378137.000000 ]
SMinor/Eccentricity? [ 6356752.314245 ]
Longitude of Central Meridian (lon)? -89.0
Latitude of True Scale (lat)? 24.0
False Easting? [ 0.0 ]
```

```
False Northing? [ 0.0 ]
maps>setmap temp
maps>toll 1 1
-98.000000 31.000000
maps>toll 1800 1400
-80.000000 18.241954
maps>toxy -80.0 17.0
1800.000000 1529.447826 - off map
```

From this last information, we will increase the size of the image (that is, the *window* over the imaginary map) to 1530. So to continue:

```
maps>create Gulf
Full Name? [ Gulf ] Gulf of Mexico
Code (3-letter)? [ Gulf ] GOM
Map Projection? [ Mercator ]
Size of Map (w h)? 1800 1530
Pt1 of Map (lon lat)? -98.0 31.0
Pt1 of Map (x y)? [ 1 1 ]
Pt2 of Map (lon lat)? -80.0 17.0
Delta (pixels)? [ 1799 ]
Aspect ratio? [ 1.0 ]
Datum (? for list)? [ WGS 84 ]
SMajor? [ 6378137.000000 ]
SMinor/Eccentricity? [ 6356752.314245 ]
Longitude of Central Meridian (lon)? -89.0
Latitude of True Scale (lat)? 24.0
False Easting? [ 0.0 ]
False Northing? [ 0.0 ]
maps>setmap
Name? Gulf
maps>toll 1800 1530
-80.000000 16.994684
maps>delete temp
maps>list
Gulf
maps>save gulf_map.hdf
maps>quit
$ ls gulf_map.hdf
gulf_map.hdf
$ hdf gulf_map.hdf list
File: gulf_map.hdf

File Attributes: (4)

createTime = "Fri Aug 18 20:04:43 2006"
createSoftware = "APS v3.8.2.3-72-g7b866d"
createPlatform = "i686-pc-linux-gnu"
createAgency = "Naval Research Laboratory, Stennis Space Center"

Data Sets: (1)

projection_t Gulf
```

```
Name         Gulf
Full Name    Gulf of Mexico
Code         GOM
Projection    5 (Mercator)
Zone         62
Datum        12 (WGS 84)
Parameters:
 0:  6378137.000000 (Semi-Major Axis)
 1:  6356752.314245 (Semi-Minor Axis)
 2:  0.000000
 3:  0.000000
 4:  -89000000.000000 (Longitude of Central Meridian)
 5:  24000000.000000 (Latitude of True Scale)
 6:  0.000000 (False Eastings)
 7:  0.000000 (False Northings)
 8:  0.000000
 9:  0.000000
10:  0.000000
11:  0.000000
12:  0.000000
13:  0.000000
14:  0.000000
Width        1800
Height       1530
Longitude_1  -98.0000
Latitude_1    31.0000
Pixel_1      1
Line_1       1
Longitude_2  -80.0000
Latitude_2    17.0000
Delta        1799
Aspect       1
```

To process this region copy the newly created map file `gulf_map.hdf` to the APS `data` directory. By default, the APS has a default map file to hold all known maps named `maps.hdf` It resides in the `data` directory. To use the default location, you can copy the `Gulf` map from the `gulf_map.hdf` file to the `maps.hdf` file. If not, then your scripts you will have to set the `MapFile` variable so that APS will know which file to look at to find the `Gulf` projection information.

Note

See reference page for **maps** for additional examples.

To copy the `gulf_map.hdf` file to the data directory use the command (remember we will have to add `MapFile` to our script):

```
cp gulf_map.hdf ~/aps_v3.8.2.3-72-g7b866d/data
```

To copy the `Gulf` area from the `gulf_map.hdf` file to the standard APS maps file (`maps.hdf`), we will use the APS provided program **hdf**.

Note

The name we select should not already be in the `maps.hdf`.

```
hdf gulf_map.hdf copy ~/aps_v3.8.2.3-72-g7b866d/data/map.hdf Gulf
```

Writing the Areas Script

Now that the projection information is done, we can write an areas script that will process data. The script must be a UNIX text file, so use your favorite text editor (vi, emacs, nedit, jot, etc.). If you use an editor that automatically creates backups, then be sure to delete that file from the `areas` directory. In this particular case, we are going to create our script in our home directory and use the `gulf_map.hdf` directly from there also. The script, which we save is `SwfGulf`, looks like:

```
#!/bin/ksh
 . $APS_DIR/lib/aps/apsScripts.sh
 . $APS_DIR/lib/aps/swfScripts.sh
MapName=Gulf
MapFile=$HOME/gulf_map.hdf
MapExt=GOM
swfProcess $1 $0
```

Note the `$1` in the last argument. It is required. The second argument `$0` is optional, but is very useful so add it.

Additionally, note that we have added the `MapFile` line, because we decided, in this example, to copy our file (`gulf_map.hdf` to the `data` directory.

Selecting Our Products

Looking up the man page for **l2gen** we see that this program can produce the chlorophyll products we are seeking. The documenation on **l2gen** indicates that the products are named: chl_oc2 and chl_oc4.

If this study is specific only to the Gulf of Mexico, then we only need to list them in the `SwfGulf` script. To do that, we define a BASH shell variable called `L3ProdList`. In this case, our script now looks like this:

```
#!/bin/bash
 . $APS_DIR/lib/aps/apsScripts.sh
 . $APS_DIR/lib/aps/swfScripts.sh
MapName=Gulf
MapFile=$HOME/gulf_map.hdf
MapExt=GOM
L3ProdList="chl_oc2 chl_oc4"
swfProcess $1 $0
```

A second option is to specify the desired products by modifying the default product file. The default file is `$APS_DATA/seawifs/seawifs_def_l2prod.dat` . This file, however, is *global* to APS. That is, for all scripts that do not define `L3ProdList`, these products will be produced.

Selecting Browse Products

To create browse images, we must select our images by setting the `L3BrowseList` variable. Or, as above, we can modify the default browse file (`$APS_DATA/seawifs/seawifs_def_l2browse.dat`). Our script will now look like this:

```
#!/bin/bash
 . $APS_DIR/lib/aps/apsScripts.sh
 . $APS_DIR/lib/aps/swfScripts.sh
MapName=Gulf
MapFile=$HOME/gulf_map.hdf
MapExt=GOM
L3ProdList="chl_oc2 chl_oc4"
L3BrowseList="chl_oc2 chl_oc4"
swfProcess $1 $0
```

By default, this version of APS will automatically generate temporal composites. These are: daily, weekly, monthly, and yearly. The weekly composites are actually 8-day composites. As with the Level-3 products, the APS has default files for both the Level-4 products and their quick-look images. The level-4 products are stored in the file `$APS_DATA/seawifs/seawifs_def_l4prod.dat`. The level-4 browse products are stored in the file `$APS_DATA/seawifs/seawifs_def_l2browse.dat`. To override these options, we add two additional lines to our script. Our script will now look like this:

```
#!/bin/bash
 . $APS_DIR/lib/aps/apsScripts.sh
 . $APS_DIR/lib/aps/swfScripts.sh
MapName=Gulf
MapFile=$HOME/gulf_map.hdf
MapExt=GOM
L3ProdList="chl_oc2 chl_oc4"
L3BrowseList="chl_oc2 chl_oc4"
L4ProdList="chl_oc2 chl_oc4"
L4BrowseList="chl_oc2 chl_oc4"
swfProcess $1 $0
```

The above will give us enough to process the data and product results. Next we'll obtain a SeaWiFS data file, uncompress it and use it as input to our script and examine the results. If we are happy, we'll then place the script into the areas directory.

Testing Our Script

The script we just created will be called by the APS as if we typed the following command:

```
SwfGulf S2001005175131.L1A_HNAV
```

Since, the script must be executable, we run the command:

```
chmod 755 SwfGulf
```

In fact, you can test your script prior to placing it in the APS by (under the BASH shell) doing the following commands:

```
$ . ~/aps_v3.8.2.3-72-g7b866d/etc/aps.conf
$ ./SwfGulf S2001005175131.L1A_HNAV
SwfGulf    : checking if file covers gulf ... yes.
SwfGulf    : converting L1 to L2 ... done.
SwfGulf    : converting L2 to L3 ... done.
```

```
SwfGulf    : making browse imagery ... done.
SwfGulf    : adding products to data base ... done.
SwfGulf    : running composites ... done.
SwfGulf    : completed.
```

After this completes, we can examine the browse images by changing to the directory created by APS for this. If we assume that the `APS_IMAG_BASE` variable was set to `/home/aps/browse` in the `aps.conf` file, then we can examine the Level-3 JPEG files with (or use any other graphics display program):

```
$ cd ~/browse/lvl3/seawifs/5.4/Gulf/2001/jan
$ kview S2001005175131.L3_HNAV_GOM_chl_oc2.jpg
```

Where did that directory come from? APS creates it automatically. By default, APS will put all the HDF product files in the `rs` directory and the JPEG images in the `browse` directory. If the defaults were selected during the installation, this would be subdirectories of your home account. That is, `/home/aps/rs` and `/home/aps/browse`.

The remaining sublevels are created automatically by APS. They follow the pattern: `level/sensor/version/area/year/month/`. This pattern is true for all Level-3 data (HDF or browse images). For Level-4, the basic pattern is the same, except we add a name to for the temporal nature of the composite and delete some subdirectories that are not needed. So, for Level-4, the directories are created for example: `lvl4/seawifs/5.4/Gulf/daily/2001/jan`, `lvl4/seawifs/5.4/Gulf/weekly/2001`, or `lvl4/seawifs/5.4/Gulf/yearly`.

You will notice that the SeaWiFS data is stored in a `5.4` directory. This version is a *algorithm processing* version and may or may not reflect that software version number. This was done because the APS can handle many different sensors, whose processing algorithms may or may not change with the software versions. For APS v3.8.2.3-72-g7b866d, the SeaWiFS data is at version 5.4, the MODIS data is at version 5.4, the OCM data is at version 1.4, the MERIS data is at version 1.4, and the AVHRR data is at version 4.0.

Thus far, we have been able to use the APS software to run the standard SeaWiFS processing for our area of interest. If we have only a few files, it would make sense to run the above commands for each SeaWiFS file. However, the strength of APS is that we don't just have a few files - we have hundreds or thousands of files. Or, we have a receive station that is continually capturing more data. Then running them by hand does not make much sense. The next section will show how to start processing the data automatically.

Another option for users who do not require automation (because you order and download a bunch of files from the DAAC and then just process them), is the **repro** option for the **aps** script. This option allows the user put a list of files into a UNIX text file and process only those files and areas that they wish.

Chapter 4. Automation

This section will take what you learned in the previous Chapter 3, *Using the APS* and automate it using the Automated Processing System. Here you will learn:

- how the automated processing system works
- how to process new areas

How The APS Works

The APS system uses three concepts to allow for automation.

- directory structure
- locking semantics
- script monitoring

Directory Structure

For directory structure, the APS requires several directories that are similar to any office worker's desk. There is a basket labeled `in' where letters, orders, etc. are placed for the worker to process. There is a work area, usually the desktop, where the worker processes the orders, letters, etc. using different protocols depending on what regions, or areas of the country the order covers. And finally, there is a basket labeled `out' which holds the completed orders.

These three directories are the heart of the system. But other directories are required. A worker may have a file cabinet which he must consult to obtain tables of information or data to process the orders. He may also need various tools, which he keeps in a box, or bin, to process the orders. If an order is incomplete or wrong, he might have a basket labeled `err' to hold these until he can consult with the boss for corrections.

Finally, if a really big order is coming from a colleague down the hall, the colleague might need a temporary place to put portions of the order before placing the complete order in the `in' basket. Several trips down the hall may be required by the colleague to get the complete order. If he places an incomplete order in the `in' basket, the worker might start trying to work on it before it is complete. The worker would throw it out!

In terms of the APS, the worker's office is located in the top-level directory (usually `aps_v3.8.2.3-72-g7b866d`). The subdirectories of this directory are: `in`, `work`, `out`, `bin`, `data`, `err`, `ftp`, `areas`, and `src`.

The basic flow of data (that is, the physical *movement* of the actual data files) is shown in Figure 1. *Of special note, is the requirement that the `ftp` and `in` directories must reside on the same disk.* It is possible to set up the directory structure such that the other directories are on seperate disks. However, normally all directories exist on the same disk.

Locking Mechanism

In order to facilitate parallel processing on multi-processor systems and on clustered systems, APS implements an exclusive locking mechanism.

The APS locking mechanism is implemented using lock files. The existence of a file with a certain name signifies that an instance of APS is holding that lock. Lock file names are generally created by appending `.lock` to the current filename being processed. In order to prevent contention, the directory in which the lock files are created must be shared by all instances of APS that will share the same `APS_IN` directory.

The environment variable APS_LOCK is used to control the location of the lock directory and by default is set to the APS_WORK directory. This may be changed in the APS configuration file.

Locking is necessary to prevent two instances of APS from corrupting each others data. For example, if two instances of APS are processing two different MODIS data files covering the Gulf Of Mexico region for the same day, and they both attempt to create a mosaic of the day's data, one instance could corrupt the file created by the other instance. This situation is handled by creating a lock file associated with the mosaic file. The first instance of APS would create the mosaic lock file and then create and store the mosaic file. The second instance of APS would recognize that a lock file already exists, and wait until the first instance of APS is finished before continuing with its task of creating the mosaic file.

Most APS locks will reside in the APS_LOCK directory, but there are a few that will exist outside of that directory. Currently there are two main locations outside of the APS_LOCK directory where lock files may be found: APS_DATA/seawifs/nav/ and APS_DATA/modis/reynolds2/. The file elements.dat will be locked with the file APS_DATA/seawifs/nav/elements.dat.lock to prevent two instances of APS from updating and overwriting the elements.dat file at the same time. The MODIS reynolds files may each be locked independently when an instance of APS attempts to retrieve the respective reynolds file.

There is one other special lock that is not associated with any particular data file. aps.lock will reside in the APS_LOCK directory and is a generic lock currently only used when APS instances are accessing the APS_IN directory.

The Locking System complicates the restart procedure since stale locks must be checked for and removed manually. This cannot be done automatically since locks may be owned by APS instances running on other machines or by APS reprocessing jobs. Before APS is started the APS_LOCK directory, the APS_DATA/seawifs/nav/ and the APS_DATA/modis/reynolds2/ directories should all be checked for existing lock files. The hostname of the machine that APS instance that created the lock is inside the file. If the APS instance that created the lock file is no longer running, the lock file should be removed.

Script Monitoring

The worker of the APS is the **aps.sh** script. Once started the script will examine the in directory for data files, process a single data file for each script in the areas directory, and place the data file in the out directory. After processing all files in the in directory, the script goes to sleep for one minute, after which it performs the above steps again. The script will do this forever. See aps(1) for more information.

Figure 4.1, “The Execution Process” shows how the execution and Level-1 data file locations correspond. Before, the input Level-1 data file is moved from the in directory to the work, a *optional* pre-processing script may be executed. This provides an oppurtunity for the Automated Processing System to do some work on the file before all the areas scripts are run.

Next, each script is executed sequentially on the input data file which now resides in the work directory. This allows the various scripts to create temporary files in a clean place. The scripts run various programs located in the bin directory, which in turn may read various data files located in the data directory.

Finally, the Level-1 data file is moved either to the err or out directory depending on whether any of the scripts failed abnormally. If the execution was flawless, then the Level-1 data file is moved to the out directory where the optional post-processing script will be run. Usually, this post-processing script is designed to remove the Level-1 file from the out directory and store it in some type of archival system.

Adding an Area

In the section called “How The APS Works”, we developed the SwfGulf script to help us generate an image database of the Gulf of Mexico. That script required a SeaWiFS Level-1A data file as input.

We will add this map to the default file containing all maps, `maps.hdf`, which is located in `APS_DATA` directory.

```
$ hdf gulf_map.hdf copy $APS_DATA/maps.hdf gulf
```

To start processing the Gulf of Mexico, simply copy or move the script into the areas directory *and give it execute permissions*. As long as the script is located here, the APS will process incoming data for that area. To stop processing the Gulf of Mexico, you can move the script to another directory, delete it, or make the script non-executable. For example:

```
$ cp SwfGulf $APS_DIR/areas
```

will add the `SwfGulf` script to the area directory. To remove it we may do:

```
$ cd $APS_DIR/areas
$ chmod 644 SwfGulf
```

or

```
$ cd $APS_DIR/areas
$ mkdir extra_areas
$ mv SwfGulf extra_areas
```

Processing Data

At NRL Remote Sensing a SeaWiFS receiving station is used to capture every satellite pass over the Gulf of Mexico. This yields a steady stream of one or two passes (80 MBs of data) a day. After a pass is captured, it is converted to a NASA Level-1A SeaWiFS data file and moved into the `in` directory of another NRL host. The APS is running on this second host. Once the APS *sees* the pass in the `in` directory it is processed.

To process new SeaWiFS data for the Gulf of Mexico, one may simply move the SeaWiFS data files to the `in` directory. However, the aps also provides another method for reprocessing. See aps(1) for more information.

APS Directory Structure

The following is a pass over the Gulf of Mexico. This yields a steady stream of one or two passes (80 MBs of data) a day. After a pass is captured, it is converted to a NASA Level-1A SeaWiFS data file and moved into the `in` directory of another NRL host. The APS is running on this second host. Once the APS *sees* the pass in the `in` directory it is processed.

```
aps_v3.8.2.3-72-g7b866d/
.../bin
.../lib
.../lib/aps
.../lib/ruby
.../include
.../share
.../share/aps
```

```
.../share/doc
.../share/doc/html
.../share/doc/html/api
.../share/doc/html/user
.../share/man/man1
.../data
.../data/browse/
.../data/common/
.../data/fonts/
.../data/hmodisa/
.../data/hmodist/
.../data/insitu/
.../data/meris/
.../data/modisa/
.../data/modist/
.../ocm/
.../seawifs/
.../etc
.../areas
.../areas/Operational
.../areas/Reprocess
.../in
.../in/Operational/priority_1
.../in/Operational/priority_2
.../in/Reprocess/priority_-1
.../in/Reprocess/priority_-2
.../err
.../work
.../out
.../var
.../var/log
.../var/lock
.../var/data
.../browse
```

Chapter 5. APS Site Configuration

This section will cover some of the other standard procedures deployed in a fully operational system, like changing your desired list of standard products, and changing the location of the database, etc. All site-specific files should go into the `APS_ETC` directory.

Use of apsPreProcess and apsPostProcess

By default, the APS was designed to take Level-1 data and pass it through all the areas scripts and place the Level-1 data in an output directory. Any Level-3 data produced by the respective areas scripts were placed into a database. However, being a receive center, we might also need to convert Level-0 data to Level-1. This conversion would take place so that the Level-1 data was left in the `in` directory and the Level-0 was thrown away. This was accomplished by using a **apsPreProcessing.sh** script.

Additionally, the **apsPreProcessing.sh** script was used to change the permissions on the incoming data so that the APS software was guarenteed to have read/write permissions on the data.

As just noted, the original plan for the Level-1 data was to just place it in the `out` directory. Unfortunately, this directory would eventually fill up the disk! So, the **apsPostProcessing.sh** script was installed to place the Level-1 data on some type of archive system. Our original archive was a tape backup system that has recently been replaced by very large RAID system. The **apsPostProcessing.sh** script would automatically build a standard directory structure based on the data type and time of the Level-1 data, and transfer it to the archive disk. Additionally, this script would compress the data.

Interfacing with a TeraScan System

To use the system in this manner, a post-processing script is still used, but its primary task is to copy the data from the receive station to the APS system. Included with the APS distribution are example scripts that can be found in the `share` directory. Of important note in these scripts is that (a) the AVHRR Level-0 data is converted to a NESDIS Level-1b using the TeraScan program **flac** and that (b) the SeaWiFS Level-0 data is transferred in TeraScan Level-0 format. In the case of (a), the APS system is set up to expect that type of input. In the case of (b), the APS system has the necessary software to convert the TeraScan Level-0 format to a NASA Level-0 format as required by the NASA Level-1A generating software. A user without a receive station will not be concerned with these matters.

The TeraScan system has a method of performing post-processing on the incoming data once it is captured. It would be possible to use the individual programs and shell scripts that make up the Automated Processing System and this post-processing mechanism to produce the output files needed. However, at NRL we prefer to have the TeraScan system dedicated to its number one responsibility - data reception. This host will then feed the data (via NIS) to a second system running APS.

Adding a 30m bathymetry line to images

The APS comes with a bathymetery file which can be used to overlay a 30m bathymetery line to all output images. The global `BATHY.DAT` is used to create an HDF file specific for the region of intertest. This HDF file is automatically created if it does not already exist and is placed in the `var/data/bathy` directory. The file name consists of the region name appended with `_bathy.hdf`. To add this overlay to the browse images, the `doBathy` should be set to 1. By default, the value is 0 and no bathymetery line is overdrawn.

ASCII Dumps of Products

The APS has the ability to create simple ASCII data dumps of products when the results are needed as input in another program. For example, the `sst` product may be used to initialize a model. To accomplish this, the APS provides a post processing function `ascii_dump`. The following example shows part of an area script that will extract the data.

The `ascii_dump` function takes at least two arguments: *`file`* and *`product`*. An optional third parameter will define the location of the output. By default, this parameter is `$ApsDataBase/ascii/$prod`. The `ascii_dump` function uses the APS program **imgDump** to perform the actual data extraction. The default format of `ascii_dump` is the 'Ko' format. This may be changed by using the `-f`. See the documentation for **imgDump** for other outputs.

```
. $APS_DIR/lib/aps/PostProcFuncs.sh
PostProcessFunc()
{
    #  Extract sst for model assimilation

    ascii_dump -r 9.0 45.0 $1 salinity
    if test $? -ne 0
    then
        apsLogMsg "$FUNCNAME(line $LINENO): ascii_dump failed for salinity"
        err=1
    fi

}
PostProcess=PostProcessFunc
```

For example, if the MODIS file `MODAMN2007200044000.L3_NOAA_GOM` is processed with the above script, then the output directory `/home/aps/rs/ascii/sst` will contain the Ko formatted file `MODAMN2007200044000.L3_NOAA_GOM_sst.dat`. The first few lines of that file are:

```
2007 07 19 04 41 30    21.52   -85.02  29.18
2007 07 19 04 41 30    21.51   -85.12  30.10
2007 07 19 04 41 30    21.51   -85.11  30.10
2007 07 19 04 41 30    21.51   -85.10  30.10
```

Create high-resolution images for Real-Time Display

The APS works with the Tactical Operations Display System (TODS) to merge satellite data with model data overlays. To produce the high-resolution satellite images required for this the script function `hrMakeBrowse` is used. This fuction is available in `PostProcFuncs.sh`.

```
    . $APS_DIR/lib/aps/PostProcFuncs.sh

#   make high resolution images from latest pixel composites
#   make high resolution true color image from daily image
    PostProcessFunc()
```

```
    {
        local err=0

#       set parameters

        hrL3BrowseList="true_color"
        hrL4BrowseList="chl_oc4"
        hrL3ImagBase='display@vanadium:/home/display/video_loop/$Sensor/$MapExt/'
        hrL4ImagBase='display@vanadium:/home/display/video_loop/$Sensor/$MapExt/'
        hrCopyMethod="scp"
        hrCopyOpts="-B -q"
        useMosaicoverlay="YES"
        doconvolve="YES"
        hrBrowseOpts="-1"

#       this function does all the work

        hrMakeBrowse $*
        err=$?

        return $err

    }
    PostProcess=PostProcessFunc
```

Define ranges of sst products

To optimize the sea surface temperature browse images, APS has several mechanisms to enhance these images. The first is the use of the `imgSetSSTRanges`.

This function is designed to be used as a PreBrowseProcess function to define the browse ranges for the 'sst' product using several different options.

For example, in AvhMissBight, we might have:

```
        SSTMinMaxByMonths[1]='12 28'
        SSTMinMaxByMonths[2]='14 28'
        SSTMinMaxByMonths[3]='16 28'
        SSTMinMaxByMonths[4]='19 30'
        SSTMinMaxByMonths[5]='20 32'
        SSTMinMaxByMonths[6]='24 32'
        SSTMinMaxByMonths[7]='24 32'
        SSTMinMaxByMonths[8]='24 32'
        SSTMinMaxByMonths[9]='24 32'
        SSTMinMaxByMonths[10]='24 29'
        SSTMinMaxByMonths[11]='22 29'
        SSTMinMaxByMonths[12]='18 26'

        PreBrowseProcessFunc()
        {
            imgSetSSTRanges $1
        }
        PreBrowseProcess=PreBrowseProcessFunc
```

to set the ranges based on month for the sst product.

To do more than on sst product (like 'sst4' and 'sst') then do this:

```
PreBrowseFunc() {
    imgSetSSTRanges $1 sst
    imgSetSSTRanges $1 sst4
}
PreBrowseProcess=PreBrowseFunc
```

The second method is to use the script function, `sst_enh` that is available in `PostProcFuncs.sh`. That script function will use the results of the **imgStat** to define the 5% histogram stretch of the sea surface temperature image.

```
. $APS_DIR/lib/aps/PostProcFuncs.sh
PostProcessFunc() {

    local err=0

    sst_enh $1 $L3BrowseDir $L3ThumbDir
    return $?
}
PostProcess=PostProcessFunc
```

Changing the Logo and text string on the browse image

The browse images contain a logo in the bottom right corner and also three text strings which can be user defined. The strings on the left can not be modifed. The logo displayed is read from the JPEG formatted file `$APS_ETC/logo.jpg`. The user should name their logo as such. The logo file should generally be small about 48 by 48 pixels. The APS software will not reduce or manipulate it in any way.

To define the three strings, the user can modify the UNIX ASCII file `$APS_ETC/imgBrowse.opt`. This file may have the standard shell script comments (that is, lines that start with #). The strings are defined using the keywords (`string1`, `string2`, and `string3`) for the top, middle, and bottom strings. The equal sign (=) that separates the keyword and its value should have not spaces between the two. The string does not have to be in quotes. For example,

```
# My image browse options
string1=UNCLASSIFIED
string2=Approved for Public Release
string3=Distribution Unlimited
```

Extracting Data at a Point of Interest

During normal data processing, APS has the capability to extract data from a point of interest and create simple ASCII text files that are continually updated as new data is processed. This can be useful for real-time match up with a stationary buoy like an AERONET site or a NDBC buoy.

To begin the user must create the `points` in the `$APS_ETC` directory. Here, the user would place the “points” files. There is a standard naming scheme `points_$MapName.pts` for the standard Level-3

data extraction. Since APS is designed to process multiple data sources and time frames, the user may also restrict the point exraction to a specific sensor or date, but creating files that are specific for them.

Table 5.1. Default Point Files

File	Description
`$APS_ETC/points/$MapName.pts`	Extract data for region regardless of sensor and time
`$APS_ETC/points/$MapName_$Sensor.pts`	Extract data for region and sensor regardless of time
`$APS_ETC/points/$MapName_$Year$DoY.pts`	Extract data for region and day of year disregarding sensor
`$APS_ETC/points/$MapName_$Sensor_$Year$DoY.pts`	Extract data for region and sensor and day of year

The format of the file is defined by that of the **imgRead** program. Basically, it contains two or three colums: Name Station/Point (optional), Latitude, Longitude. The latitude must in the range of 90.0 to -90.0. The longitude must be in the range of -180.0 to 180.0. Below is an example file (`MissBight.pts`) which contains the name of each point.

> **Note**
> If the name is given, the user must set `PTOptions` to include the `-s` which will be passed to **imgRead**.

```
$ more MissBight.pts
NDBC_42007 30.09 -88.77
NDBC_42040 29.18 -88.21
NDBC_42039 28.79 -86.02
```

The example below shows an example the area script.

```
#!/bin/sh
. $APS_LIB/apsScripts.sh
MapName="MissBight"
#must include -s since our files contain the names of positions
PTOptions="-s"
PTProdList="chlor_a Kd_490"
modProcess $1 $0
```

The following is an example of an output file. In this case, the Martha Vineyard Coastal Observatory was the point of interest and the tau_865 was one of the products.

```
$ cd /rs/lvl5/points/MVCO/
$ ls
MVCO_nLw_412.dat  MVCO_nLw_490.dat  MVCO_nLw_555.dat   MVCO_taua_443.dat  MVCO_tau
MVCO_nLw_443.dat  MVCO_nLw_510.dat  MVCO_taua_412.dat  MVCO_taua_490.dat  MVCO_tau
$ more MVCO_taua_765.dat
orbview-2.2002001.0101.165223.D.L3.seawifs.MVO.v08.1100m.hdf,730851.7060,01/01/200
orbview-2.2002002.0102.173356.D.L3.seawifs.MVO.v08.1100m.hdf,730852.7351,01/02/200
orbview-2.2002005.0105.180141.D.L3.seawifs.MVO.v08.1100m.hdf,730855.7543,01/05/200
```

```
orbview-2.2002008.0108.165126.D.L3.seawifs.MVO.v08.1100m.hdf,730858.7055,01/08/200
orbview-2.2002008.0108.183122.D.L3.seawifs.MVO.v08.1100m.hdf,730858.7732,01/08/200
orbview-2.2002015.0115.165019.D.L3.seawifs.MVO.v08.1100m.hdf,730865.7046,01/15/200
orbview-2.2002018.0118.171725.D.L3.seawifs.MVO.v08.1100m.hdf,730868.7236,01/18/200
orbview-2.2002022.0122.164904.D.L3.seawifs.MVO.v08.1100m.hdf,730872.7037,01/22/200
orbview-2.2002024.0124.181333.D.L3.seawifs.MVO.v08.1100m.hdf,730874.7615,01/24/200
orbview-2.2002025.0125.185537.D.L3.seawifs.MVO.v08.1100m.hdf,730875.7905,01/25/200
orbview-2.2002027.0127.184055.D.L3.seawifs.MVO.v08.1100m.hdf,730877.7803,01/27/200
orbview-2.2002029.0129.164732.D.L3.seawifs.MVO.v08.1100m.hdf,730879.7026,01/29/200
```

The last value is the taua_765 value. The values which are zero are due to cloud cover, atmospheric failure, or other flag. The one marked “invalid” was a file that covered the region of interest, but did not cover the point in question. See the documentation for **imgRead** for information about the other fields and additional options that may be used with `PTOptions`.

Extracting Time Series at Region(s) of Interest

During normal data processing, APS has the capability to extract time series (mean, standard deviation, etc.) data from user selected regions of interest and create simple ASCII text files that are continually updated as new data is processed. a NDBC buoy.

To begin the user must create the `blotchs` in the `$APS_ETC` directory. Here, the user would place the “blotch” files. There is a standard naming scheme `$MapName.blotchs`. The file would consist of polygons of latitude and longitude. See the **imgTSeries** and **imgBrowse** for more information.

The example below shows an example the area script.

```
#!/bin/sh
. $APS_LIB/apsScripts.sh
MapName="MissBight"
TSProdList="chlor_a Kd_490"
modProcess $1 $0
```

Note

The time series program works only on 2-D geophysical data products, so, it will remove `l2_flags`, `true_color`, and `water_mass` products from `TSProdList` automatically.

The following is an example of an output file. In this case, the Mobile Bay Lake Pontchartrain, Miss Sound Shelf, and an offshore region were the regions of interest.

```
$ cd /rs/lvl5/blotchs/
$ ls
MissBight_monthly_seawifs.dat  MissBight_seawifs.dat  MissBight_weekly_seawifs.dat
$ cat MissBight_seawifs.dat |grep chlor_a | grep MissBight | head
orbview-2.2002003.0103.181811.D.L3.seawifs.MSB.v08.1100m.hdf,1010103854,01/03/02 1
orbview-2.2002003.0103.181811.D.L3.seawifs.MSB.v08.1100m.hdf,1010103854,01/03/02 1
orbview-2.2002004.0104.171945.D.L3.seawifs.MSB.v08.1100m.hdf,1010186886,01/04/02 1
orbview-2.2002004.0104.171945.D.L3.seawifs.MSB.v08.1100m.hdf,1010186886,01/04/02 1
orbview-2.2002004.0104.190026.D.L3.seawifs.MSB.v08.1100m.hdf,1010192756,01/04/02 1
orbview-2.2002004.0104.190026.D.L3.seawifs.MSB.v08.1100m.hdf,1010192756,01/04/02 1
orbview-2.2002005.0105.180141.D.L3.seawifs.MSB.v08.1100m.hdf,1010275792,01/05/02 1
```

```
orbview-2.2002005.0105.180141.D.L3.seawifs.MSB.v08.1100m.hdf,1010275792,01/05/02 1
orbview-2.2002006.0106.170537.D.L3.seawifs.MSB.v08.1100m.hdf,1010358838,01/06/02 1
orbview-2.2002006.0106.170537.D.L3.seawifs.MSB.v08.1100m.hdf,1010358838,01/06/02 1
```

Reading APS products in SeaDAS

There are two types of data files produced by programs in the APS which can not be directly read into SeaDAS. The first is a flat-binary format similiar to the PC-SEAPAK format, except that the restriction of data type and image size has been lifted. The second is an HDF formatted file defined by APS.

Several `.pro` files are available in the `share` directory that can read these two formats. To implement these, begin by copying them into the `idl_lib` directory of your SeaDAS installation. Next, you will need to modify the the SeaDAS file `seadispx.pro` (be sure to keep a backup). The file creates the menu for the 'Load' button under SeaDISP. The procedures below *will not* work with the embedded version of SeaDAS.

To add the NRL `.pro` files to `seadispx.pro` use your favorite UNIX text editor and add two lines for each format. The first new line must go in the function `SEADISPX_EVENT` defined at the top of `seadispx.pro` in the case statement for event.value of the 'Load Menu' (approximately at line 50). Insert either or both of these lines:

```
'NRL (HDF)':            SDP_LOAD_NRL, GROUP=event.top
'NRL (SeaPAK)':         SDP_LOAD_NRL_SPK, GROUP=event.top
```

The second location in `seadispx.pro` is in the function `SEADISPX` where menu1 is created (approximately at line 180). At this location insert either or both of these lines:

```
{ CW_PDMENU_S, 0, 'NRL (HDF)' },    $
{ CW_PDMENU_S, 0, 'NRL (SeaPAK)' }, $
```

When you restart SeaDAS, you will see either or both of the above entires under the 'Load' pull-down menu for the program SeaDISP.

Setting up a PostgreSQL database

Log in as postgres after installing the database software.

See your systems' administrator for questions about installing the database software, PostgreSQL.

The first thing we must do is create a database. Generally, you should create user names for individuals who will use the database, and a "root" database user(Using the postgres system user is ok in which case you don't need to create a special user for that). The command "createuser username" will create a user with name->username. It will ask if the user can create databases, and if the user can create new users. Answer these as appropriate for the user you are creating.

Next we should create a database. Use the command "createdb dbname" to create a database with name dbname. The name should be "rs_lvl#", where # is the Level of the datafiles we will keep(lvl3,lvl4,etc). In order to set up the tables and permissions, we must log into the database. First we need to locate the text-file that will create the database we want. They are named like the dbname("setupRS_LVL3", "setupRS_LVL4", etc). Use the command "psql -Uusername dbname" to log into dbname as username. at the prompt, use "\i textfile". This will read the text file which in turn creates a Group g_aps, an aps User,

a web7333 User, and all the tables. The 'aps' user is used by the aps system to catalog and modify files by default, but if you installed the APS system under a different user account, you should create an identical username in SQL. Use the command "CREATE username IN GROUP g_aps;". Any user in the Group g_aps will have the same permissions as the aps User. The web7333 user is given only Read permissions on the tables except the webparameters table which it may modify through other programs.

```
$ createuser admin
Shall the new user be allowed to create databases? (y/n) y
Shall the new user be allowed to create more new users? (y/n) y
CREATE USER
$ createdb rs_lvl3
CREATE DATABASE
$ psql -Uadmin rs_lvl3

Welcome to psql, the PostgreSQL interactive terminal.
Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help on internal slash commands
       \g or terminate with semicolon to execute query
       \q to quit

rs_lvl3=#\i setupRS_LVL3
rs_lvl3=#\q
```

Postgres will spit out a bunch of information after the \i command. This is all normal.

One of the many features of using a database system is the fast and accurate query returns. PostgreSQL may be used in several Programming Languages. Many of these can be run as CGI scripts on the Internet, and even some(PHP) may be embedded right inside an HTML file. This provides a convenient and fast way to share product information through the Internet. Anything from finding product images and displaying them to getting detailed information about a file can be done in this manner. Examples of what can be done through the Internet specifically related to image querying is on the NRL web site at "http://www7333.nrlssc.navy.mil/imagery/imageryMenu.hmtl". All the querying and displaying of the results is done through CGI programs written in C. JavaScript is used on the Latest pages for display purposes, but the images are found and placed in a JavaScript array through a C program.

Clusters (using Ruby)

Setting up APS Clusters is fairly simple. There are two directories which must be shared: `APS_IN` and `APS_LOCK`. Though these are the only two mandatory, the entire APS directory hierarchy is generally shared across hosts. To reduce network traffic and speed up processing a separate `APS_WORK` may be preferable for each cluster host. An APS configuration file should be created for each host and named with the full hostname of the host(i.e. `aps.$(hostname).conf` where $(hostname) is replaced with the fully qualified host name like localhost.localdomain). Inside the configuration file, the environment variables `APS_IN`, `APS_LOCK`, `APS_WORK`, etc. should be set appropriately, along with unique values for the lower-cased APS environment variables such as `apsLogFile` and `apsPIDFile`. The APS configuration file should reside in the `APS_ETC` directory. That's it. APS can be started on each machine and they will all work together to process the files appearing in the `APS_IN` directory.

Note

The previous section on Locking Semantics should be read as there are changes relating to the restarting procedure of APS.

Sun Grid Engine

Sun Grid Engine is an open source batch-queuing system. SGE is typically used on a computer farm or high-performance computing (HPC) cluster and is responsible for accepting, scheduling, dispatching, and managing the remote and distributed execution of large numbers of standalone, parallel or interactive user jobs. It also manages and schedules the allocation of distributed resources such as processors, memory, disk space, and software licenses. A typical Grid Engine cluster consists of a master host, and one or more execution hosts. Moreover, multiple shadow masters can be configured as hot spares, which take over the role of the master when the original master host crashes. [1]

To take advantage of the Sun Grid Engine, the user would use the APS script **apsSGE.sh**. As with other APS frontends, the same command options **init**, **term**, **start**, **stop**, etc. are used. The script should be run on the SGE submit-host(s) for the scheduling of jobs.

Once **apsSGE.sh** is started, it enters an infinite loop of which most of its time is spent sleeping. When it is awake, it looks for data in the `in` directory. If it finds a file [2] , for example, the SeaWiFS file `S2008335131809.L1A_HROT`, then a `S2008335131809.L1A_HROT.lock` is created in the `var/lock` directory signaling that processing has begun on that file.

The file is examined to determine its sensor type (SeaWiFS, MODIS, etc.) so that the appropriate areas sub-directory will be examined for scripts to schedule. The table below shows the name of the sub-directory consulted based upon the exit status of the **filefmt** program. If the file is compressed, an uncompressed copy is placed in the `work` and used for this test.

Table 5.2. Areas directory structure for apsSGE.sh

Type	Directory	File Description
39	$APS_AREAS/seawifsl0	SeaWiFS Level-0 Data File
40	$APS_AREAS/seawifs	SeaWiFS Level-L1A Data File
101	$APS_AREAS/modis	MODIS 1KM Data File
102	$APS_AREAS/modis250	MODIS QKM Data File
123	$APS_AREAS/meris	MERIS RR Data File
125,119	$APS_AREAS/hmeris	MERIS FR/FRS Data File

Only scripts designed to work with the specific sensor should be placed in the appropriate area script. Otherwise, the Sun Grid Engine scheduler will be used inefficiently.

Once the sensor specific area script directory is known, the appropriate areas program (**swfArea**, **modArea**, etc.) is used to determine which scripts in the directory contain maps for which the file covers. To do this, **apsSGE.sh** will grep the
<variable>mapName</variable>
from each areas script, run the areas program passing the map and input file. The default map file is `$APS_DIR/etc/maps`. If the result of the area program is anything other than 'No coverage', the area script is added to the list of area scripts to process.

For each area script, a SGE script file (.sge) is prepared in the `$APS_DIR/var/sge` directory. The file nomenclature includes the input file name, submit hostname, and area script. For example: `prepare.submithost.MOD021KM.P2009135.1220.hdf.ModMayumba.sge`. After each SGE script file for each area has been created, a post-processing SGE script is created. This

[1] Wikipedia Article on Sun Grid Engine
[2] The file must be readable.

script contains only the input file name and submit hostname with a 'process_end' prefix. For eaxmple: `prepare.submithost.process_end.MOD021KM.P2009135.1220.hdf.sge`. Once all the SGE files have been created, the **apsSubmitJobs** function will list all SGE files that begin with 'prepare', rename them (changing prefix to 'run') and use the Sun Grid Engine **qsub** command to submit the job to the Sun Grid Engine scheduler.

Note

The 'process_end' SGE job will be started in “hold” (-h)

For example, if the `areas/seawifs` directory contains 5 areas, then five SGE files will be created in the `$APS_DIR/var/sge` directory and submitted to the Sun Grid Engine. These five SGE files are for processing each area on the single SeaWiFS L1A file. A sixth SGE job is issued for post-processing of the SeaWiFS L1A file, but placed in hold state. As each SGE job is completed, the script will remove itself from the `$APS_DIR/var/sge` directory.

The **apsSGE.sh** will return to the main loop where it will examine the SGE script files that remain for each file. When no files are found, then APS will resume the suspended post-processing job by using the SGE **qalter** command. This will allow the system to remove the file from processing.

For example, when all five areas for the `S2008335131809.L1A_HROT` are completed, the post-process job will resume. The post-processing job will then delete the copy of `S2008335131809.L1A_HROT` in the `$APS_DIR/work` directory and remove the lock file `S2008335131809.L1A_HROT.lock` from the `$APS_DIR/var/lock` directory. The original file (or link) in the input directory is removed.

APS will then sleep for a specified time frame (usually 60 secs) before coming awake to perform the above steps once again.

```
do forever
    check for new file in input directory
    if compressed, put uncompressed copy in work directory
    for new file
        check sensor type
        check for areas in areas/sensor
        check that file covers area
        for each area
            create .sge file in sge directory
        create post-proc .sge for file
    done
    submit all SGE jobs
    check SGE jobs to see that all jobs for a file are run
    if all lava jobs for file run
        resume the post-processing job
    sleep 60
end
```

Data Extraction from L2 files

To extract a suite of products from all L2 files that fall within a defined map, define `L2_data_extract_output_file` in each area script that you would like to extract data for. The data is written to this defined file only if data for the satellite data file being processed isn't already in the the defined `L2_data_extract_output_file` Define `L2_data_extract_lat,L2_data_extract_lon` to specify one station location to extract data from. Define `L2_data_extract_latlon_file` to specify a series of stations listed in a file which

has lat,lon locations, one per line. If neither a single lat/lon station nor a list of lat/lon's is defined, a default location (Martha's Vineyard Coastal Observatory) will be extracted from each image. There should be a different output file defined for each sensor, but different area scripts for the same sensor can point to the same output file. This is because the number of channels, and therefore the number of output columns in the output file will be different for each sensor which would make reading the output file difficult. A header file is automatically created for each data file that describes the format of the data extraction file.

Part II. Data Format - Version 3.2

The sections in Part II contain a data user's guide which describes the files created by Automated Processing System.

The Regional Data Products produced by the Navy's Automated Processing System (aps) contains atmospherically corrected geophysical products in a standard map projection for a specific region of interest derived from one of several different satellites (SeaWiFS, MODIS, AVHRR, ...). This Data Product Guide will describe version 3.2 of the aps file format.

Chapter 6. The APS Format

The APS format is based on the Scientific Data Sets interface in version 4 of the Hierarchical Data Format (HDF4). No other interface or objects are used or allowed in a valid APS file.

Within the Scientific Data Sets subset, a valid APS file is limited to an array of no more than three dimensions one of which may be UNLIMITED. All standard number types (INT8, UINT8, INT16, UINT16, INT32, UINT32, FLOAT32, FLOAT64, and CHAR8) may be used.

None of the pre-defined attributes using such API's as SDgetdatastrs(), for example, are used. The APS format supports both file and dataset attributes. The APS format has several required file and dataset attributes that must exist with each file or dataset, respectively.

There is no limit to the number of datasets other than those imposed by the HDF4 library. There is, however, some limits placed on the names of general dataset names.

The APS IO library contains routines for accessing all objects from the APS file. The use of the library is encouraged so that the underlying file structure may change. This docuement will describe the file format structure as well as the use of the library to get or obtain that particular object.

Standard APS File Attributes

The following file attributes are required in all APS files. In the tables below, the data type of the attribute precedes the name of the attribute. If the attribute is an array (limited to a single dimension), the size is noted.

Table 6.1. File Attributes

Name	Description
file	The name of the product
fileClassification	Always set to UNCLASSIFIED
fileStatus	Either EXPERIMENTAL or OPERATIONAL
fileTitle	see Table 6.2, “FileType Values”
fileVersion	3.2
createAgency	The agency which created the file
createSoftware	The version of the software which created the file
createPlatform	The hardware/software platform the file was created on
createTime	The date and time when the file was created
createUser	The name of the user that created this file

The fileType attribute describes the type of data stored in the APS file. It may contain any of the following values.

Table 6.2. FileType Values

Value	Description
NRL Level-3	NRL file containing Level-3 data

Value	Description
NRL Level-3 Mosaic	NRL file containing a mosaic of several Level-3 data files
NRL Level-4	NRL file containing Level-4 data

This information provides information about the starting and ending time of the data used to create this product. This information is relative to the input product file.

Table 6.3. Time Attributes

Name	Description
timeStart	UTC start time as an ASCII string
timeStartYear	UTC year of data start, e.g. 2003
timeStartDay	UTC day-of-year of data start (1-366)
timeStartTime	UTC milliseconds-of-day of data start (1-86400000)
timeEnd	UTC end time as an ASCII string
timeEndYear	UTC year of data end, e.g. 2003
timeEndDay	UTC day-of-year of data end (1-366)
timeEndTime	UTC milliseconds-of-day of data end (1-86400000)
timeDayNight	Flag indicating if data collected during day or night. May be one of Day, Night, Day/Night

These attributes are used to describe the specific sensor from which the Level-3/Level-4 Regional Data Product was derived. The APS software can handle several data streams including AVHRR, MODIS, and SeaWiFS.

Table 6.4. Sensor Attributes

Name	Description
sensor	AVHRR/3, SeaWiFS, MODIS, MERIS, etc.
sensorPlatform	Platform carrying sensor, like Orbview-2, NOAA-12, MODIS-AQUA
sensorAgency	Agency/Owner of Sensor
sensorType	Type of sensor: scanner, pushbroom, whiskbroom
sensorSpectrum	Description of spectrum: visible, near-IR, thermal
sensorNumberOfBands	Number of Bands
sensorBandUnits	Units of wavelengths, like nm
sensorBands	Center wavelengths
sensorBandWidths	Nominal width of bands
sensorNominalAltitudeInKM	Nominal Altitude of sensor
sensorScanWidthInKM	Distance on earth of Field of View in kilometers
sensorResolutionInKM	Distance on earth of a single pixel in kilometers
sensorPlatformType	Type of platform

Chapter 7. Level 3 Regional Data Products

A Level-3 Regional Data Product file contains atmospherically corrected geophysical products in a standard map projection for a specific region of interest derived from one of several different satellites (AVHRR, “High-Res”MODIS, MERIS, MODIS, OCM, SeaWiFS). A Level-3 Regional Data Product can be stored in one of three underlying formats. The default format is the Heiarchical Data Format (HDF) developed by the National Center for Supercomputer Applications (NCSA) at U. of Illinois Urbana-Champaign (version 4.2r2). Additionally, version 5 of the HDF format can be written (using HDF5 v1.8.2) as well as the netCDF v3 format.

Naming Convention (Newer)

This naming scheme will be used when fileNomenclature is set to 1.
The form of a Level-3 Regional Data Product file name is `satellite.yyyyddd.mmdd.hhmmss.X.L3.sensor.region.version.resolution.format`, where `satellite` is

- `orbview-2`
- `noaa-12`
- `noaa-14`
- `noaa-17`
- `noaa-18`
- `noaa-19`
- `terra`
- `aqua`
- `envisat-1`
- `oceansat-1`

`yyyyddd.mmdd.hhmmss` is the concatenated digits for the UTC year, day of year, month of year, day of month, hours, minutes and seconds of the first scan line processed to form the Level-3 file. `X` is a day/night flag indicator:

- `D`for day-time only scans
- `N` for night-time only scans
- `B` for both day and night scans
- `U` for unknown

`sensor` is

- `avhrr` for the Advanced Resolution Radiometer

- `hmeris` for MERIS (300m)
- `hmerisl2` for MERIS (300m) using ESA Level-2
- `hmodis` for “High-Resolution” MODIS
- `meris` for MERIS (1200m)
- `merisl2` for MERIS (1200m) using ESA Level-2
- `modis` for Moderate Resolution ,
- `ocm` for Ocean Color Monitor
- `seawifs`for SeaWiFS

`region` is a user-defined code to represent the region. `resolution` is a indication of the resolution of the original input data. `version` is the algorithm processing version. `format` may be one of:

- `hdf`for HDF v4 format
- `h5` for HDF v5 format
- `nc` for netCDF

Examples of Level-3 Regional Data Product file names are:

- `aqua.2003030.0130.190000.D.L3.modis.MSB.v06.1000m.hdf`
- `envisat-1.2008177.0625.183241.D.L3.hmerisl2.MNT.v03.300m.hdf`
- `noaa-18.2009090.0331.070743.N.L3.avhrr.MSB.v05.1100m.hdf`
- `orbview-2.2001148.0528.181616.D.L3.seawifs.SWR.v07.1100m.hdf`

Naming Convention (Older)

This naming scheme will be used when fileNomenclature is set to 0.
The form of a Level-3 Regional Data Product file name is `XXyyyydddhhmmss.L3_HYYY_RRR`, where `XX` is

- `S` for SeaWiFS,
- `ND` for AVHRR NOAA-12,
- `NJ` for AVHRR NOAA-14,
- `NK` for AVHRR NOAA-15,
- `NL` for AVHRR NOAA-16,
- `NM` for AVHRR NOAA-17,
- `NN` for AVHRR NOAA-18,
- `MODAM` for MODIS on TERRA,
- `MODAMQKM` for “High-Resolution” MODIS on TERRA,

- `MODPM` for MODIS on AQUA,
- `MODPM` for “High-Resolution” MODIS on AQUA,
- `MER` for MERIS,
- `OCM` for OCM

`yyyydddhhmmss` is the concatenated digits for the UTC year, day of year, hours, minutes and seconds of the first scan line processed to form the Level-3 file. `YYY` is the HRPT station code and `RRR` is a user-defined code to represent the region. Examples of Level-3 Regional Data Product file names are: `S2001219180008.L3_HNAV_GOM` and `NJ2001026211209.L3_HNAV_NYB`.

File Attributes

The file attributes are associated with all products in the HDF file. These attributes contain such information as the creation of the file, the sensor platform for which the data was derived, and geographical coverage. The attributes are divided into several groups and are discussed below.

The processing meta data includes a description of the inputs used to create this Level-3 Regional Data Product file. Not all attributes given here will be present in a given file - several are sensor specific.

The `prodList` attribute is a list of the geophysical products stored in the file. It excludes the standard data sets like CP_Pixels, etc.

Table 7.1. Input Parameters Attributes

Name	Description
inputCalibrationFile	Name of the calibration file used. SeaWiFS/MOS specific.
inputParameters	A string indicating the options used during the processing of the file
inputMasksInt	The mask defined as an integer
inputMasks	A comma seperated list of flags that were used as masks during processing.
prodList	A comma seperated list of products stored in this file.
processingVersion	Version of processing

This meta data contains information about the geographical coverage of the data file and the resulting regional geographical coverage.

Because the Level-3 Regional Data Product files are mapped images, the following information provides the navigation information of the projection used. The projection system used by APS v3.7 is based on the NASA PC-SeaPAK projection system which uses two tie-points and the USGS projection software to perform the navigation.

Table 7.2. Navigation Attributes

Name	Description
navType	Navigation type of data. Always set to 'mapped'

Name	Description
mapProjectionSystem	Map projection system used. Always set to NRL(USGS)
mapProjection	Name of the SDS included in the file that contains the map projection parameter values.
mappedUpperLeft	Latitude and longitude of upper left (1,1) point of each product.
mappedUpperRight	Latitude and longitude of upper right (1,n) point of each product.
mappedLowerLeft	Latitude and longitude of lower left (m,1) point of each product.
mappedLowerRight	Latitude and longitude of lower right (m,n) point of each product.

This meta data contains information about the geographical coverage of the data file. This data is still in sensor view.

Table 7.3. Input Geographical Coverage Attributes

Name	Description
localeUpperLeft	latitude and longitude of upper left (1,1) point of original input data.
localeUpperRight	latitude and longitude of upper right (1,n) point of original input data.
localeLowerLeft	latitude and longitude of lower left (m,1) point of original input data.
localeLowerRight	latitude and longitude of lower right (m,n) point of original input data.
localeNWCorner	latitude and longitude of NorthWestern point of original input data.
localeNECorner	latitude and longitude of NorthEastern point of original input data.
localeSWCorner	latitude and longitude of SouthWestern point of original input data.
localeSECorner	latitude and longitude of SouthEastern point of original input data.

Special Data Sets

There are a *variable* number of datasets in an APS Level-3 Regional Data Product file. However, several data sets are standard. These data sets are meta data sets containing information related to the actual geophysical data products. The first group of data sets listed below provide geographical coverage and data quality information. The last group of data sets are the actual geophysical products and vary in number.

The following data sets provide for a grid of control points that relate the image locations (pixel,line) with a geographical location (lat,lon). The grid is a 2-D regular “square” grid with each latitude and longitude positioned at the given pixel and line locations. These are single dimensional arrays indicating the pixel and line locations for each grid point. Though every Level\-3 Regional Data Product file is map projected,

this control points array is provided to give the user an alternative method of georeferencing each pixel in the data products.

Table 7.4. Geographical Location Datasets

Map Projections	
Name	**Description**
mapProjection	Map projection parameters
Control Points	
Name	**Description**
CP_Pixels[m]	Control Point pixel locations
CP_Lines[n]	Control Point line locations
CP_Latitudes[m][n]	Latitude in decimal degrees for each control point
CP_Longitudes[m][n]	Longitude in decimal degrees for each control point

The data quality data sets provide information about the data quality of each pixel in the geophysical product data sets.

Table 7.5. Data Quality Datasets

Name	Description
l2_flags	Level-2 Processing Flags

The l2_flags contain 32 bits. Each bit represents a condition or state for the pixel in question. Each bit and their associated keywords are shown in the following table.

Table 7.6. l2_flags Bit Flag Values

Bit	Name	Description
1	ATMFAIL	Atmospheric correction algorithm failure
2	LAND	Land pixel
3	SPARE3	reserved data
4	HIGLINT	Sun glint
5	HILT	Total radiance greater than the knee value
6	HISATZEN	Large sensor zenith angle
7	COASTZ	Shallow water (>30m deep)
8	NEGLW	Negative Water-Leaving Radiance
9	STRAYLIGHT	Stray Light
10	CLDICE	Cloud or ice
11	COCCOLITH	Coccolithophores
12	TURBIDW	Case 2 waters
13	HISOLZEN	Large solar zenith angle

Bit	Name	Description
14	HITAU	High aerosol concentration
15	LOWLW	Low water-leaving radiance at band 5
16	CHLFAIL	Chlorophyll algorithm failure
17	NAVWARN	Questionable navigation
18	ABSAER	Absorbing aerosol
19	STUMPF	Absorbing aerosol correction (Stumpf 2007) applied to pixel
20	MAXAERITER	NIR algorithm exceeded maximum iteration
21	MODGLINT	Moderate Sun glint
22	CHLWARN	Chlorophylla out of range
23	ATMWARN	Epsilon out of range
24	DARKPIXEL	Dark pixel
25	SEAICE	Sea/ice
26	NAVFAIL	Navigation failure
27	FILTER	Data filtering failure
28	SSTWARN	SST out of range
29	SSTFAIL	SST computation failure
30	HIPOL	High polarization
31	PRODFAIL	Product algorithm failure
32	OCEAN	Ocean pixel

Product Data Sets

The remaining datasets in the Level-3 Regional Data Product data file are general in nature and may or may not exist in the given dataset. A description of each product follows. All products below that exist in the Level-3 Regional Data Product file are listed in the file attribute `prodList`.

To describe the product, several attributes are attached to the data set. These include a long descriptive name, the units and valid range of the data.

The long descriptive name is an ASCII string. In some cases, the algorithm used is also provided in the name. Examples are “Remote Sensing Reflectance at 443 nm” and “Chlorophyll Concentration, OC4 Algorithm”.

The units are stored in an ASCII string. Examples of unit strings are “mg m^-3” and “mW cm^-2 um^-1 sr^-1”.

The valid range attribute provides a suggested range of valid data. This attribute can be used to filter out values that are outside the given range, for example, for compositing several scenes together. In general, values outside this range are considered suspect.

Additionally, the value used to represent invalid values is also defined. An invalid value represents locations where a value can not be computed. For example, when trying to compute a chlorophyll-a value

in which the remote sensing reflectance at 443 nm is negative. Or, when a particular pixel is over land and the land is being mask out (the normal procedure).

In general, most of the archival products are stored as signed 16-bit integers. In all cases for products produced by APS to date, the scaling is linear. Note, that not all products are stored as integers; some are stored as floating point values. The products listed below that indicate floating point representation are generally not archival products. In the cases of floating point products, these attributes are not available.

If the product is stored as integers, then it will also have the next two attributes, which are the slope and intercept for converting the integers into floating points. The integer value should be read from the SDS and multiplied by the slope and then have the intercept added.

These attributes are for the “standard” units of the product. If the product has the attribute `additionalUnits` set, then it will have additional attributes similar to these that allow for the conversion of the data in those units. The attributes will be named according to the unit name. For example, if `attionalUnits` contains “Fahrenheit” the product's slope and intercept will be contained in the attributes `FahrenheitSlope` and `FahenheitIntercept`. See the example below

To provide for making quick-look browse images, suggestions about the scaling and data ranges for making images of the product data are stored in the following two attributes. The APS program imgBrowse will use these attributes as default scaling and data ranges for automatic creation of quick-look browse images.

For some products, like diffuse attenuation, it is generally accepted that a logarithmic scaling for images brings out the best detail. Therefore, for this product a logarithmic function will be used. Other products like sea surface temperature are generally best displayed using linear scaling.

The display range of the data may or may not be the same as the `validRange` attribute above since in some cases (e.g. rrs_412), the data has been known to fall outside a geophysical valid range, but we wish to display the un-physical data since some physical oceanographical structure or feature may still be present in the data.

Table 7.7. Product Data Set Attributes

Name	Description
createSoftware	This string contains the version of the software which created the product.
createTime	This string contains the date and time when the product was created
createPlatform	This string contains a triple describing the cpu-machine-os which created the scientific data set
productName	This is a description of the product.
productAlgorithm	This is a notation about the algorithm, usually a paper reference.
productUnits	This is a description of the units of the product.
productVersion	This is a version number of the product used to indicate changes in the algorithm.
productType	This is a type of product. For example, 'chl_oc4v4' and 'chl_oc3m' would both set this to 'chl'.
additionalUnits	This is a space delimetered string of additional units available for this product. For example, an sst product may set this string to “Kelvin Fahrenheit”

Name	Description
productStatus	This new SDS attribute will give an indication of the status this product.
validRange	This is a suggested range of valid data.
invalid	This is the geophysical value which will represent invalid data for the given product.
productScaling	The type of scaling of the product. Currently, always Linear
scalingSlope	The slope for product scaling.
scalingIntercept	The intercept for product scaling.
browseFunc	This is a suggested function to apply to convert the data in the SDS into an image. A value of 1 indicates linear scaling; a value of 2 indicates log10 scaling.
browseRanges	This is a suggested display range when converting the data in the SDS into an image. This may or may not be the same as validRange because in some cases (e.g. rrs_412), the data has been known to fall outside the range, but we wish to display the invalid data. This attribute is used by the APS program imgBrowse when creating quick-look browse images of different products.

Chapter 8. Level-4 Regional Data Products

A Level-4 Regional Data Product file contains atmospherically corrected geophysical products in a standard map projection for a specific region of interest derived from one of several different satellites (AVHRR, “High-Res”MODIS, MERIS, MODIS, OCM, SeaWiFS). A Level-4 Regional Data Product may be stored in one of several formats. The default is stored using the Heiarchical Data Format (HDF) developed by the National Center for Supercomputer Applications (NCSA) at U. of Illinois Urbana-Champaign (version 4.2r2). Additionally, version 5 of the HDF format can be written (using HDF5 v1.8.2) as well as the netCDF v3 format.

File Attributes

The file attributes are associated with all products in the HDF file. These attributes contain such information as the creation of the file, the sensor platform for which the data was derived, and geographical coverage. The attributes are divided into several groups and are discussed in the the section called “File Attributes”

The Level-4 Regional Composite Data Product Files also contain an additional set of attributes.

Table 8.1. Input Parameters Attributes

Name	Description
compType	Set to `Daily Composite', `Weekly Composite', `Monthly Composite', `Yearly Composite', `Rolling Composite`, `Latest Pixel Composite', `Seasonal Composite'
compStartTimeFrame	The intended start time of the composite (not set for `Rolling Composite' or `LatestPixel Composite'
compEndTimeFrame	The intended end time of the composite (not set for `Rolling Composite' or `LatestPixel Composite'
compTimeFrame	The time frame of composite in days (like 8, 30, 31, 365)
compMaxPixels	The maximum number of pixels used in the latest pixel composite. Required only for `Latest Pixel Composite'
inputFiles	A list of files used to produce the average product

Special Data Sets

There are a *variable* number of datasets in an APS Level-4 Regional Data Product file most of which contain a oceanographic parameter. However, several data sets are special. These data sets are meta data sets containing information related to the actual geophysical data products. These data sets are described in the section called “Special Data Sets”.

Product Data Sets

The product data sets of a Level-4 Regional Data Product file are similar to Level-3 Regional Data Products described in the section called “Product Data Sets”. The difference in the Level-4 files is that the product is an average of the input files.

Additionally, other data sets which describe the minimum, maximum, standard deviation, and number of pixels may also be present. These datasets use the same name as those described in above, except that an `_min`, `_max`,`_stddev`, and `_num` are appended to each product name. For example, the file may contain the data set `sst_max`, which is the maximum composite of the `sst` product.

Part III. Command Line Reference

The sections in Part II contain the reference pages for each program available in the Automated Processing System.

Name

aps — aps driver script

Synopsis

`aps.sh` [*`init`* | *`start`* | *`stop`* | *`kill`* | *`term`* | *`config`* | *`setup`*]

`aps.sh` *`repro`* [*`list`* | *`dir`*] [*`area1 area2`*...]

`aps.rb` [*`init`* | *`start`* | *`stop`* | *`kill`* | *`term`* | *`config`* | *`setup`*]

`aps.rb` *`repro`* [*`list`* | *`dir`*] [*`area1 area2`*...]

Description

The **aps.sh** or **aps.rb** script is the driver for the Automated Processing System. The **aps.sh** script is a Bourne Shell script for single processor systems (sequential processing). The **aps.rb** is a Ruby script for a cluster of single or multiple processor systems (parallel processing). Either is also the administrative interface to the Automated Processing System - the script the user controls the system. Additionally, the script is used for reprocessing data. It is recommended that the **aps.rb** be used.

init

The *init* command is used to start up the aps driver. The prompt will return immediately, but the script has restarted itself in the background and will enter an infinite loop.

When initialized, the aps driver will create three files: `aps.$(hostname).pid` , `aps.$(hostname).process`, and `aps.$(hostname).log`. The first file contains the processing ID of the driver (**aps.sh** or **aps.rb**). It is used to terminate and/or kill a running APS driver. The second file is used to tell the aps driver whether to process any data. It is checked for existance once a second (by default). If non-existant, no data will be processed. It provides the user a method to temporarily halt data processing without having to terminate the aps driver (see the *start* and *stop* commands).

The last file is a logging file. The aps driver will write messages about the files it has processed to this file. For the Ruby based driver, this is a template for the log file naming. By default, the Ruby driver uses a rolling logger which automatically appends a six digit sequential number preceded by a dash before the `.log` extension.

term

The *term* command is used to terminate the aps driver whose PID is in the file `aps.$(hostname).pid`. This may take a few moments or up to a few hours to complete as the aps driver will wait for any children processes to complete first.

kill

The *kill* command is used to immediately kill the aps driver whose PID is in the file `aps.$(hostname).pid`. Try *term* first and if it does terminate fast enough for you, then use this. The children of the aps driver *may* still be running, however. This is usually called by the `rc.aps` script, prior to a system shutdown.

start

The *start* command creates the `aps.$(hostname).process` file which is used to signal the aps driver to proceed with data processing. Usually this is only called if *stop* had been previously run.

stop

The *stop* command deletes the `aps.$(hostname).process` file which is used to signal the aps driver to stop processing data. When the file is non-existant the aps driver will continue to poll the `in` directory, but will never process the data.

One use of the two commands *start* and *stop* is within *cron* to prevent the aps from running (and taking up computer resources) during the work day. This was its original intent, though the aps is now usually run on a dedicated system.

config

The *config* command is used print out the configuration file to be used by the script if any.

There exists an search pattern for the configuration file. First a host specific configuration file is looked for and then an static configuration fie. For example, the **aps.sh** will first look for, `etc/aps.$(hostname).conf` and if found use that as the configuration file. Otherwise, the file `etc/aps.conf` is used. If this file is not found, the **aps.sh** will exit with an error message.

The **aps.rb** uses an XML configuration file. The search pattern for this file has several different levels. To begin, if the environment variable `APS_CONFIG` is defined, that is taken to be the XML configuration file. Otherwise, the directory search pattern: `APS_ETC`, `APS_DIR/RUBY_PLATFORM/etc`, `APS_DIR/etc`, *`directory_of_executable`*`/etc.`

setup

The *setup* command is used open a GUI interface for the creation of the configuration file to be used by the *init* command.

repro

The *repro* option is the preferred way to reprocess data. The user must either provide the program with a list of data files to reprocess stored in a UNIX text file or (2) provide a directory path for which the files are contained.

If a text file list is used, it is best to use full pathnames in it. Each file to be reprocessed must reside on a single line in the text file. A line which starts with '#' will be skipped. Given the list file, the aps driver will execute each script in the `areas` directory sequentially on each file. The files listed must include the full pathname. They may be compressed using **gzip**, **bzip2**, or **compress**. If any files are compressed, then the path to the local copy of the gzip program must be defined by the keyword *CmpGzip* found in the `aps.conf` file. (Normally, it is defined during the installation of the aps software.) The *repro* option will automatically place an (uncompressed) copy of each input file in the `APS_WORK` directory before processing. The copy is removed automatically. *The original input files are not touched*.

The user can also specify which of the *areas* scripts are used in the reprocessing. For those area names listed on the command line that have relative paths, the `APS_AREAS` directory will automatically be searched for the given area script. If the user does not specify which areas to process, then *all* areas in the `APS_AREAS` directory are used.

If the environmental variable `REPROC_MAIL` is set, then an e-mail message will be issued to the user defined in `REPROC_MAIL` for each file when it is completed and a final message when all files have been reprocessed.

For example, suppose we wish to reprocess the entire 1999 year of SeaWiFS data for the MissBight and GulfOfMexico regions. Then we might do the following:

```
$ dir=/rs/lvl1/seawifs/hrpt/HNAV/1999
$ find $dir -name "S*gz" -print | sort > ~/repro.list
$ export REPROC_MAIL=username@local.domain
$ ~/aps_v3.8.2.3-72-g7b866d/bin/aps repro ~/repro.list SwfMissBight SwfGulfOfMexic
Processing these regions:
    /home/aps/aps_v3.8.2.3-72-g7b866d/areas/SwfMissBight
    /home/aps/aps_v3.8.2.3-72-g7b866d/areas/SwfGulfOfMexico
Working in S1999091174814.L1A_HNAV ...
SwfMissBight : checking if file covers MissBight ... yes.
SwfMissBight : converting L1 to L2 ... done.
SwfMissBight : converting L2 to L3 ... done.

output continues ...
```

Environmental Variables

The environmental variables in this section are required to define the directory structure used by the aps system. Many of the environmental variables have defaults if the normal directory structure is followed. They should only be re-defined if your configuration requires different values. They are defined in the configuration file (`aps.conf` for **aps.sh** or `aps.conf.xml` for **aps.rb**) located in the `etc` directory. Modify the configuration file if you need to set these to some other values.

`APS_DIR` The top level directory for the aps system.

`APS_ETC` The location of configuration files for the aps system. Defaults to `$APS_DIR/etc`.

`APS_BIN` The location of executables for the aps system. Defaults to `$APS_DIR/bin` or `$APS_DIR/$target_cpu-$target_os/bin`.

`APS_LIB` The location of libraries for the aps system. Defaults to `$APS_DIR/lib` or `$APS_DIR/$target_cpu-$target_os/lib`.

`APS_AREAS` The directory containing executable scripts for processing data received by the aps system. These scripts will be passed a single argument being the name of the input file with no pathname. Defaults to `$APS_DIR/areas`.

`APS_IN` The directory which the aps driver will continuously poll for new data to process. This directory will probably have write permissions for to allow other users to write data to this directory. Defaults to `$APS_DIR/in`.

`APS_WORK` The directory where all the data processing will be performed. Defaults to `$APS_DIR/work`.

`APS_OUT` The directory where all the final products will be moved. Defaults to `$APS_DIR/out`.

`APS_ERROR` The directory where all the files will be moved, when there is an error during the processing. Defaults to `$APS_DIR/err`.

`APS_DATA` The directory where all the data files exist. It is intended that this directory is static. Site-specific data (esp. related to configuration) should be located in `APS_ETC`. Dynamic data will be located in `APS_VAR_DATA`. Defaults to `$APS_DIR/data`.

`APS_VAR` The directory is for any dynamic files. Defaults to `$APS_DIR/var`.

`APS_VAR_DATA` The directory is for any dynamic data files. Defaults to `$APS_DIR/var/data`.

`APS_LOG` The directory used for logging. Defaults to `$APS_DIR/var/log`.

`APS_LOCK` The directory where all locks are created. Defaults to `$APS_DIR/var/lock`.

Configuration Variables

The configuration variables are obtained from the aps configuration file. See above for information about the location of the configuration file.

apsPollingTime This variable controls the number of seconds the aps driver will sleep before examining the `APS_IN` directory. The default is sixty seconds.

apsLogFile Name of the file to receiving all logging information. The default is `$APS_LOG/aps.$(hostname).log`.

apsPIDFile Name of file to receive the PID of the running aps. This file is used to terminate and/or kill the aps system. It is also used as a lock; file. If the user attempts to initialize the aps, the attempt will fail if this file exists. This variable defaults to `$APS_DIR/var/lock/aps.$(hostname).pid`.

apsProcessFile Name of the file used as an on/off switch. This file allows the user to temporarily stop/start the aps processing. It will only effect the *next* file. That is, if the aps is currently processing a file, it will continue to do so. However, any remaining files will not be processed. The existance of this file will turn the aps on while its non-existance will turn the aps off. This file defaults to `$APS_DIR/var/lock/aps.$(hostname).process`.

apsPreProcess Name of a script to run prior to running the areas scripts for each file found in the `in` directory. This script will only be executed when a file is found and the script has execution permissions. The version included with aps is used to convert a SeaWiFS Level-0 file to Level-1A. To stop apsPreProcess from running, remove its execution permissions.

apsPostProcess Name of a script to be run after the areas scripts have been run for a given file. This script will be run giving it the name of the file. It will only run when a file is to be processed and the script has execution permissions. At NRL the script is used to move the input file to the archive system. To stop apsPostProcess from running, remove its execution permissions.

Files

The following files are used by **aps.sh**

1. `$APS_ETC/aps.conf`
2. `$APS_DIR/var/lock/aps.$(hostname).pid`
3. `$APS_DIR/var/lock/aps.$(hostname).process`
4. `$APS_DIR/var/log/aps.$(hostname).log`

The following files are used by **aps.rb**

1. `$APS_ETC/aps.xml`

2. `$APS_DIR/var/lock/aps.$(hostname).pid`
3. `$APS_DIR/var/lock/aps.$(hostname).process`
4. `$APS_DIR/var/log/aps.$(hostname)-000xx.log`

XML Configuration

The Ruby driver may be configured using an XML configuration file. The Ruby variables reside in a <ruby> element. From here, the user can set the Ruby variables to non-default values for non-standard installations.

Name

daylight — determine if sun is up for given time and place

Synopsis

`daylight` *`year day hour min lat lon`*

Description

The program **daylight** is intended to be used by shell scripts when two courses of action must be made: one for night time and the other for daytime. The program will determine the elevation of the sun based on a location and time of day. If the elevation is greater than 20 degrees, the program exits with a status of 1 (day). Otherwise, the exit status is 0 (night).

The time must be given as a four digit year, day of the year, hour and minute of the day in UTC. The location is given by the latitude and longitude of the point of interest. These must given in decimal degrees ranging from (-90.0 to 90.0) and (-180.0 to 180.0) respectively.

Options

-e #	Used to change the default value (20.0) for the elevation.
-v	Use verbose mode.
--help	Print out a small help page.
--version	Print out version of software and quit.

Name

filefmt — determines format of file(s)

Synopsis

`filefmt` *`file`*...

Description

The program **filefmt** is intended to be used by shell scripts when it must determine the type of the file in question. The formats supported are most graphics file formats and several *remote sensing* file formats. The program reads information from the file to determine its type, and therefore, must have read permissions on the input file(s). If the format can be determined, the program prints to stdout: the filename, a tab, and the type as an ASCII string. See the section called “Formats” below for a list of known types.

For the last (or only) file on the command line, a numeric code will also be set as the exit status of the program. This capability can be used by shell scripts to determine different courses of action depending on the file type.

Options

--help Print out a small help page.

--version Print out version of software and quit.

Formats

The table below provides the numeric, ASCII string, and description of the formats known by this program.

Exit Code	String	Description
0	UNKNOWN	Unable to determine format
1	PostScript	PostScript file
2	GIF	Graphics Interchange Format
3	JPEG	Joint Photographic Experts Group
4	SRF	Sun Raster file
5	SGI	Silicon Graphic's .rgb file
6	BMP	BMP file
7	TIFF	Tagged Image File
8	PBM	Portable BitMap File
9	PGM	Portable Gray File
10	PPM	Portable PixMap File
11	PBM_RAW	Portable BitMap File (Raw)
12	PGM_RAW	Portable Gray File (Raw)
13	PPM_RAW	Portable PixMap File (Raw)
14	XBM	X BitMap File
15	PNG	Portable Network Graphics
16	HDF	Heirarchical Data Format

Exit Code	String	Description
17	HDF5	Heirarchical Data Format v5
18	NETCDF	Network Common Data Format
32	NASA/GSFC CZCS Level-1	CZCS CRTT Level-1 Format
33	NESDIS AVHRR 1b (LAC)	NOAA-defined format
34	NESDIS AVHRR 1b (GAC)	NOAA-defined format
36	NSIPS	Navy Satellite Image Processing System
37	Terascan HRPT Telemetry	Level-0 HRPT (AVHRR)
38	Terascan SWHRPT Telemetry	Level-0 swapped HRPT (SeaWiFS)
39	SeaWiFS Level-0	NASA-defined Format
40	SeaWiFS Level-1A Data	NASA-defined Format
41	SeaWiFS Level-2 Data	NASA-defined Format
42	SeaWiFS Level-2 Q/C Data	NASA-defined Format
43	SeaWiFS Level-3 Binned Data	NASA-defined Format
44	NESDIS AVHRR 1b (HRPT)	NOAA-defined format
45	NESDIS AVHRR 1b (FRAC)	NOAA-defined format
46	SeaWiFS Level-1A Browse Data	NASA-defined Format
47	SeaWiFS Level-2 Browse Data	NASA-defined Format
48	SeaWiFS Level-3 Browse Data	NASA-defined Format
49	SeaWiFS Level-3 Standard Mapped Image	NASA-defined Format
50	Sensor Calibration Data	NASA-defined Format
70	MOS Level-1B Data	NASA-defined Format
71	POLDER Level-1B Data	NASA-defined Format
72	NCEP Data	NASA-defined Format
73	EPTOMS Data	NASA-defined Format
74	TOVS Data	NASA-defined Format
75	OCTS Level-1B Data	NASA-defined Format
76	OCTS Level-1B Data	NADAQ-defined Format
77	OCM Level-1B Data	NRL-defined Format
78	NRL Level-2 Data	NRL-defined Format
79	NRL Level-3 Data	NRL-defined Format
80	NRL Level-3 Mosaic Data	NRL-defined Format
81	NRL Level-4 Data	NRL-defined Format
82	NRL Level-5 Data	NRL-defined Format
87	OCM Level-1B Data	NRL-defined Format
100	MODIS Level-1B Data (1KM)	NASA-define Format
101	MODIS Level-1B Data (500 meter)	NASA-defined Format

Exit Code	String	Description
102	MODIS Level-1B Data (250 meter)	NASA-defined Format
103	MODIS Geolocation Data	NASA-defined Format
104	MODIS Level-2 Cloud Mask Data	NASA-defined Format
105	MODIS Level-1B Data (Subsetted)	NASA-define Format
106	MODIS Level-1A Data	NASA-define Format
120	MODIS Level-1B Data (1KM)	Wisconsin-defined Format
121	AVHRR Level-1B Data	NRL-defined HDF/TeraScan format
122	MERIS Level-2 Data (MER_RR__2P)	ESA-defined format
123	MERIS Level-1 Data (MER_RR__1P)	ESA-defined format
124	MERIS Level-2 Data (MER_FRS_2P)	ESA-defined format
125	MERIS Level-1 Data (MER_FR__1P)	ESA-defined format
126	MERIS Level-2 Data (MER_FR__2P)	ESA-defined format

Examples

In this example, notice that the second command echo's the return value (40) of the file since the file is a SeaWiFS Level-1A File.

```
$ filefmt S1999350182100.L1A_HNAV
S1999350182100.L1A_HNAV SeaWiFS Level-1A Data
$ echo $?
40
```

Name

gregor — converts from day-of-year to month/day and vice-versa

Synopsis

`gregor` *`year yday`*

`gregor` *`year month mday`*

Description

The program **gregor** will convert a day of the year into a month and day or vice versa. The year must be a four-digit year.

Options

-d n	Add *n* number of days to input date
-r	Compute the "Reynolds day" for given date. The "Reynolds day" is the Wednesday of each week and corresponds to file naming scheme used by Reynolds for his optimum interpolation sst values.
--help	Print out a small help page.
--version	Print out version of software and quit.

Name

hdf — general manipulation functions on an HDF file

Synopsis

`hdf` *`file.hdf`* [*`boundary`* | *`cat`* | *`copy`* | *`export`* | *`fattr`* | *`history`* | *`import`* | *`list`* | *`note`* | *`sattr`* | *`track`* | *`validate`*] [*`options`*] [*`parameters`*]

Description

The program **hdf** is used to manipulate HDF files (currently it only supports the SDS interface). With this program you can copy an SDS (with or without attributes) from one HDF file to another, dump an SDS to the screen, list all SDSs and file attributes in a format similar to NetCDF's Common Data Language (CDL).

The `file.hdf` parameter is the input file to perform the given action on. There will be a different series of parameters for each action taken. The actions that can be performed on an HDF file are: cat, list, copy.

Actions

boundary

The *boundary* command is used to dump data from the boundary array. A boundary is a closed polygon that outlines the image.

cat

The *cat* command is used to dump data from the given data set to stdout. The data will be printed according to its type (char's as string, int's as integers, and float's as floating point numbers). There will be 16 columns across the page for 8- and 16-bit signed/unsigned integers; and 8 columns across the page for 32-bit signed/unsigned interger or 32- and 64-bit floating point numbers. Characters will be printed to 72 columns.

By default, the attributes of the data set will be printed. To prevent this, the `-q` can be used. It will quiet down the output.

copy

The *copy* command is used to copy an SDS from one file to another.

To copy all file attributes from one HDF file to another file, use the comand:

```
hdf from.hdf copy -attr to.hdf
```

To copy only file attributes using a regular expression (for example, only the time attributes) from one HDF file to another file, use the comand:

```
hdf from.hdf copy -attr -re 'time*' to.hdf
```

To copy all attributes from an SDS in one file to an SDS in another file, use the command:

```
hdf from.hdf copy -attr to.hdf from.sds to.sds
```

To copy only specified attributes from one SDS in one file to an SDS in another file, use the command:

```
hdf from.hdf copy -attr to.hdf from.sds to.sds attr1 attr2 ...
```

To copy the attributes from an SDS in one file to an SDS in another file renaming the attributes in the process, use the command:

```
hdf from.hdf copy -attr -rename to.hdf from.sds to.sds attr1 newattr1 attr2 newatt
```

To copy an SDS with its associated attributes to another file, use the command:

```
hdf from.hdf copy to.hdf from.sds to.sds sds1 sds2 ...
```

To copy the SDS without the associated attributes add the -noattrs option:

```
hdf from.hdf copy -noattrs to.hdf from.sds to.sds sds1 sds2 ...
```

To rename the SDS's during the copy:

```
hdf from.hdf copy [-noattrs] to.hdf from.sds to.sds sds1 new1 sds2 new2 ...
```

export

The *export* command is used to export an object that was imported into the HDF file. Usually the imported file is an image or Encapsulated Post Script file.

fattr

The *fattr* command is used to add, change, or view a file attribute. The command has a single option -nt used to define the number type of the attribute. The default number type is “APS_NT_CHAR8” normally used to append strings attributes.

To add an attribute called `browseList` with a value of “chl_oc2,chl_oc4”, the user would run the command:

```
hdf in.hdf fattr browseList chl_oc2,chl_oc4
```

To view an attribute called “createUser” the user would run the command:

```
hdf in.hdf fattr createUser
```

history

The *history* command is used to get an ASCII dump of the contents of the history of the aps file.

import

The *import* command is used to import a file into an HDF file. This allows the user to embed an image or Encapsulated Post Script file (for example a histogram plot) into the HDF file. The file **export** is then used to extract.

list

The *list* command is used to get an ASCII dump of the contents of the HDF file in a format similiar to the netCDF Common Data Language format. The user can also request the output to be in HTML format.

```
hdf from.hdf list [ type ]
```

The parameter *type* can be either `-ascii` or `-html`. If type is not supplied or understood by **hdf**, then it defaults to `ascii`

sattr

The *sattr* command is used to add, change or view an SDS attribute. The command has a single option -nt used to define the number type of the attribute. The default number type is `APS_NT_CHAR8` normally used to append strings attributes.

To add an attribute called `comment` with a value of “very nice image”, the user would run the command:

```
hdf in.hdf sattr chl_oc4 comment very nice image
```

To view an attribute called `comment`, the user would run the command:

```
hdf in.hdf sattr chl_oc4 comment
```

validate

The *validate* command is used to validate that the input file is an aps formatted file.

Options

--help Print out a small help page.

--version Print out version of software and quit.

Name

maps — interactive program to manipulate maps

Synopsis

`maps`

`maps` [*options*] *MapFile* *MapName*

Description

The program **maps** is an interactive program used to manipulate image maps. It is *not* intended to be very robust. It was originally designed to test the Aps_MapsXXX set of routines. But is also the only way to currently create and write an image map to a file for use in the areas scripts.

When the program is executed, the user is provided with a prompt to receive commands. The quick summary of the commands can be displayed using the "help" command. Below is a quick summary of the commands.

exit quit

Quits the program.

help

Prints a simple help menu. This list all the valid commands and one line on their purpose.

load

Adds any maps found in the user-specified file to the internal list.

save

Saves the internal list of maps to the user-specified file.

create

Creates a new map. This option will prompt the user for various parameters of the image map. Map projection 5 (Mercator) is the only one that has been extensively tested.

delete

Deletes the user-specified map from the internal list.

create

Copy a map to a new name.

create

Make a zoom copy of a map. The user selects the map to be copied, the destination name of the newly created map and the zoom factor.

dist_xy

Compute the distance in meters between two points on a given map. The map projection must be set prior to using this option.

dump

Dump the current navigation structure. A debugging option.

list

Lists all the maps in the internal list.

current

Lists the currently set image map.

setmap

Sets the user selected map to the current image map.

show

Prints out the parameters for a user-specified image map.

toll x y

Used to convert the user specified image coordinates (x,y) to geographic coordinates (lon,lat).

toxy lon lat

Used to convert the user specified geographic coordinates (lon,lat) to image coordinates (x,y).

toij lon lat

Used to convert the user specified geographic coordinates (lon,lat) to map coordinates (i,j).

onmap lon lat

Used to determine if the user specified geographic coordinates (lon,lat) on the image map.

grid

Using the currently set image map, a world grid (lon,lat) is converted to image coordinates. The results are printed as a simple table.

dist_xy x1 y1 x2 y2

Computes the distance in meters between the two given image coordinates (x1,y1) and (x2,y2).

This program will be replaced by a more robust and GUI-oriented version some time in the future.

version

Prints out the version of this program.

Options

-b	Print out boundary of specified *image map*.
-c	Print out the center longitude/latitude of a specified *image map*.
-C *n*	Print out the cornet longitude/latitude of a specified *image map*. Where *n* is 0 for upper left corner, 1 for upper right corner, 2 for lower right corner, 3 for lower left corner.
-h	Print out image height of specified *image map*.
-l	Print list of all maps in mapFile.
-w	Print out image width of specified *image map*.
--help	Print out a small help page.
--version	Print out version of software and quit.

Map Projections

The following is a list of map projections supported by aps.

Number	Name
0	Geographic
1	Universal Transverse Mercator (UTM)
2	State Plane Coordinates
3	Albers Conical Equal Area
4	Lambert Conformal Conic
5	Mercator
6	Polar Stereographic
7	Polyconic
8	Equidistant Conic
9	Transverse Mercator
10	Stereographic
11	Lambert Azimuthal Equal Area
12	Azimuthal Equidistant
13	Gnomonic
14	Orthographic
15	General Vertical Near-Side Perspective
16	Sinusiodal
17	Equirectangular
18	Miller Cylindrical
19	Van der Grinten
20	(Hotine) Oblique Mercator
21	Robinson
22	Space Oblique Mercator (SOM)
23	Alaska Conformal
24	Interrupted Goode Homolosine
25	Mollweide
26	Interrupted Mollweide
27	Hammer
28	Wagner IV

Spheroids

The following is a list of spheriods supported by aps.

Table 15. Spheroids

Number	Name
0	Clarke 1866
1	Clarke 1880
2	Bessel
3	International 1967
4	International 1909
5	WGS 72
6	Everest
7	WGS 66
8	GRS 1980
9	Airy
10	Modified Everest
11	Modified Airy
12	WGS 84
13	Southeast Asia
14	Australian National
15	Krassovsky
16	Hough
17	Mercury 1960
18	Modified Mercury 1968
19	Sphere of Radius 6370997 meters
20	Bessel 1841(Namibia)
21	Everest (Sabah Sarawak
22	Everest (India 1956)
23	Everest (Malaysia 1969)
24	Everest (Malay Singapr 1948
25	Everest (Pakistan)
26	Hayford
27	Helmert 1906

Acknowledgements

The map projection software used by the aps came from the United States Geological Survey and is known as the General Cartographical Transformation Package. The aps uses version 2.0 of the C library.

Examples

For scripting purposes, the user might want to know the image dimensions of a an *image map*. To do that, we might use the following:

```
$ maps -w ~/aps_v3.8/data/maps.hdf GulfOfMexico
2430
$ maps -h ~/aps_v3.8/data/maps.hdf GulfOfMexico
1810
```

For the center and four corner points we have:

```
$ maps -c ~/aps_v3.8/data/maps.hdf GulfOfMexico
-89.007407 25.061878
$ maps -C0 ~/aps_v3.8/data/maps.hdf GulfOfMexico
-98.000000 31.000000
$ maps -C1 ~/aps_v3.8/data/maps.hdf GulfOfMexico
-80.007407 31.000000
$ maps -C2 ~/aps_v3.8/data/maps.hdf GulfOfMexico
-80.007407 18.811032
$ maps -C3 ~/aps_v3.8/data/maps.hdf GulfOfMexico
-98.000000 18.811032
$ maps -C4 ~/aps_v3.8/data/maps.hdf GulfOfMexico
invalid corner
```

An example of creating an equirectangular projection such that each pixel is exactly 0.01 degrees between each other. In other words, so that there are 100 pixels for each degree of latitude or longitude.

```
maps> create eq
Full Name? [ eq ]
Code (3-letter)? [ eq ]
Map Projection (? for list)? [ Mercator ] 17
Size of Map (w h)? 101 101
Pt1 of Map (lon lat)? -90 30
Pt1 of Map (x y)? [ 1 1 ] 51 51
Pt2 of Map (lon lat)? -91 31
Delta (pixels)? [ 100 ]
Aspect ratio? [ 1.0 ]
Datum (? for list)? [ WGS 84 ]
Radius of Reference Sphere? [ 6378137.000000 ]
Longitude of Central Meridian (lon)? 0
Latitude of True Scale (lat)? 0
False Easting? [ 0.0 ]
False Northing? [ 0.0 ]
maps> setmap eq
maps> show eq
```

```
    Name         eq
    Full Name    eq
    Code         eq
    Projection   17 (Equirectangular)
    Zone         62
    Datum        12 (WGS 84)
    Parameters:
     0:  6378137.000000 (Radius of Reference Sphere)
     1:  0.000000
     2:  0.000000
     3:  0.000000
     4:  0.000000 (Longitude of Central Meridian)
     5:  0.000000 (Latitude of True Scale)
     6:  0.000000 (False Eastings)
     7:  0.000000
     8:  0.000000
     9:  0.000000
    10:  0.000000
    11:  0.000000
    12:  0.000000
    13:  0.000000
    14:  0.000000
    Width        101
    Height       101
    Longitude_1  -90.0000
    Latitude_1    30.0000
    Pixel_1      51
    Line_1       51
    Longitude_2  -91.0000
    Latitude_2    31.0000
    Delta        100
    Aspect       1

Upper Left Corner       -90.5 30.5
Upper Middle Corner     -90 30.5
Upper Right Corner      -89.5 30.5
Middle Left             -90.5 30
Center                  -90 30
Middle Right            -89.5 30
Lower Left Corner       -90.5 29.5
Lower Middle Corner     -90 29.5
Lower Right Corner      -89.5 29.5

Map Distances
(   1,   1)     ( 101,   1)     95748.323212    957.483
(   1,  51)     ( 101,  51)     96236.863890    962.369
(   1, 101)     ( 101, 101)     96718.075759    967.181
(   1,   1)     (   1, 101)     111125.110164   1111.25
(  51,   1)     (  51, 101)     111125.110164   1111.25
( 101,   1)     ( 101, 101)     111125.110164   1111.25
maps> toll 1 1
-90.500000 30.500000
maps> toll 1 2
-90.500000 30.490000
```

```
maps> toll 1 3
-90.500000 30.480000
maps> toll 2 1
-90.490000 30.500000
maps> toll 3 1
-90.480000 30.500000
```

An example of making a “high-resolution” MODIS map of a previously created one kilometer map for the MissBight region.

```
maps> load $APS_ETC/maps.hdf
loaded 20 maps
maps> zoom
Name? MissBight
Name of new map? MissBight250
Zoom factor? 4
maps> show MissBight250
    Name         MissBight250
    Full Name    Mississippi Bight
    Code         MSB
    Projection    5 (Mercator)
    Zone         62
    Datum        12 (WGS 84)
    Parameters:
     0:  6378137.000000 (Semi-Major Axis)
     1:  6356752.314245 (Semi-Minor Axis)
     2:  0.000000
     3:  0.000000
     4:  -87030000.000000 (Longitude of Central Meridian)
     5:  29030000.000000 (Latitude of True Scale)
     6:  0.000000 (False Eastings)
     7:  0.000000 (False Northings)
     8:  0.000000
     9:  0.000000
    10:  0.000000
    11:  0.000000
    12:  0.000000
    13:  0.000000
    14:  0.000000
    Width        2400
    Height       1200
    Longitude_1  -90.5000
    Latitude_1    31.0000
    Pixel_1      1
    Line_1       1
    Longitude_2  -84.5000
    Latitude_2    28.0000
    Delta        2396
    Aspect       1
maps>quit
```

An example generating a series of boundary files from all the maps in the map file.

```
$ cat ./gmtMaps.sh
#! /bin/bash -x

APS_DIR=$HOME/aps_v3.8.2/
APS_BIN=$APS_DIR/bin
mapFile=$1

list=$($APS_BIN/maps -l $mapFile)
for m in $list
do
    $APS_BIN/maps -b $mapFile $m > $m.pts
done

$ ./gmtMaps.sh $HOME/aps_v3.8.2/etc/maps.hdf
$ ls *.pts
Adriatic.pts  MissBight.pts Texas.pts
Gomex.pts     SouthAus.pts
```

Using GMT all the boundary files can be put over an world map to show all the regions that are in maps.hdf file.

```
$ ./gmtBoundary.sh overall.eps MissBight.pts Gomex.pts Adriatic.pts SouthAus.pts T
$ convert -density 144 overall.eps overall.png
$ file overall.png
overall.png: PNG image data, 1726 x 1016, 8-bit grayscale, non-interlaced
$ display overall.png
```

See the Automated Processing Systems User's Guide for another interactive example of this program's usage.

Name

apsScripts — Bourne shell scripting functions

Decription

This file contains a series of Bourne shell script functions which are useful for the writing scripts needed to process satellite data.

Note: All shell variables are “global” so the *apsScripts* functions uses the convention that all variables that are intended for use *outside* the script function start with a capital letter. All lowercase variables are “local” Unfortunately, this does not prevent them from being overwritten by other script functions called within a particular script function.

apsAddToDatabase file directory

The script function *apsAddToDatabase* is intended to be used by shell scripts to move *file* to a simple directory-type database.

The function will exit if *file* is not a regular file, or if path given as directory already exists but is not actually a directory. If the variable *DataPerms* is defined, then *chmod* is run to change the its permissions.i Next, the file is copy to the *$SIPRNETOutDir* if defined. The directory argument is now checked. If the given directory path does not exist, it is created. Next, the file is compressed using the command as defined by the *$CmpOpt* variable. If the variable does not exist, it defaults to the UNIX **touch** command (that is, no compression is performed.) Then *directory* is checked for existence and is created (with any missing limbs) if it does not. Finally, the (compressed) file is moved to the directory.

apsAddToWWW file www_dir

The script function *apsAddToWWW* is intended to be used by shell scripts to move the file to a directory controlled by a Web server. If the *www_dir* is of the form *hostname:directory*, i.e., it contains a colon ":", then *file* is remotely copied (using *rcp*) to that directory. Otherwise it is simply moved to that directory using *mv*. If *www_dir* does not exist, the *apsMakeDir* script function is called to create it.

If *file* is copied to a remote machine, it's permissions are changed to those indicated by the variable *WWWPerms* (if defined) before being copied. Otherwise, the *DataPerms* is used (again only if it is defined).

apsAppend var str

Appends to the string *str* to the variable *var*.

apsCheckDir file

The script function *apsCheckDir* is intended to be used by shell scripts to determine if the argument represents an directory or not. This script calls *apsFatalErr* (which terminates the script) if the directory does not exist or is unreadable.

apsCheckExec

The script function *apsCheckExec* is intended to be used by shell scripts to determine if the argument represents an executable file or not. This script calls *apsFatalErr* (which terminates the script) if program is not a regular file or not executable.

apsCheckFile file

The script function *apsCheckFile* is intended to be used by shell scripts to determine if the argument represents an file or not. This script calls *apsFatalErr* (which terminates the script) if the directory does not exist or is unreadable.

apsFindExec program

This function searches through the colon-seperated directory list of the *PATH* environment variable to determine full pathname of the given program. If not found, *apsFatalErr* is called and the script is terminated.

apsFatalErr arg1 arg2

The script function *apsFatalErr* is intended to be used by shell scripts when a fatal error has occured. The arguments are passed to *apsLogMsg* and *exit 1* is called, thereby, terminating the script.

apsInfo ifile zLat zLon

The script function *apsInfo* is used to gain information about an input file. The satellite specific routine initialization (*avhInit*, *swfInit*, etc.) routine should be called prior to this function. These will define the satellite specific info program in the variable *$ApsInfo*.

If the variable *$DayLight* is not defined (set in *apsInit*), then the output variable *$Day* is set to 9 (unknown).

These variables are created by this function. These can (and are) used by other scripts to for such things as the default image data directory structure.

Year	Set to Year (YYYY) of data file
DoY	Set to Day of Year (DDD) of data file
DoM	Set to Day of Month (DD) of data file
Month	Set to Month (mmm) of data file (e.g. 'jan')
Time	Set to Time (hhmmss) of data file
Hour	Set to Hour (hh) of data file
Min	Set to Minute (mm) of data file
Sat	Set to Satellite ID (sss) of data file
MonthNumber	Set to Month (1-12) of data file
Day	Set to 1 for Day, 2 for Night, or 3 for Day/Night or twilight To compute *Day*, the variables *mapCLat*, *mapCLon*, *mapC1Lat*, *mapC1Lon*, *mapC2Lat*, *mapC2Lon*, *mapC3Lat*, *mapC3Lon*, *mapC4Lat*, and *mapC4Lon*, need to be defined. These are normally defined by *apsInit* by calling *apsMapInfo*.

apsInit

The script function *apsInit* is intended to be used by shell scripts to set up some shell variables used by other functions within the file apsScripts. These allow for system differences to be incapsulated in a single

location. The use of variable names for executables also allows the user to replace an executable to a modified one for testing new algorithms with little change to the standard scripts.

These variables are set using the environmental variables that are usually defined in *aps.conf* file by a script executed as a child process of the APS driver process. In a user script, these may be overwritten before any call to *apsInit*.

ApsDir	Set to APS_DIR environment variable
ApsBin	Set to APS_BIN environment variable
ApsLib	Set to APS_LIB environment variable
ApsData	Set to APS_DATA environment variable
ApsWork	Set to APS_WORK environment variable
ApsOut	Set to APS_OUT environment variable
ApsError	Set to APS_ERROR environment variable
ApsIn	Set to APS_IN environment variable
ApsLock	Set to APS_LOCK environment variable
ApsDataBase	Set to APS_DATA_BASE environment variable
ApsImagBase	Set to APS_IMAG_BASE environment variable
ApsThumbBase	Set to APS_THUM_BASE environment variable
	These variables are set by using the *apsFindExec* script function to search the *PATH* environmental variable.
Remove	Set to fullpath of **rm** command with `*-f*' appended
Move	Set to fullpath of **mv** command
Copy	Set to fullpath of **cp** command
RCopy	Set to fullpath of **rcp** command
Rsh	Set to fullpath of **rsh** command
Chmod	Set to fullpath of **chmod** command
Chgrp	Set to fullpath of **chgrp** command
MkDir	Set to fullpath of **mkdir** command
Touch	Set to fullpath of **touch** command
Tar	Set to fullpath of **tar** command
Cat	Set to fullpath of **cat** command
	These variables define the various compression options known by the APS. Most are found by using the *apsFindExec* script function to search the *PATH* environmental variable.

CmpOpt	Set to "none".
CmpNone	Set to fullpath of **touch** command
CmpNoneExt	Set to an empty space.
CmpCompress	Set to fullpath of **compress** command with `*-f*' appended
CmpUnCompress	Set to fullpath of **uncompress** command with `*-f*' appended
CmpCompressExt	Set to *.Z*
CmpGzip	Set to fullpath of **gzip** command with `*-f*' appended
CmpGunzip	Set to fullpath of **gunzip** command with `*-f*' appended
CmpGzipExt	Set to *.gz*
LogFile	Set to `/dev/null`
DirPerms	Set to 755
DataPerms	Set to 644
WWWPerms	Set to 644
DayLight	Set to **ApsBin/daylight**
FileFmt	Set to **ApsBin/filefmt**
Gregor	Set to **ApsBin/gregor**
Hdf	Set to **ApsBin/hdf**
Maps	Set to **ApsBin/maps**
ApsCatalog	Set to **ApsBin/apsCatalog**
MapFile	Set to `ApsData/maps.hdf`
DefaultMapFile	Set to `ApsData/maps.hdf` (can not be overwritten)

apsIsLeap year

This script function requires one argument: a year. It will return a successful exit status if the year is a leap year and unsuccessful if not.

apsLock lockfile

The script function *apsLock* is intended to be used by shell scripts to provide a locking mechanism when certain functions must only run atomically. This function will continuously try to obtain the lock. Used in combination with the *apsUnlock* function.

apsLogMsg arg1 arg2 ...

The script function *apsLogMsg* is intended to be used by shell scripts to write time stamped messages to a log file. The arguments are passed to echo and redirected to the *LogFile* variable. If *LogFile* is not set, *apsLogMsg* as no effect.

apsMakeBathy file

The script function *apsMakeBathy* is used to create a bathymetry file for the specific area of interest in file. The bathymetry file will be stored in the $ApsData/areas directory as ${MapName}_bathy.hdf. If this file already exists and is readable, *bathyOpts* is set. Otherwise, the file is created and smoothed. The *bathyOpts* is added to the creation of the browse images.

apsMakeDir directory

The script function *apsMakeDir* is intended to be used by shell scripts to create the complete directory path, recusively. Each part of the directory is analyized for existence. Any part that does not exist is created.

If the directory path is of the form *hostname:directory* then the *Rsh* command will be used to attempt to create this directory on the remote system provided it has permissions.

Currently, this function cannot handle relative paths.

apsMakeMine file

The script function *apsMakeDir* is intended to be used by shell scripts to make the script user the owner of the file if not already owner. Additionally, it sets the permissions to 444 (read only).

apsMapInfo

This script function *apsMapInfo* makes successive calls to the *Maps* program to obtain infromation about the map. It creates the variables *mapSamples*, *mapLines*. Additionally, it extracts the lat/lon location of the four corner points and center to be used by apsInfo to determine if the file is day, night, or twilight.

apsMkTemp variable file-prefix

This script function is used to return a unique temporary filepath in variable

apsProdList

This script function is used to set the following five environment variables: L3ProdList, L4ProdList, L4ProdCompFunc, L3BrowseList, and L4BrowseList.

L3ProdList, L4ProdList, L3BrowseList, and L4BrowseList may be preset. If they are not preset, this function will set their values by reading the configuration files described respectively by the environment variables: L3ProdFile, L4ProdFile, L3BrowseFile, and L4BrowseFile. If these configuration file environment variables have not been preset, they will default to $ApsData/$Sensor/${Sensor}_def_l2prod.dat $ApsData/$Sensor/${Sensor}_def_l4prod.dat $ApsData/$Sensor/${Sensor}_def_l2browse.dat and $ApsData/$Sensor/${Sensor}_def_l4browse.dat respectively.

The L4ProdCompFunc environment variable is a string which will be used with the imgMean compositing program describing which function to apply to the product data when compositing. An entry in this string will be made for each product in L4ProdList. If an environment variable of the form <p>_CompFunc, where <p> is the name of a product in L4ProdList, exists then the value of that environment variable will be appended to L4ProdCompFunc for that product. If the <p>_CompFunc environment variable does not exist a zero will be appended for that product function.

apsSIPRNET file

This script is used to copy the attributes from the file into another file with the same name but in the *$SIPRNET* directory for manual transfer by user.

apsSetClimatology

This script function is used to set the following environment variables: MET1, MetDir1, MET2, MetDir2, MET3, MetDir3, OZONE1, OzoneDir1, OZONE2, OzoneDir2, OZONE3, OzoneDir3, use_climatology, and clean_climatology.

If UseClimatology is unset the two functions *apsSetMet* and *apsSetOzone* will be called to set the `MET` and `OZONE` variables respectively. If UseClimatology is set or if either *apsSetMet* or *apsSetOzone* fail, the "MET" and "OZONE" variables will be set to the paths of default files containing climatological values and the environment variable use_climatology will be set to 1.

if UseLocalDisk is set and UseClimatology is not set and use_climatology is not set, then a local copy of the files described by the "MET" and "OZONE" environement variables will be made and clean_climatology will be set to 1.

This function requires that the ApsWork and ApsData environment variables be set for proper operation in addition to any requirements of the *apsSetMet* and *apsSetOzone* functions.

apsSetCompress name

This script function will set the *opt* and *optExt* variables based on the input name. The input name can be one of `compress`, `gzip`, `bzip2`, or `none`. Otherwise, it defaults to *opt* to **touch** and *optExt* to a blank string.

apsSetMet

This script function is used to set the following environment variables: MET1, MetDir1, MET2, MetDir2, MET3, and MetDir3. The appropriate MET files will be determined based on the following environment variables which must be preset: Year, DoY, Month, Hour, mons, and ANCIL_DIR. The mons variable is an array of month names. If any of the determined MET files do not exist, use_climatology will be set to 1.

apsSetOzone

This script function is used to set the following environment variables: OZONE1, OzoneDir1, OZONE2, OzoneDir2, OZONE3, and OzoneDir3. The appropriate OZONE files will be determined based on the following environment variables which must be preset: Year, Doy, Month, mons, and ANCIL_DIR. The mons variable is an array of month names. If any of the determined OZONE files do not exist, use_climatology will be set to 1.

apsTmpCopy var ifile oprefix

This script function is used to copy a file to a temporary location. *apsMkTemp* will be called to create a temporary file name. This script function takes three arguments. The first is the name of a variable which will be set to the filename of the created copy. The second is the name of the file to be copied, and the third is the prefix of the temporary filename which will be modified by *apsMkTemp*.

apsTrylock lockfile

The script function *apsTryLock* is like *apsLock* except that is tries only once to obtain the lock.

apsUnlock lockfile

The script function *apsUnlock* is intended to be used by shell scripts to provide a locking mechanism when certain functions must only run atomically. Used in combination with the *apsLock* function.

Name

apsDownloadMailFilter.rb — filters e-mail to register downloads

Synopsis

```
apsDownloadMailFilter.rb
```

Description

The program **apsDownloadMailFilter.rb** parses e-mail messages from several outside agencies that can provide data. It is not called directly but by creating a procmailrc file and using a .forward file.

NASA GSFC Data Notifications

The MET/OZONE data is currently obtained from NASA Goddard. The messages from this site follow this pattern.

```
Subject: Data Ready for FTP--Subscription 291
[snip]
Or, you can download the whole directory as a TAR file with one click using
URL: ftp://eosdata.gsfc.nasa.gov/.ops/dist/16384/R180860_58.485.5052.tar
[snip]
Files Ready for Transfer:
        S200303700_NCEP.MET.Z  (486 KB)
        S200303706_NCEP.MET.Z  (491 KB)
        S20030370003723_TOVS.OZONE.Z  (49 KB)
        S20030380003823_TOVS.OZONE.Z  (49 KB)
        S200303712_EPTOMS.OZONE.Z  (44 KB)
        S200303812_EPTOMS.OZONE.Z  (44 KB)
```

Goddard DAAC ECS Notifications

The MODIS data ordered NASA Goddard DAAC has the following messages:

```
Dear User,

THE DATA YOU ORDERED HAVE BEEN STAGED FOR PICKUP AND WILL BE AVAILABLE
ONLY UNTIL THE FTP EXPIRATION (FTPEXPR) DATE LISTED BELOW.
[snip]
301-614-5304 [Fax]
Email: help@daac.gsfc.nasa.gov



++++++++++

ORDERID: 0400088092
REQUESTID: 0400088105
USERSTRING:
FINISHED: 07/29/2003 12:51:37
```

```
MEDIATYPE: FtpPull
FTPHOST: g0acg01u.ecs.nasa.gov
FTPDIR: 0400088105UDXqhB
FTPEXPR: 07/31/03 14:16:43
MEDIA 1 of 1
MEDIAID:

     GRANULE: UR:10:DsShESDTUR:UR:15:DsShSciServerUR:13:[GSF:DSSDSRV]:24:SC:MOD02H
     ESDT: MOD02HKM.004

             FILENAME: MOD02HKM.A2003204.1740.004.2003205210034.hdf
             FILESIZE: 275066720

             FILENAME: MOD02HKM.A2003204.1740.004.2003205210034.hdf.met
             FILESIZE: 55213

     GRANULE: UR:10:DsShESDTUR:UR:15:DsShSciServerUR:13:[GSF:DSSDSRV]:24:SC:MOD021
     ESDT: MOD021KM.004

             FILENAME: MOD021KM.A2003204.1605.004.2003206125634.hdf
             FILESIZE: 343362608

             FILENAME: MOD021KM.A2003204.1605.004.2003206125634.hdf.met
             FILESIZE: 57156
```

The same type of e-mail messages (ECS-like) are produced by a cron job on the anon ftp server for transferring SeaWiFS data from NAVO, MODIS data from NOAA, and MODIS data from HNSG.

Name

apsDownloadManager.rb — manages uploads/downloads from/to aps

Synopsis

```
apsDownloadManager.rb
```

Description

The program **apsDownloadManager.rb** uses an SQL data base to handle the transfer of data to and from aps. The data base contains two main tables: “uploads” for aps data being sent to other users and "downloads" for obtaining outside data that is to be processed by the aps.

For downloads, the user can obtain the file using one of three protocols: ftp, http, or scp. Once the file has been obtained, it might be compressed or uncompressed. If the download fails part-way through, **apsDownloadManager.rb** can resume from previous point at a latter time. The downloads can have a priority. For example, *apsDownloadMailFilter.rb* prioritizes the MODIS data in this order (MOD03, MOD021KM, MOD02HKM, MOD02QKM). The SeaWiFS data has next priority, followed by the MET/OZONE data which has lowest priority. If the file is to be processed, the file is given to the aps for processing.

For uploads, the user can either ftp it to another host, or send it as an e-mail attachment (mailto protocol). For HDF data, the preferred method is ftp. In addition, an “NRL Notification” e-mail can be issued once the file has be uploaded. For images, the mailto protocol can be used. This allows users to obtain images via e-mail as processed by the aps.

Classes

DownloadManager

This class is used to perform the uploading and downloading of data from aps. It contains the following methods:

process
: This method is the main method. It queries the *aps_file_transfer* data base for a list of files which are to uploaded or downloaded. It will thread out up to *max_threads* (presently 10) concurrent file transfers. It will continue until all possible file transfers are complete.

download
: This method handles the downloading of a single entry from the *aps_file_transfer* database. If verifys the local file (directory or if pre-existing), calls *fetch* to obtain file, calls *postprocess* if any post processing is required. If the entry is to be processed by aps, *processfile* is called. This method handles updating the data base entry.

download_local_file(file)
: This method verifies that the local file path is valid. If the file pre-exists, the resume flag is set. The resume flag allows *fetch* to resume partial downloads.

fetch
: This method handles the downloading of a single entry from the *aps_file_transfer* database. It calls one of three protocol specific fetchs.

scp_fetch This method will fetch a file using *scp*.

ftp_fetch This method will fetch a file using anonymous *ftp*.

http_fetch This method will fetch a file using HTTP.

postprocess This method handles post processing of the downloaded file. It can be used to compress or uncompress (using **compress**, **gzip**, or **bzip2**) the downloaded file.

processfile This method moves the downloaded file to aps.

upload This method handles the uploading of a single entry from the *aps_file_transfer* database. It verifies that the local file exists, then either calls *mailto* or *pitch* depending on the protocol. It handles updating the data base entry for this file. If a notication of uploading is desired, the *upload_msg* will be called.

upload_local_file This method verifies that the local file to uploaded can be opened for reading.

mailto This method attaches the local file to an e-mail message and sends to addressee.

pitch This method handles the uploading of a single entry from the *aps_file_transfer* database. It calls *ftp_pitch*.

ftp_pitch This method uses FTP to push a file to an anonymous ftp site.

preprocess This method handles pre processing of the uploaded file. It can be used to compress or uncompress (using **compress**, **gzip**, or **bzip2**) the file to be uploaded. NOTE: It will act upon the local file.

upload_msg This method will issue an e-mail to addressee indicating the file was successfully or unsuccessfully uploaded to user's ftp machine.

Files

`apsDownloadManger.lock` This file is located in the $APS_DIR/var/lock and is used to prevent multiple copies of aps from running.

Name

apsDownloadSearch.rb — search for data on remote hosts

Synopsis

`apsDownloadSearch.rb` [*`config`*]

Description

The program **apsDownloadSearch.rb** uses an XML configuration file to examine download hosts for new data which are registered into a data base through e-mail (see apsDownloadManager.rb) or pulls down the data itself. The XML file provides methods for limiting searches using regular expressions based on the filenames of the data to be obtained.

It will maintain a list of files that have been downloaded previously and only register or fetch those data that match the critera in the XML file. By searching backwards it can maintain data access when connectivity to the remote host has been severed.

XML file

The XML file consists of multiple “download” entries which provide the required information to download a set of data. Within each “download” entry, the XML file must provide information about the host, location, file searching, and protocols used to find and register the data.

XML elements

The basic XML file structure contains a downloads entry which will contain one or more download entries. Each download entry is may have the following attributes.

`downloads` This entry is the main container for the XML document. It must be present.

`download` This entry will contain all the XML elements that are required to search a single download source. There may be multiple entries in the root download entry. The following attributes may be set for this entry.

- `name` This attribute provides a descriptive name for the download entry. It is used as the prefix for the listing file that maintains a list of previously downloaded files.
- `active` The active attribute is used to turn the particular entry on or off. The only valid values are "yes" or "no".

download elements

These entries are part of the download entry.

`host` Each download entry must contain a host entry that indicates the ftp host to get data from. If a user and password are required, then appropriate user and password entries should be provided. By default 'anonymous' and 'aps@nrlssc.navy.mil' will be used.

`user` Username for provided host. If not defined, will default to 'anonymous'.

`password` Required password for provided host. If not defined, will default to user@hostname.

`chdir` If the desired data resides in a particular directory, this entry may be provided. This entry may contain special control entries to allow for dynamic directory locations. This special characters are described below.

`listing` This entry provides the ftp LIST glob pattern to be used to list the files to be obtained. This entry may contain special control characters to allow for dynamic listing options.

`regexp1, regexp2` These entries are used to filter listing file and the ftp glob statement respectively to reduce both lists to common formats. This must be done as these common lists are compared to determine which new data needs to be obtained.

`aps_dir` This entry is used to push the data to the APS directory indicated. This is most useful for data that will be downloaded and processec by APS in real-time.

`back` The value of this entry is used to go back X number of days. The default value is 3.

`wait` The value of this entry represents the number of seconds to wait before attempting to reconnect to the ftp host. Some hosts require a longer wait period before you can reconnect to them. The default value is 10.

`type` This entry indicates the method to use to aquire the data. There are only two valid options: "mail" and "ftp". The "mail" option is the default.

`time_out` The value of this entry represents the time out in seconds for an ftp download connection. If the transfer does not complete in the number of seconds alotted it will sever the connection. The default value is 240.

regular expressions

The following regular experssions may be used in the above entries.

`%Y` Is replaced by current 4-digit year.

`%J` Is replaced by current 3-digit day of year.

`%M` Is replaced by current 2-digit month.

`%D` Is replaced by current 2-digit day of month.

Files

`apsDownloadSearch.lock` This file is located in the $APS_LOCK and is used to prevent multiple copies of apsDownloadSearch.rb from running.

`name.listing` This file contains the listing information from the download host with the given name. This file is located in the `$APS_VAR/download` directory.

Environment Variables

`$APS_LOCK` Defines the location for the lock file. If not found, will be set to `$HOME`.

`$APS_ETC` Defines the location for XML file. If not found, will be set to `$HOME`.

Examples

Example 1.

This example will download all the most recent MODIS Aqua geo-location data from the Goddard host. It will search backwards up to 4 days and has a wait value of 20 seconds. The data is accessed using ftp.

If this is run on June 20th, 2009, it will first use anonymous ftp to login to `oceans.gsfc.nasa.gov` and move the directory `/MODISA/Recent_Day/Quicklook/2009171` on that host. Next, it will use the UNIX ftp glob of "A2009171*GEO" to access all data that satisfy that each. For each file that has not previously been downloaded, it will downloaded and place in the APS input directory `/home/aps/in`. After all data in this directory has been examined, it will then move to the `/MODISA/Recent_Day/Quicklook/2009170` and examine all files that match "A2009171*GEO", downloading any that were not previously retrieved. It will continue to look for data until day 169.

```
<downloads>

<download>

<host>oceans.gsfc.nasa.gov</host>


<user>anonymous</user>


<password>email</password>


<chdir>/MODISA/Recent_Day/Quicklook/%Y%J</chdir>


<listing>A%Y%J*GEO</listing>


<regexp1>(.*) (A%Y%J.*GEO)</regexp1>


<regexp2>(A%Y%J.*GEO)</regexp2>


<back>4</back>


<wait>20</wait>

        ftp
    </download>

  </downloads>
```

Name

FileTransferDB.rb — manages aps SQL file transfer data base

Synopsis

```
FileTransferDB.rb
```

Description

This Ruby library **FileTransferDB.rb** is a class that communicates with the PostgreSQL data base *aps_file_transfer*. This data base is used to upload processed data from the aps and download data to the aps. This Ruby script does not actually perform the transfer, but is used to manipulate the *aps_file_transfer* data base.

Options

-U, --upload local_file URI

Register a local file to be uploaded to a URI. For example,

```
FileTransferDB.rb --upload MODPM2003335194500.L3_MSB_chl_
                  mailto:zero@dev.null
```

will register the given file to be e-mailed to zero@dev.null.

These commands will register the same file to be transferred to the NRL anonymous ftp server. The second command *apsDownloadManager.rb* actually does the transfer.

```
FileTransferDB.rb --upload MODPM2003335194500.L3_MSB_chl_oc3m.jpg \
    ftp://ftp.nrlssc.navy.mil/pub/incoming/aps/MODPM2003335194500.L3_MSB_chl_oc3m.
apsDownloadManager.rb
```

Note: A FTP URI should contain the full path name (including filename). Also, in general, the input file should be an absolute path. Otherwise, *apsDownloadManager.rb* will need to be run in the directory where the file is located.

-D, --download URI local_file

This will register a URI to be downloaded to a local file. For example,

```
FileTransferDB.rb --download http://www7333.nrlssc.navy.n
apsDownloadManager.rb
```

will add an entry so that the aps Users Guide will be downloaded to the $HOME account and renamed aps_users_guide.pdf.

-A, --process

This option will cause the downloaded file to be placed into the aps for processing. Note that the file should not be downloaded directly into the *in* directory of the aps. Please use the *ftp* directory.

-a, --addressee

This option will issue an e-mail to all addressees given once the a file has been uploaded. This will give the recpient a notification of successful upload.

-p, --postproc command

This option selects a post processing option to be performed on the file once downloaded or uploaded. It may be set to *gzip*, *gunzip*, *compress*, *uncompress*, or *tar.gz. For example, a file may be compressed after downloaded. Note:* For an uploaded file the user may not have permissions or capability to perform the post processing command.

-p, --preproc command

This option selects a pre processing option to be performed on the file once downloaded or uploaded. It may be set to *gzip*, *gunzip*, *compress*, or *uncompress*. For example, a file may be compressed before uploading. *Note 1:* For a downloaded file the user may not have permissions or capability to perform the pre processing command. *Note 2*: If uploading a local file, the pre processing command acts on the local file -- it does not make a temporary copy.

Classes

TableEntry

The TableEntry class is a reflection of the core SQL tables for the file transfer database. Each instance of the class represents a single row in the SQL database. It has the following entries:

column	type	description
id	integer	SQL id
queued	timestamp	Time when entry added
finished	timestamp	Time when transfer completed
attempts	integer	number of attempts to transfer
remote_path	varchar(256)	path of remote file or 'subject' if protocol is 'mailto'
local_path	varchar(256)	path of local file
local_file	varchar(128)	name of local file
protocol	varchar(8)	protocol (`http` `ftp` `scp` or `mailto`
host	varchar(128)	host
user	varchar(128)	user
password	varchar(128)	password
file_size	bigint	size of file
aquired	bigint	number of bytes aquired
postproc	varchar(128)	post processing of file
preproc	varchar(128)	pre processing of file
priority	integer	priority of entry
status	integer	status
done	boolean	transfer completed
delete	boolean	delete remote (downloading) or local (uploading) file
process	boolean	process this data by aps

mailto	varchar(256)	addressee of NRL upload Notifications

remote_uri This method the entry as a URI.

sql This method returns the entry as an SQL INSERT formatted string.

to_ary This method returns the entry.

DownloadTableEntry

A subclass of TableEntry class for “downloads” SQL table.

UploadTableEntry

A subclass of TableEntry class for “uploads” SQL table.

FileTransferDB

An instance of FileTransferDB class makes an SQL data base connection to the *aps_file_transfers* data base. It's methods handle all communication to and from the SQL data base.

It has only two public variables: conn and params

conn This read only variable represents the PostgreSQL connection

params This read only variable contains parameters

mail_host	SMTP host used for `mailto` protocol
max_threads	Maximum number of threads for downloading
aps_dir	Location of aps 'in' directory for data to be processed

The FileTransferDB has the following methods

each(table,upto=nil,&block) Call the block for each entry in the SQL data base for given table (up to *upto*) passes an TableEntry from each SQL data base entry which has a not-in-progress status and is flagged as not done. The SQL entries are ordered by priority.

add_entry(entry) Add an instance of a TableEntry to the SQL data base

get_entry(table,id) Return an instance of a TableEntry of the given SQL id from the given table (“uploads”, or “downloads”)

mark_as_done(entry) Update the SQL data base of this entrying settind done to true

mark_as_trying(entry) Update the SQL data base of this entrying by setting status to -1.

update_aquired(entry,size) Update the SQL data base for this entry by setting number of bytes quired to size.

update_status(entry,status) Update the SQL data base for this entry by setting the status

db_error()	Print PostgreSQL error message and exit process
create_dl_table()	Used to call SQL commands needed to created the “downloads” table. *Use only for initialization of a new database*.
create_ul_table()	Used to call SQL commands needed to created the “uploads” table. *Use only for initialization of a new database*.
delete_cb()	Used to call SQL commands to delete the data base. *This function will destroy* the data base.

Chapter 9. Release Notes

The following notes are summaries of the changes to the APS software for each release. The most recent changes are at the top.

Release v3.8.2.3-72-g7b866d

The release date was released on 18 November 2009.

- Field Spectrometer Processing of SPECTRIX data

  The **fldspec** program had an issue in which the very last spectrum was never used due to the way the program counted the number of spectra in the file.

Release v3.8.2

The release date was September 8th, 2009.

- MERIS 300m FRS L1 Processing

  The MERIS L1 300m processing has been enabled.

- MODIS L1A Processing

  The programs required to process MODIS from L1A has been installed. If a L1A file is placed in the input directory, it will process for that region of interest.

- NOAA-19 Processing

  The AVHRR sensor processing from NOAA-19 has been enabled. The SST coefficients come from NAVOCEANO.

- maps program updates

  The maps program has been updated to add several new options including an edit. Other new options include boundary, copy and zoom.

- imgBrowse updates

  Many modifications to imgBrowse to improve output image quality, grid line drawing, etc.

Release v3.8.1

The release date was March 12th, 2009.

- MERIS L2 Processing

  The MERIS L2 processing was modified to fix output of MERIS CLDICE/ LAND flags to 'l2_flags' product. The MERIS L2 processing includes the addition of the Stumpf 412 iteration for correcting negative radiance values. The handling of the MER_FR__2P files was corrected.

- MERIS L1 Processing

The ability to process MERIS L1 has been installed. The Rayleigh tables have been uniquely generated for the MERIS sensor (Note: the 412 Rayleigh table contained some bogus data and was replaced with the OCM 414 Rayeigh table).

- QAA v5

  The QAA algorithm has been updated to version 5. In this version, the 640nm was been removed and replaced with the 670 nm channel.

- Processing Version

Table 9.1. Processing Version

Sensor	APS v3.8.0	APS v3.8.1
avhrr	v05	v05
meris	v03	v04
modis	v07	v08
ocm	v03	v04
seawifs	v07	v08

- Field Spectrometer

  The field spectrometer processing code was renamed from rrs_v2.0 to fldpsec. The program now uses the relative spectral response functions from the l2gen program. It can produce OCM and MERIS convolved output files. The QAA output was updated to v5 of the algorithm.

- SQL data base

  The SQL data base tables were redesigned to improve data base integrity and the libraries were modified to reflect the table redesign; improve software maintainence; improve library integrity; handle MERIS L1/L2, HDF (TeraScan-based) OCM L1B files and HDF (TeraScan-based) AVHRR files; and update SeaWiFS L1A, MODIS L1 code which had fallen in disarray.

- APS Plotting

  The APS Plotting functionality was improved by correcting the plotting of log10 data on an axis, proper clipping of lines outside plot, and the addition of the star and triangle symbols.

- Sensor Merging

  The sensor merging functionality was fixed to handle new file nomenclature and perform the 7-day latest pixel compositing.

- Memory Leaks

  Using the valgrind program the use of memory by APS was improved.

- APS Ruby reprocessing

  The reprocessing option using the Ruby scheduler was fixed and allows multiple reprocessings to occur simultaneously. The shell APS scheduler is sequential only.

Release v3.8

- APS netCDF/HDF4/HDF5 library

  The APS input/output library was re-written so that the underlying format could be a netCDF or HDF (v4 or v5) file. The new libapshdf library reads and writes netCDF/HDF4/HDF5 files. The new libapsrs library is a higher level library for reading 'remote sensing' files The new libapsnav library contains all the navigation codes have been gathered together in this library The new libapsnavrs library contains the IO portion of the navigation library based on the libapsrs library The origina libapsio library was removed.

- Ocean Color Level-2 Processor Updates

  The **l2gen** was updated from v5.5.3 to v5.8.2 and includes the following updates.

  - new K_PAR_lee products
  - new Kd_morel products
  - new BT_XXX products
  - new Wang/Shi SWIR/MS78 atmospheric switching
  - new MUMM atmospheric correction
  - new empirical Carder chlorophll-a product
  - new Zeu_lee product
  - new euphotic depth algorithms from Morel
  - new GSM01 using Levenberg Marquart fit option
  - new NO2 climatology
  - new SeaWiFS calibration
  - new poc_stramski product
  - new brightness temperature products (all thermal bands)
  - new SMA algorithm
  - fix MERIS processing of Level-2 files

- Ruby controller

  - divided into parts (installed in ruby/lib/1.8/aps)
  - replace original logging with Log4r
  - add '"--debug"' option to allow lots of logging
  - new configuration using XML
  - allow standard defaults when no configuration file used

- Documentation

  The APS documentation is now based on DocBook which provides three available outputs which APS includes in the `share` directory. These include man pages, HTML documentation, and PDF.

  Additionally, all API documentation is available in HTML using the Doxygen processor.

- Remove Miami codes

  The MODIS processing has been upgraded to use the NASA Goddard SeaDAS based processing for both ocean color and sst products.

- Remove GLIB dependency

  The GLIB library is a third party library used through the GNOME development. However, APS only used a small part of this library (data structures). Those parts of the original library were removed and placed into a "Small GLIB" library which is part of APS.

- new imgFillGaps

  This new program is designed to fill-in areas of cloud cover and/or missing data (no overpass).

- imgMap2 program replaces imgMap

  The **imgMap2** program was a developmental version which used new methods for mapping data files to a projection. It was designed with the MODIS sensor in mind. It now becomes the official APS warper.

- Separate site-specifics

- AVHRR Processing

  - changed to use the new libapshdf library
  - changed to use libgs rather than GLIB
  - handle bad times in NESDIS Level-1 format
  - update KLM format to handle version 3 format
  - update SST coefficients for NOAA-17, NOAA-18, and METOP-01 using latest NAVOCEANO regressions
  - changes for METOP processing the albedo channels work well the brightness temperatures seem to be having problems
  - updates to avhTurbid rudimentry estimates of horiz_vis from cp_660

- Processing Version

Table 9.2. Processing Version

Sensor	APS v3.6	APS v3.8
avhrr	5.0/v05	5.0/v05
meris	1.2/v02	3.0/v03
modis	5.4/v06	7.0/v07

Sensor	APS v3.6	APS v3.8
ocm	1.4/v02	3.0/v03
seawifs	5.4/v06	7.0/v07

Note

Prior to APS v3.6, only one file nomenclature of APS data files was available and verison numbers had “minor” release values. From APS v3.8 on, the version numbers will increment by “major” values only and consist of positive integers.