

The Automated Satellite Data Processing System

Data Base/WWW Features

The Automated Satellite Data Processing System: Data Base/WWW Features

Published 28 March 2008

Table of Contents

- I. Data Base and Web Features 1
- 1. Main Table 1
- II. Command Line Reference 2

List of Tables

1.1.	1
-----------	---

Part I. Data Base and Web Features

The chapters in Part I form a User's Guide for the web and data base features available within the Automated Processing System.

Chapter 1. Main Table

The main table is a representation of each and every HDF file produced by the APS and stored under the /rs/lvl[1,3,4,5] directory. This includes all L1 passes and granules, L3 data for each area of interest, any L3 Mosaics, and all L4 and L5 files.

Table 1.1.

Number	Type	Description
main_id	added	ids generated automatically by database
added	timestamp	when this entry was initially added
modified	timestamp	last time entry was modified
update_count	int8	number of times entry was modified
file	varchar(128)	name of the hdf file (basename)
path	varchar(256)	path of the hdf file (dirname)
stime_t	timestamp	start time of data in ISO 8601 format
etime_t	timestamp	end time of data in ISO 8601 format
proc_ver	int	processing version
file_ver	int	file version
status	enum	status of file: OPERATIONAL or DEVELOPMENTAL
klass	int	see classification table
lvl_table	varchar(16)	name of level table
lvl_id	level_id	see level table named above
sensor	sensor_id	see sensor table
ftype	filetype_id	see file_type table
prod_ids	prod_ids[]	see product table (array)
image_ids	image_ids[]	see image table (array)

Part II. Command Line Reference

The chapters in Part II form a reference guide for each program available in the Automated Processing System for data base and web features.

Name

apsCatalog -- register an APS file to the SQL database

apsCatalog
apsCatalog [*options*] *file*

Description

The **apsCatalog** program is used to register (insert/update) a file into the SQL database *db*. Currently, all level 3 and 4 data are supported.

Options

- B
Set browseList to given string.
- d
Uses the database given as the source/destination.
- D
Sets the Debug option to display debugging information (SQL commands) to stderr.
- f
Get input files from given file. 1 per line. Must be usable path.
- g
If the file ends in .gz, the file is unzipped to specified directory, and removed when finished!
Do not unzip to same directory or the file will be lost!
- n
Only catalogs new files. No updating of information is done for files already in the database.
- S
Set station code for level1 data
- v
Sets the verbose option to display processing information to the screen.
- help
Print out a small help page.
- version
Print out version of software and quit.

Examples

In this example, we add a valid APS file to the data base with debug output turned on.

```
$ apsCatalog -D S2003009185200.L3_HNAV_GOM
Debug: SELECT aoi_id FROM aoi WHERE sensor_id=101 AND name='GulfOfMexico'
Debug: SELECT prod_id from products WHERE name='rrs_412' and sensor='SeaWiFS'
Debug: SELECT prod_id from products WHERE name='rrs_443' and sensor='SeaWiFS'
\.
\.
\.
Debug: SELECT main_id FROM main WHERE file='S2003009185200.L3_HNAV_GOM' AND
proc_ver=24 AND file_ver=25;
Debug: UPDATE main SET modified=now(),stime_t='2003-01-09
18:55:42',etime_t='2003-01-09 19:01:21',status=0,klass=0,
prod_ids='{46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94}',update_count=update_count,
browse_ids='{52,53,57,63,68,69,72,73,77,46,47,48,50,89,90,91,92,93,94}',update_count=update_count,
WHERE main_id=0;
```

In this example, we attempt to add an invalid APS file to the Database.

```
$ touch ~/invalid_file
$ apsCatalog ~/invalid_file
apsCatalog.c: Unknown APS Filetype
```

The following example adds level1 data to the database setting the browseList and station codes and indicating that the file should be temporarily unzipped to the /tmp directory.

```
$ apsCatalog -B"swath" -S HBHR -g/tmp S2001161075634.L1A_HBHR.gz
```

The next example uses the find command to generate a list of files for apsCatalog to run on. Also the n option is specified so apsCatalog only adds new entries and does not modify any entries already in the database.

```
$ find . -type f -exec echo '{}' ';' > list.txt
$ apsCatalog -nvf list.txt
```

Name

dbCheck -- Performs integrity checks on the database and local file system and generates reports

dbCheck

dbCheck [*options*][*filename* | *stdin* | *for* | *H,I* | *options*]

Description

The **dbCheck** program runs data integrity checks on the remote sensing database. It can check entries in the database to see if they physically exist on the drives as well as check for a list of files for entry into the database. Also, certain reports can be made from this program(not yet implemented).

Options

- b
Specifies the path to the top-level directory to search from.
- d
Uses the database given as the source/destination.
- D
Sets the Debug option to display debugging information (SQL commands) to stderr.
- h
Check file entries in database against local filesystem
- H
Check local filesystem against file entries
- i
Check image entries in database against local filesystem
- I
Check local filesystem against image entries
- o
Specifies the output should be written to the given file.
- v
Sets the verbose option to display processing information to the screen.
- help
Print out a small help page.
- version
Print out version of software and quit.

Examples

This first examples show how to check that a file on the file system data base is registered with the SQL data base (that is as an entry in the main table). To see if a particular file is in the SQL data base use:

```
$ dbCheck MODPM2004011194000.L3_NOAA_MSB
```

To see if a list of files are in the SQL data base use:

```
$ dbCheck -f file.lst
```

Using find you can check all files in a directory:

```
$ find /rs/lvl3/seawifs/2.4/GulfOfMexico/2003/nov -type f -exec dbCheck {} semi;
```

Or just put the name of the directory on the command line and dbCheck will transverse down the directory.

```
$ dbCheck /rs/lvl3/seawifs/2.4/GulfOfMexico/2003/nov
```

If the file list contains images, then since images are registred with the image table, the image table needs to be check. To indicate this, use the -i option.

```
$ find /projects/browse/lvl3/seawifs/2.4/GulfOfMexico/2003/nov -type f -exec dbCheck -i {} semi;
```

The next set of options allow the user to This example checks the image entries in the database against the Filesystem starting at /projects/web/web7333/html/browse. Since the database directory entries start with the level, you should either be at that top level, or specify the path to the directory containing the level directories through the -b.

```
$ web/dbCheck -b/projects/web/web7333/html/browse -i > ~/junk.log
```

Name

insituCatalog -- inserts a SIMBIOS in situ file into the in situ SQL database

```
insituCatalog
insituCatalog [options] file
```

Description

The **insituCatalog** program is used to register (insert/update) a SIMBIOS *in situ* data in the *in situ* SQL data base.

Options

-D	Sets the Debug option to display debugging information (SQL commands) to stderr.
-v	Sets the verbose option to display processing information to the screen.
--check=type	Checks that the given input file is a valid file of type. Valid types are: simbios.
--help	Print out a small help page.
--version	Print out version of software and quit.

Examples

In this example, we insert a valid SIMBIOS file with verbose on.

```
$ insituCatalog -v LS01T_v1.2.sim
insituCatalog: Working on file LS01T_v1.2.sim
```

In this example, we insert an invalid SIMBIOS file.

```
$ touch invalid_simbios
$ insituCatalog invalid_simbios
insituCatalog: Unrecognized insitu file
```

In this example, we check that the given file is a valid SIMBIOS file.

```
$ insituCatalog --check=simbios 23may2002_flowthru_ocolor.post
Running simbios check on 23may2002_flowthru_ocolor.post
```

Error: "data_type" is not valid. Is it one of the following?
"cast" "flow_thru" "above_water" "sunphoto" "mooring" "drifter" "scan" "lidar"↵
"pigment"
Error with time. Check that time is in GMT
Error with time. Check that time is in GMT
Symbios Header is invalid.

See Also

apsCatalog(1)

Name

matchup -- match in situ data with satellite data

matchup

matchup [*options*] [*parameter file*] [*parameter=value*]

Description

The **matchup** program is used to match *in situ* data with satellite data products (e.g. chlorophyll-a, remote sensing reflectance, or total absorption). The *in situ* data is stored in an SQL data base along with time, location, observation, etc. The SQL data base will consist of a single entry for each data point uniquely based on time and location. Only the collected data that is registered into the SQL data base will be used in the match. The *in situ* data is inserted into the SQL data base using the program insituCatalog(1).

The Level-3 remote sensing data has meta data (time, location, product, etc.) stored in an SQL data base with the actual data stored in HDF files on a hierarchical file system. In order to be matched with *in situ* data, the satellite data must be registered with the SQL data base. The remote sensing data's meta-information is inserted into the SQL data base using the program apsCatalog(1).

Using several parameters passed on a command-line or given in an input file, the user determines what data (based on parameter, time, location, quality, etc.) is to be matched and how the match is computed, reported and/or plotted.

In situ to Satellite Matching

To produce a match, the user must select an *in situ* parameter (or field) to be matched with a selected remote sensing product. For example, the user might wish to match the above-water computed remote sensing reflectance with the remote sensing reflectance produced from the MODIS Aqua sensor. The collected *in situ* data may be filtered by cruise name, data type, time frame, or bounding box. See below for a complete list of *in situ* filtering criteria.

The matchup program will start with a list of *in situ* data points that passes the user's criteria and then find a matching satellite data file from the satellite SQL data base that contains that point (in time and space). Like the *in situ* data, the Level-3 satellite data may be filtered by various criteria including sensor, area of interest, file type, or data quality. See below for a complete list of remote sensing filtering criteria.

This initial matching of *in situ* data to satellite data will yield a 1:n match. Multiple satellite data files may potentially match the same *in situ* data. For example, a Gulf Of Mexico and MissBight file may exist; Two passes collected on the same day as the *in situ* data; Or, two different processing versions. The user may reduce this by selecting appropriate filtering criteria.

Gathering Data

Now that the files have been matched up, the next step is to gather the data from both the insitu file and the satellite file and produce a match. First, the wavelengths are determined from the specified rs products. Then, the corresponding insitu wavelengths are determined in one of two ways: (1) If the user chose to convolve the data, we don't determine the corresponding wavelengths, rather the spectral data is convolved with the satellite response file specified, and the appropriate values at the satellite wavelengths are used. (2) The closest matching wavelength is used as long as the closest wavelength is within 30 nm of the satellite wavelength. If the closest wavelength is farther than 30nm away, that wavelength is not used, but other matching wavelengths for the insitu and satellite files may be used. The data can be gathered for the specific (x,y) on the image, or an average of a box around the image.

Data File

If the user has asked to write out a data file via the parameter file, this is done now which just prints out a text matrix of data where the wavelengths go across alternating between insitu and satellite and the each set goes down.

```
(insitu 1,lambda 1) (RS 1,lambda 1) (insitu 1,lambda 2) (RS 1,lambda 2)...  
.  
.  
.  
(insitu n,lambda 1) (RS n,lambda n) (insitu n, lambda 2) (RS n,lambda 2)...
```

Plots

If the user has indicated for plots to be made, we make the plots here. First, the data structures from the matchup are converted to a more easily plottable form and also additional filtering is done on rs data ONLY. Additional filters include checking the data ranges against user defined values if set. Also, if the data was marked in the satellite file as invalid (DBL_MIN), these are thrown out too. Second, the plot setup parameters are read from the parameter file specifying the plot directory and base name. If the plot directory is missing, the current working directory is assumed. If no base name is given, the default is "plot". The plots can be plotted with 3 different point types: square, diamond, and point number. The last one is convenient to use to locate information about a point in the report if one is made.

Reports

There are 3 reporting levels. Each level contains the information listed plus all information listed above that level.

1. No reporting is done
2. Report for these reasons.
 - a. Number of possible matchups with the insitu data
 - b. Parameter file used
 - c. The valid range for data
 - d. How many files were flagged and for what flag
 - e. How many files had navigation failures
 - f. How many files could not be opened
 - g. Normalization factor
 - h. Invalid data, too-high, and too-low values broken down by wavelength
3. Report for these additional reasons.
 - a. Latitude/Longitude, Sample/Line, filenames, and geophysical data foreach point used on the plot along with a point number matching the number on the plot if point-type 2 was used
 - b. A list of files that had any type of error, and what error it was

Station Plot

If specified in the parameter file, this will create a plot of the world with points overlaid showing where the insitu data was taken from. Only points used on plots are shown.

Overlays

If the user specifies this option in the parameter file, we will make images of the Satellite data with points overlaid where the insitu data was located. This is done by creating a text file for each satellite file used that contains data readable by the `-p` option of `imgBrowse`. For information on the format, see the manpage for `imgBrowse` or run `imgBrowse --help overlays`. Only one picture of each satellite file is made, multiple points are put on one image if needed. This is accomplished by appending data to a file. Care should be taken to make sure none of the files already exist, or they could contain errors.

Level1 Matchup

If the user has specified to do a level 1 matchup, we take each insitu file, and query the database for any level 1 data within a time tolerance(adjustable by the user) containing the point of the latitude and longitude. No check is done for the point in the satellite file being valid. The list of level 1 files is then written out to stdout(the file specified by the user in the parameter file one file per line).

Options

- `-d` This option turns on debugging information.
- `-v` sets the verbose option to display processing information to the screen. This option can be used more than once to increase the verbosity of the program.
- `--help` Print out a small help page.
- `--version` Print out version of software and quit.

Parameter File

The parameter file will contain all the information needed to create a matchup between the *in situ* data and the remote sensing data.

in situ Data Selection

The following keywords define the *in situ* data that will be used in the match. Of these, *FIELD*, is required to define the *in situ* parameter that will be matched with the satellite data.

INSITU_DB

This parameter contains the SQL data base name that holds the *in situ* data.

INSITU_HOST

This parameter contains the hostname of the SQL data base.

FIELD

This parameter *must* be defined and sets the insitu field type to match with remote sensing data. This should be one data field type, but may be multiple wavelengths. For example, 'a_443,a_412,a_670' is valid, whereas, 'modrrs,a_412' is invalid.

FIELD_MATCH

This parameter should contain either *AND* or *OR*. It is used in conjunction with *FIELD* to limit search to all or any fields specified.

START_TIME_INSITU

This parameter defines the lower time limit for insitu files to be used.

END_TIME_INSITU

This parameter defines the upper time limit for insitu files to be used.

BOUNDARY

To restrict the initial *in situ* data selection to a geographical region, this parameter may be used. The user must define a polygon of this form: ((lon_0,lat_0),(lon_1,lat_1),...,(lon_n-1,lat_n-1),(lon_n,lat_n))

CRUISE

This parameter limits the insitu data used to specified cruises

DATA_TYPE

This parameter limits the insitu data used to files of the specified data types. Valid data types are: *above_water*, *cast*, *flow_thru*, *pigment*, *lidar*, *mooring*, *drifter*, *sunphoto*, *scan*, and *surface*.

EXCLUDE_INSITU

This parameter is used to exclude specific insitu_ids. This is best used after viewing a plot and determining a file is questionable, matching the point up on the report, excluding the point using this parameter, then replotting.

FILTER_MULTIPLE_INSITU

This parameter is set to yes if more than one insitu point corresponding to the same satellite file and sample,line should not be used. *INSITU_FILTER_METHOD* is used to set how multiple matches are handled. If *INSITU_FILTER_METHOD* is not set, the default value is 0.

INSITU_FILTER_METHOD

This parameter currently has 2 values as described below. (0) The closest insitu match chronologically is used. The rest are flagged and listed in the Error Report. (1) All matching insitu points are averaged together. The plot report will list for insitu_id avg_## where "##" is the corresponding number in the Average Report, printed directly below the plot report.

Remote Sensing Data Selection

The keywords that follow define the remote sensing data that will be used in the matchup.

RS_DB

This parameter contains the SQL data base name that holds the remote sensing data

RS_HOST

This parameter contains the hostname of the SQL data base.

AOIS

This parameter contains the list of area-of-interests to limit the search.

SENSOR

This parameter contains the sensor name to limit the search.

PLATFORM

This parameter contains the sensor platform to limit the search.

FILE_TYPE_RS

This parameter contains the type of level 3 file to use to limit the search. Set to 82 for MODIS Level-3 Files.

in situ Data Processing

The following keywords define the *in situ* data that will be used in the match. Of these, *FIELD*, is required to define the *in situ* parameter that will be matched with the satellite data.

NORMALIZE_PARAMETER

This parameter is set to a constant to normalize the insitu data (i.e. if set to 1.1, this would take all the insitu data and divide each one by 1.1). If not set, no normalization is done.

CONVOLVE

If instead of finding the closest wavelength a convolution is desired, set this parameter to yes and specify the remote sensing response file in the parameter *RESPONSE_FILE*. This will multiply the insitu data at a particular wavelength and the satellite response at that wavelength. This is a discrete convolution over the remote sensing wavelengths.

RESPONSE_FILE

If a convolution of the insitu data has been specified with the CONVOLVE parameter, this parameter must be set to the remote sensing response file that should be used in the convolution.

MODIS_DETECTOR

If this parameter is set, the response function of a particular MODIS detector is used rather than an average response of all the detectors (the default).

FLAGS

This parameter contains the comma-separated list of flags used to exclude a matchup with a remote sensing file. The values must be valid flag names like: LAND, CLDICE, or HIGLINT.

EXCLUDE_RS

This parameter is used to exclude specific rs_ids. This is best used after viewing a plot and determining a file is questionable, matching the point up on the report, excluding the point using this parameter, then replotting.

Remote Sensing Data Selection

The keywords that follow define the remote sensing data that will be used in the matchup.

RS_PRODUCTS

This parameter **MUST** be set and contains a comma-separated list of products to be used in the matchup. Only 1 product/algorithm may be used. Combining a_412_arnone with a_412_carder is **INVALID**. Multiple wavelengths should be specified such as: a_412_arnone,a_443_arnone. This is Valid since absorption (arnone alg.) is the only product used.

DATA_MIN, DATA_MAX

These parameter filters the remote sensing data for a valid data range. This allows the user to ignore outliers or to concentrate the matchup to specific regions of Data Values.

BOX_SIZE

Setting this parameter to a value causes the remote sensing data in an square of side equal to the value of the parameter to be averaged instead of one pixel.

Matchup Control

TIME_TOLERANCE

This parameter **SHOULD** be set by the user. This defines the number of minutes plus or minus the insitu time that remote sensing data is considered valid for that insitu data. i.e. if set to 180 and the insitu data is at 1200, then remote sensing data found between 0900 and 1500 will be considered a match.

Data Plots

The keywords that follow relate to the plots that can be generated by this program.

PLOT

This parameter is set to yes if plots are to be generated.

PLOT_DIR

This parameter is set to the directory where the plots are to be created.

PLOT_FILE_BASE

This parameter is set to the base name of the file. The wavelength will automatically be appended to this name.

PLOT_TITLE

This parameter holds the title of the resulting plots

XTITLE

This parameter holds the title along the X-axis. Normally, this data is the *in situ* data.

YTITLE

This parameter holds the title along the Y-axis. Normally, this data is the *remote sensing* data.

POINT_TYPE

This parameter sets the type of point to be used on the plot. If none is specified, the diamond is used. The point options are

1. Diamond
2. Square
3. Point Number
4. Color-coded diamonds. Each color represents a specific Remote Sensing area of Interest. If this option is chosen, a list will be printed in the report before the plot report listing the color and it's AOI.

XMIN

This parameter holds the desired minimum limit of the plot data for the X-axis.

XMAX

This parameter holds the desired maximum limit of the plot data for the X-axis.

YMIN

This parameter holds the desired minimum limit of the plot data for the Y-axis.

YMAX

This parameter holds the desired maximum limit of the plot data for the Y-axis.

XGRID, YGRID

These parameters set the gridding of the plot. A value of 1 will draw grids at each major tick mark. A value of 2 will draw grids at every other major tick mark. Likewise, for larger numbers. A value of zero will turn off the grid lines.

Other Plots

The following parameters are used to specify other types of plots that can be used in conjunction with data plots. More than one of these may be used in the same parameter file.

STATION_PLOT

This parameter is set to yes if plots of station lat/lon locations are to be generated.

STATION_DIR

This parameter should be set to the directory that the station plots should be located. If not specified and station plots are generated, the current working directory is assumed.

STATION_FILE

This parameter should contain the name of station plots without the extension.

RS_OVERLAYS

This parameter is set to yes if satellite images should be made with points overlaid where the insitu data is located.

OVERLAY_SCRIPT_FILE

This parameter holds the name that the script to generate the overlaid images should be called.

Reports/Exports

INSITU_SUMMARY

This parameter should be set to yes if a summary of the insitu data used is desired which contains a list of experiments, cruise, investigators, and contact information for the insitu data used in the plots.

DATA_FILE

If this parameter is set, a data file containing a text-matrix of data is written to the parameter value useful for importing into analysis programs such as matlab or Microsoft Excel. The format is alternatin Insitu and Remote Sensing data ordered by wavelength in columns, and records going down rows.

Examples

Matchup data in verbose mode using parameter file terra.param

```
$ matchup -v terra.param
```

Matchup data in verbose mode using parameter file terra.param, but redefine the TIME_TOLERANCE and REPORT_LEVEL in the file.

```
$ matchup -v terra.param TIME_TOLERANCE=60 REPORT_LEVEL=0
```

Matchup data in verbose mode using parameter file terra.param, but redefine the REPORT_LEVEL in the file, and add values to the AOI parameter in the file.

```
$ matchup -v terra.param REPORT_LEVEL=0 AOI=+Adriatic,NYBight
```