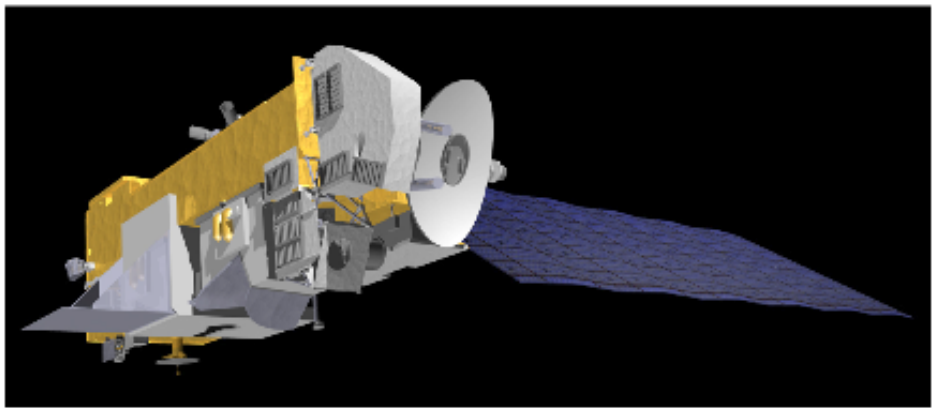
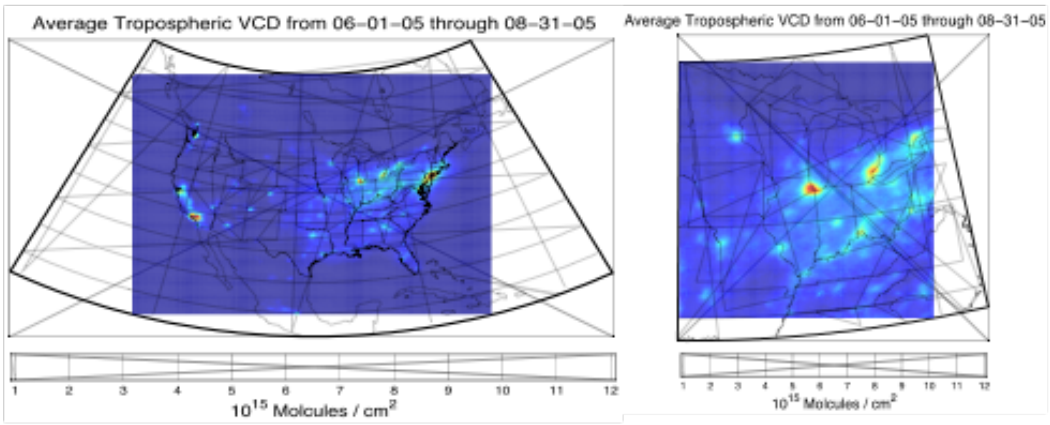
WHIPS

(Wisconsin Horizontal Interpolation Program for Satellites) v3.0.0 User Guide

Prepared for v3.0 Update, April 2019





Tracey Holloway, Jacob Oberman and Peidong Wang

Authors

Principal Investigator: Tracey Holloway

Initial Implementation: Jacob Oberman

Major Updates: Peidong Wang, Erica Scotty, Keith Maki, Xiaomeng Jin

Acknowledgements

We wish to thank the University of Wisconsin-Madison for the use and development of the Wisconsin Horizontal Interpolation Program for Satellites (WHIPS). WHIPS was developed by Tracey Holloway, Jacob Oberman, Peidong Wang and other students and staff, with funding from the NASA Air Quality Applied Science Team (AQAST) and the NASA Health and Air Quality Applied Sciences Team (HAQAST).

Contents:

Installing WHIPS (traditional way):

We prefer users to install WHIPS through conda environment (see next section starts on page 16)

The following are instructions on how to install the WHIPS program on your local Linux/Unix machine. The information here is the same as that in the INSTALL.txt file included with the distribution.

WHIPS depends on the following 3rd party pieces of software:

1. python (version 2.5.x or newer, but NOT version 3)

The software used here was all written for Python version 2.5.x. or later. Though there is a version 3 of Python, it is not backwards compatible and should not be used with WHIPS.

Python may already be installed on your system. Try executing

```
which python
```

to see if this is the case. If you have it, try

```
python -version
```

and if the version number is greater than 2.5 you can skip this step.

If you do have to install python, or simply want a fresh copy, then do the following:

1.1. Obtain the source code from the official python website <http://python.org/download/releases/>.

1.2. Untar the source

```
tar -xzvf python-2.x.y.tar.gz
```

1.3. Set any desired compilation environmental variables. These include all variables commonly recognized by autoconf scripts, including:

CC for the C compiler
CFLAGS for the C compiler flags
CXX for the C++ compiler
CXXFLAGS for the C++ compiler flags
and many others

If you have no idea what any of this means don't worry about it. It is more than likely safe to ignore it.

1.4. Change to the unpacked directory and run the configuration script. The script accepts numerous arguments, the most important among them the --prefix argument that sets the base directory for the install.

```
cd python-2.x.y
./configure --prefix=/path/to/base/directory
```

1.5. Make the program and install it to the base directory

```
make install
```

1.6. Confirm that your default python is now the one you just installed. Run the following commands to regenerate the list of commands as if you'd just logged in, then look at the full path to the default copy of python.

```
rehash
which python
```

This should print out

```
/path/to/base/directory/bin/python
```

If it does not, you need to adjust your path in your shrc file (~/.tcshrc, ~/.cshrc, ~/.bashrc, depending on your shell) to point to the new copy of python. Do NOT attempt to use an alias to point to the correct copy of python as this can cause serious errors with python's installation tools.

2. Install HDF4

The software was originally written with HDF 4.2.6. It should be backwards compatible beyond that but no attempts have been made to determine how far back it goes.

HDF4 software is necessary to read in the MOPITT data.

The following instructions detail how to install from source. Many of the programs listed have binaries, and if those work for your machine it might be a good idea to try those first.

2.1. Download the HDF4 source code from <http://wwww.hdfgroup.org/ftp/HDF/HDF_Current/src/>.

2.2. Untar the source code

```
tar -xzvf hdf-4.2.6.tar.gz
```

2.3. Install JPEG.

The HDF4 installation depends on the JPEG libraries, so we'll need to install those before we can proceed with installing HDF4.

2.3.1. Download the JPEG (v66 or newer) source code. A link is provided at <http://www.hdfgroup.org/release4/obtain.html>.

2.3.2. Untar JPEG

```
tar -xzvf jpegsrc.v66.tar.gz
```

2.3.3. Move to the untarred jpeg directory and run the configure script. Again, make sure to set the --prefix flag to the base directory of the install.

```
cd jpeg-6b
./configure
--prefix=/path/to/base/directory
```

2.3.4. Build the libraries

```
make
```

2.3.5. Install the libraries to the base directory. Note that unlike most well-behaved distributions, this actually has a chance of failing. If it complains about a file called 'cjpeg.1', try explicitly creating the filetree in which it resides, then re-running the install.

```
make install
```

2.4. Install zlib.

The HDF4 installation depends on this library as well. We build it from source, though binaries may work better for your system.

2.4.1. Obtain zlib. Note that though it is linked from the HDF4 site, the link was broken as of the last time I checked. The source code can be obtained directly from <http://zlib.net>.

2.4.2. Untar the zlib source distribution

```
tar -xzvf zlib-1.2.6.tar.gz
```

2.4.3. Change to folder and configure the zlib install. As always, set the base directory using the --prefix flag

```
cd zlib-1.2.6
./configure --prefix=/path/to/base/directory
```

2.4.4. Test the installation if desired

```
make test
```

2.4.5. If the tests pass, install the libraries to the base directory.

```
make install
```

2.5. Set some environmental variables that will be needed during the install. Note that the F77 variable might not be necessary, but CFLAGS does need to include the -fPIC flag

```
setenv F77 gfortran
setenv CFLAGS -fPIC $CFLAGS
```

2.6. Configure HDF for installation. Note that unlike most configurations, this step requires more than just the –prefix flag. Make sure to give the correct path the base directory under which the zlib and jpeg installs (not the source code!) live. The --disable-netcdf is also critical, but may prevent you from using this HDF build for other purposes.

```
cd hdf-4.2.6
./configure --with-zlib=/path/to/base/directory\
            --with-jpeg=/path/to/base/directory \
            --disable-netcdf \
            --prefix=/path/to/base/directory
```

2.7. Build the HDF4 libraries

```
make
```

2.8. Test the installation

```
make test
```

2.9. Install to the base directory

```
make install
```

3. Install HDF5

This is a completely separate install from HDF4. You have to install BOTH even though they are both HDF. Provides read functionality for HDF-EOS files used by OMI instruments, as well as supporting the netCDF installation (step 4)

As per usual, it is probably possible to acquire binaries, but the instructions below are for compilation from source.

3.1. Acquire the HDF5 source code from <http://wwww.hdfgroup.org/ftp/HDF5/current/src>.

3.2. Untar the source code.

```
tar -xzvf hdf5-1.8.8.tar.gz
```

3.3. Set an environment variable to point the configuration script to the desired FORTRAN compilers. The program has a really buggy compiler autodetect "feature" that makes this necessary.

```
setenv FC gfortran
```

3.4. Configure, noting that this is again a case where the flags passed to the configure script are mandatory. Make sure that the --with-zlib flag points to a valid directory that contains the zlib library and another with the include files. You should already have installed zlib under step 2.4 above.

```
cd hdf5-1.8.8
./configure --prefix=/path/to/base/directory \
--enable-fortran \
--with-zlib=/path/to/base/directory/lib,\
        /path/to/base/directory/include
```

3.5. Build HDF5

```
make
```

3.6. Test the build

```
make check
```

3.7. Install the build

```
make install
```

4. Install netCDF

netCDF is necessary to write out to the commonly used netCDF file format. The software has only been tested with a netcdf-4.1.3 installation. Earlier versions of netCDF-4 might work, and there's even a chance netCDF-3 would work, but you're best off with the latest version.

4.1. Download the source from <http://www.unidata.ucar.edu/downloads/netcdf/index.jsp>.

4.2. Decompress the source code

```
untar -xzvf netcdf-4.1.3.tar.gz
```

4.3. Set environmental variables that will tell netCDF where the HDF5 installation lives. We need to include both of these so it can find the libraries and the include files

```
setenv LDFLAGS -L/path/to/base/directory/lib
setenv CPPFLAGS-L/path/to/base/directory/include
```

4.4. Configure the installation using the prefix flag to specify the base directory for the install.

```
cd netcdf-4.1.3
./configure --prefix=/path/to/base/directory
```

4.5. Build the program

```
make
```

4.6. Test the build

```
make check
```

4.7. If the test ran successfully, install.

```
make install
```

5. Install the geos framework

The geos framework (short for geometry engine - open source) is a library containing many common GIS functions. We need it to build shapely, the python module that we use to perform the geometry operations needed in interpolation.

5.1. Obtain the source for the geos framework from <http://trac.osgeo.org/geos>. 5.2. Decompress the installation.

```
bunzip2 geos-3.3.2.tar.bz2
tar -xvf geos-3.3.2.tar
```

5.3. Configure the installation, using the --prefix flag to specify the base directory.

```
cd geos-3.3.2
./configure --prefix=/path/to/base/directory
```

5.4. Build geos

```
make
```

5.5. Install geos to the base directory

```
make install
```

5.6. If the library directory under your base directory is not in a default location (i.e. /usr/local/lib) where the shell automatically looks, you need to add it to the library path. These lines should probably be added to your .tcshrc or .bashrc or .cshrc file (depending on your shell) as whips won't work properly if they are not.

```
setenv LD_LIBRARY_PATH \
/path/to/base/directory/lib:$LD_LIBRARY_PATH
```

6. Install pyhdf

This is, to the best of my knowledge, the only python library that can read HDF4 files (like MOPITT). It should theoretically be installable using the same technique as other python modules, but seems to be more finicky.

6.1. Download the source for pyhdf from <http://sourceforge.net/projects/pysclint>.

6.2. Untar the source

```
tar -xzvf pyhdf-0.8.3.tar.gz
```

6.3. Set environment variables so that the pyhdf installation will know where to find the hdf, zlib, and jpeg libraries. Note that if these are in different libraries, you'll need to set the variables to point to multiple

directories. Note that because these are only needed for the install they won't need to go in the shrc file for your shell.

```
setenv INCLUDE_DIRS /path/to/base/directory
setenv LIBRARY_DIRS /path/to/base/directory
```

6.4. Go into the pyhdf folder and invoke the python setup script (which will both compile and install the script)

```
cd pyhdf-0.8.3
python setup.py install
```

If you run into problems, make sure that you properly set the -fPIC flag when compiling HDF4 (see step 2.7)

7. Install easy_install

easy_install is a widely-implemented framework for the installation of python modules. It performs many of the features of a traditional package manager like yum or rpm, but most critically does NOT uninstall packages and does NOT chase down dependencies. We're going to use it extensively from here on out to install the python modules needed by whips.

7.1. Obtain the "egg" (distributed package) of easy_install from the maintainers at <http://pypi.python.org/pypi/setuptools#downloads>. Make sure to select the egg that matches your installation of python.

7.2. Run the egg as a shell script and easy_install should install itself under the binary directory associated with your python installation.

```
sh setuptools-0.6c11-py2.7.egg
```

8. Install netCDF4-python

This particular python module requires a different installation approach than the others listed below, so we take care of it first. It's used for writing results to the popular netCDF file type.

8.1. Download the source code for netcdf4-python
from .google.com/p/netcdf4- python/downloads/list

8.2. Unpack the source distribution

```
tar -xzvf netcdf4-0.9.9.tar.gz
```

8.3. If you haven't done so already, set the environmental variable HDF5_DIR to point to the base directory for the HDF5 install. In addition, set NETCDF4_DIR to point to the base directory for the netCDF install.

```
setenv HDF5_DIR /path/to/base/directory/
setenv NETCDF4_DIR /path/to/base/directory/
```

8.4. Build the netCDF4-python library

```
cd netCDF4-0.9.9
python setup.py build
```

8.5. Assuming the build didn't throw any errors, install the library.

```
python setup.py install
```

8.6. If you want to, test the installation. Note that in some circumstances the library will work fine even if the tests refuse to run entirely.

```
cd test
python run_all.py
```

9. Install python modules available through easy_install

These python modules are all used by WHIPS to perform various operations. They will be installed to whichever copy of python the operating system finds first, which makes it all the more important that you double check that this is the version you want (see step 1.6).

9.1. Install numpy, which is a basic array operation library.

```
easy_install numpy
```

9.2. Install shapely, a geometry library used by the interpolation routines in whips

```
easy_install shapely
```

9.3. Install pyproj, which is a port of the popular PROJ4 fortran library. It provides the optimized projection functions used by WHIPS.

```
easy_install PyProj
```

9.4. Install numexpr, which is required for pytables

```
easy_install numexpr
```

9.5. Install cython, which is also required for pytables

```
easy_install Cython
```

9.6. Install pytables, a program allowing python read/write access to HDF5 files.

9.6.1. Tell pytables where the HDF5 installation is. Give it the base directory (as opposed to the library directory) via the environment variable HDF5_DIR. This environment variable is only needed for the installation.

```
setenv HDF5_DIR /path/to/base/directory
```

9.6.2. Install the actual pytables module itself (note that it isn't called pytables!)

```
easy_install tables
```

9.6.3. Add the lib directory where you installed tables to your library path. This should go in your .tcshrc or .bashrc or .cshrc file (depending on your shell) for whips to function properly.

```
setenv LD_LIBRARY_PATH \
```

```
/path/to/base/directory/lib:$LD_LIBRARY-PATH
```

10. Install WHIPS

WHIPS is available through pip install (a different package manager).

10.1. Install WHIPS using the pip install command

```
pip install WHIPS3
```

10.2. The whips.py executable will be installed in

```
/path/to/base/directory/bin/
```

If that folder isn't on your path, you'll need to make it accessible from the command line. The easiest means is probably to add it to the path in your remote config file (.tcshrc or .bashrc or .cshrc or ...). This can be accomplished from .tcshrc by adding the following line.

```
setenv PATH $PATH:/path/to/base/directory/bin/
```

Note that the first time you do this, you'll need to run

```
source ~/.tcshrc
```

for it to take effect.

Installing WHIPS (through conda environment):

The following are instructions on how to install the WHIPS program on your local Linux/Unix machine through conda environment.

1. Install Miniconda with Python 2.7
   a. Check if you already have Anaconda or Miniconda installed by running:

      ```
      conda
      ```

      If your computer recognizes this command, congratulations! Skip to step 2.
   b. If not, instructions for installing Miniconda can be found here: https://docs.conda.io/en/latest/miniconda.html
   c. Click on your operating system and download the bash installer.
   d. Miniconda is recommended (Anaconda and Miniconda are the same, but Miniconda is smaller. Anaconda comes with a lot of preinstalled packages and is a HUGE file)
   e. Select an installer. **You must choose Python 2.7.** WHIPS is not compatible with Python 3.X.

2. Make a new Conda environment.

   ```
   conda create –name whipsenv python=2.7
   ```

   Where “whipsenv” is the name of your environment. You can type whatever name you want for your environment in place of “whipsenv.” Again, make sure that you set the Python version to 2.7.

3. Enter your conda environment

   ```
   source activate whipsenv
   ```

Make sure that you ONLY install packages AFTER you have entered your environment.

4. Install the packages WHIPS needs to run

```
conda install cython
conda install shapely
conda install pyproj
conda install netCDF4
conda install pytables=2
conda install jpeg=8
```

Accept all dependencies installed by each package, and downgrades forced by Pytables.

Note: it's very important that you type pytables=2 and jpeg=8. WHIPS is not compatible with later versions of pytables and jpeg.

5. Install pyhdf
Unfortunately, you can't use conda to install Pyhdf directly. You can either install it through a third-party provider from:

```
conda install -c rshekhar pyhdf
```

Or you can download the package online, and then install it using pip install (a different package manager).

a. Make sure pip is up to date (it should have been installed already):

```
conda update pip
```

b. Download pyhdf from http://sourceforge.net/projects/pysclint
c. Untar the file:

```
tar -xzf /path/to/pyhdf_file
```

d. Install pyhdf

```
pip install /path/to/pyhdf_file
```

Replace /path/to/pyhdf_file with the path to the folder your just downloaded.

6. Install WHIPS

```
pip install WHIPS3
```

7. Run WHIPS
You need to activate your conda environment in order to run WHIPS:

```
source activate whipsenv
whips.py
```

Step-by-step Guide: Command line interface

There are two primary methods of running whips. The first method is to give whips all the information and settings it needs as flags in a single command line call. These calls become very long; typing them in by hand is therefore inadvisable. The main benefit to this approach is that whips can easily be batch run by building, executing, and modifying the call in a script.

This section will walk through how to use WHIPS from the command line step by step, outlining each decision that must be made.

Steps from 2 onward list the flags associated with that step. Put them all together to get a complete command.

Step 0 - Download/installation

• WHIPS can be found at <http://pypi.python.org/pypi/WHIPS/>.

• Installation instructions included as INSTALL.txt or can be referenced in the previous section of this document.

• Install depends on:

  – HDF4

  – HDF5

  – netCDF 4

  – Python 2.3 or later

  – Other dependencies

Step 1 – Help

• The best resource if you get stuck at any point is the README.txt document included with the program.

• WHIPS also has a built-in help feature that prints usage and quits. It can be accessed by the command

```
whips.py --help
```

• If you need help with individual functions, WHIPS has help tailored to those functions. It can be accessed with the command.

```
whips.py --AttributeHelp funcName [funcName ...]
```

Step 2 – Choosing input files

• Select the directory that you would like to search for input files

```
--directory /path/to/directory
```

• If no directory is selected, the current working directory is used.

• Also put in a list of the specific filenames to process

```
--fileList fileName1 [fileName2 ...]
```

• If the --fileList flag is not passed, all files in the directory will be processed.

• The file type must be specified so that the program will know how to interface with the files. This is specified as

```
--filetype yourType
```

• There are four currently supported generic filetypes, and a handful of specialized filetypes that make WHIPS easier to use with common versions of the products. It is suggested that if the input files used match one of the specialized filetypes, that filetype should be selected (rather

than the appropriate generic filetype). The currently supported filetypes are as follows:

– *HDFknmiomil2_generic*: KNMI's Level 2 OMI retrievals
– *HDFmopittl2_generic*: NASA's Level 2 MOPITT CO retrieval
– HDFnasaomil2_generic: NASA's Level 2 OMI retrievals
– HDFmodisl2_generic: NASA's Level 2 MODIS AOD retrieval

– *MOPITT_CO_NASA_HDF_V5*: NASA Level 2 MOPITT CO (Version 5)
– *OMI_NO2_KNMI_HDF_v2_0_preFeb2006*: KNMI's Level 2 OMI retrievals (Version 2.0 pre- February 2006)
– *OMI_NO2_KNMI_HDF_v2_0_postFeb2006*: KNMI's Level 2 OMI retrievals (Version 2.0 post- February 2006)
– *OMI_NO2_NASA_HDF_v1_2*: NASA's Level 2 OMI retrievals (Version 1.2)
– (New to 1.3.0) *OMI_NO2_NASA_HDF_v3*: NASA's Level 2 OMI NO_2 retrievals (Version 3)
– (New to 1.3.0) *OMI_HCHO_NASA_HDF_v3:* NASA's Level 2 OMI HCHO retrievals (Version 3)
– (New to 3.0.0) *OMI_SO2_NASA_HDF_v3:* NASA's Level 2 OMI SO2 retrievals (Version 3)
– *MODIS_AOD_NASA_Collection_5*: NASA's Level 2 MODIS AOD retrievals (Collection 5)

Step 3 - Choosing and Defining a Grid

• Choose projection `--gridProj yourProj`

• Available projections are as follows:

- *lcc2par*: Lambert conic conformal projection

- *latlon*: Equidistant cylindrical projection (better known as the unprojected or lat/lon projection)

• Provide projection and grid parameters

```
--projAttrs AttName1:AttValue1
            AttName2:AttValue2 …
```

• Lists of required attributes and suggested values are available using --AttributeHelp as described above

• Projection and grid parameters should exactly match those used in your model for the best satellite-model comparison.

– For CMAQ/SMOKE, the values are provided to the model in the GRIDDESC file.

Step 4 – Choosing an Interpolation Function

• The interpolation function should be selected as

```
--mapFunc yourMapFunc
```

• Available functions:

– *point_in_cell*: Maps each pixel to one and only one grid cell. Pixels are assigned to the grid cell in which their center lat/lon lies. This is effectively "nearest neighbor" interpolation.

– *regional_intersect*: Maps each pixel to all grid cells that it intersects. Pixel data can be used by multiple gridcells. Area of overlap is not considered.

– *global_intersect*: Maps each pixel to all grid cells that it intersects. Pixel data can be used by multiple grid cells. Area of overlap is not considered.

• The use of *regional_intersect* is encouraged when processing OMI data to improve data quality and resolution. MOPITT and MODIS retrievals do not provide pixel size information and cannot be used with *regional_intersect*.

• The *global_intersect* interpolation function is designed to properly handle grids where the east and west edges lie on the same meridian, where the *regional_intersect* interpolation function does not. If a regional grid projection is being used, *regional_intersect* should be chosen; if a global projection where the east and west edges lie on the same meridian is being used, *global_intersect* should be chosen instead. Once again, because pixel size information is not provided for MOPITT and MODIS retrievals, *global_intersect* cannot be used for those filetypes.

Step 5 – OPTIONAL - Choosing an Output Function

• WHIPS will automatically select the most appropriate output function for you. You can override it's selection with the following parameter:

```
--outFunc yourFunc
```

• Available functions

– *OMNO2e_netCDF_avg***:** Averages input data according to the algorithm used by NASA to grid OMI NO2 Level 2 to Level 3. Outputs to a netCDF file.

– *OMNO2e_netCDF_avg_v3*: Averages input data according to the algorithm used by NASA to grid OMI NO2 Level 2 (version 3) to Level 3. Pixels affected by row anomaly are discarded in this function. Outputs to a netCDF file.

– *OMIHCHO_netCDF_avg*: Averages input data using the same algorithm as *OMNO2e_netCDF_avg_v3.* Pixels affected by row anomaly are discarded. Outputs to a netCDF file.

– *OMISO2_netCDF_avg*: Averages input data using the same algorithm as *OMNO2e_netCDF_avg_v3.* Pixels affected by row anomaly are discarded. Outputs to a netCDF file.

– *unweighted_filtered_MOPITT_avg_netCDF*: Averages a single input file according to the algorithm used by NASA to grid MOPITT Level 2 to Level 3. Outputs to a netCDF file.

• Each output function has additional parameters that must be specified as

```
--outFuncAttrs AttName1:AttValue1
AttName2:AttValue2 ...
```

• Refer to README.txt or --AttributeHelp for description of the required parameters.

Step 6 – Choosing an Output Directory

• The directory to which to write output files must be specified as

```
--outDirectory /path/to/directory
```

• Optionally, the name of the output file can be selected

```
--outFileName FileName
```

• If no name is provided, the output file will be named “output1”

Step 7 – Additional Options

• Optionally, the grid coordinates (lat/lon) can be output to a separate file. It is good practice to output the grid coordinates once for each individual grid you use. Provide the absolute path to the file where you want this output.

```
--includeGrid GridFileName
```

• Command line output while running can be silenced by setting verbosity to False. Default is True, and WHIPS will print status information as it runs.

```
--verbose {True, False}
```

• When the program tries to read a file that is formatted incorrectly, behavior is governed by the interactive flag. Setting the flag to True will cause the program to pause and wait for user input. Setting the flag to False (the default) will cause the program to skip invalid files and try the next file.

```
    --interactive {True, False}
```

Step 8 – Running the Program

• Currently, there is no way to output multiple timesteps with a single call to WHIPS. Thus, the program must be invoked for each desired output time period.

• For comparison to models, it is highly recommended to process data with a minimum temporal resolution of one day.

• The program is amenable to shell scripting.

Extra - Comparison to models

• The averaging kernel for the data should be applied to the model to generate a "pseudo-satellite" model column.

• The model should only be sampled on days when the satellite retrieval was valid.

• The model should be sampled at satellite overpass time.

Finally – What does it look like when you put it together?

Line continuation characters (IE '\') are not necessary, they just allow the "pretty" formatting of the command. Note that the indentation here is meant for visual clarity and should not be included in the command.

```
whips.py \
    --directory /where/you/keep/the/files/ \
    --filetype HDFknmiomil2 \
```

```
--fileList OMI-Aura_L2-OMDOMINO_2006m0701t0023\
     -o10423_v003-2010m1008t224420.he5 \
     OMI-Aura_L2-OMDOMINO_2006m0701t0112\
     -o10424_v003-2010m1008t224845.he5 \
--gridProj lcc2par \
--projAttrs xOrig:-2916000 yCell:36000\
     refLon:-97 refLat:40 \
     nCols:162 nRows:126 stdPar2:45 \
     stdPar1:33 xCell:36000 \
     earthRadius:6370000 yOrig:-2268000\
--mapFunc regional_intersect \
--outFuncAttrs \
    overallQualFlag:TroposphericColumnFlag \
    cloudFrac:CloudFraction \
    solarZenithAngle:SolarZenithAngletime:Time\
    longitude:Longitude \
    inFieldNames:Time,AveragingKernel,\
    TroposphericVerticalColumn \
    outFieldNames:time,avKern,tropVCD \
    "outUnits:TAI93,unitless x \
    1000,molec/cm^2x1^-15" \
    extraDimLabel:none,Layers,none \
    extraDimSize:0,34,0 \
    timeStart:00:00:00_07-01-2006 \
    timeStop:23:59:59_07-01-2006 \
    timeComparison:UTC fillVal:-9999\
    cloudFractUpperCutoff:0.3 \
    solarZenAngUpperCutoff:85 \
    pixIndXtrackAxis:1 \
 --outDirectory /where/you/want/output \
 --outFileName OMI_DOMINO_20060701_test.nc \
 --includeGrid \
          /some/path/OMI_DOMINO_GridFileName.nc\
 --verbose True \
 --interactive True
```

Step-by-step Guide: Text input file

The second method of running WHIPS is to specify all the options and information it needs in a specially-formatted input file. The command line call then consists only of specifying the location of the input file WHIPS should read from.

Below is an explanatory example of the format

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
FAKE INPUT FILE
BEGIN

. This line begins with a period. That designates it as a comment
. Comments will be ignored by WHIPS when processing the file
. All required flags must be specified with the flag name . and the desired value. These flags must be in capital case. . All possible flags (including optional flags) are shown below:

DIRECTORY = /where/you/had/your/input/files
FILETYPE = some_file_type
GRIDPROJ = some_grid_projection
MAPFUNC = some_map_function
OUTDIRECTORY = /where/you/want/output/files
OUTFILENAME = name_of_output_file
VERBOSE = True_or_False
INTERACTIVE = True_or_False
OUTFUNC = some_output_function
INCLUDEGRID = /absolute/path/to/output/file/for/grid


. FILELIST is delimited by spaces. Leave out to use all files . In DIRECTORY FILELIST = list of files

. Projection and output function attributes are specified in . the same manner as flags. Just as with the command line call, . the correct attributes must be present or the program will exit
```

```
stdPar1 = 33
inFieldNames = ColumnAmountNO2Trop,Time

. Note that unlike with the command line call, quotations
are
. not needed when whitespace is part of an attribute
value
outUnits = Molecules cm^-2,seconds since epoch
. You can specify parameters in any order.  This
includes
. Intermixing attributes and flags
time = Time
OUTFILENAME = whatever_we_want
. WHIPS input files must begin with BEGIN and end with
. END

END
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Choose each parameter value in the same manner described in the step-by-step guide for command line calls in the previous section of this guide. Once the input file has been completed, it can be used to run Whips, and is easily modified for future runs.

Finally – How do I call whips using the file? Supposing your input file is located at /whips/input.txt, whips can be run using the following command line call:

```
whips.py –inFromFile /whips/input.txt
```

Example workflow

. 1) Install the program and run the built-in test module to confirm that it is working properly. Installation instructions can be found in the file INSTALL.txt

. 2) Download whatever data you plan to process. Currently, the program is designed to process OMI NO2 DOMINO level 2 data, OMI NO2 NASA level 2 data, OMI HCHO NASA level 2 data, MODIS NASA AOD level 2 and MOPITT CO data. See the detailed documentation for the --fileList argument for locations of data.

. 3) Navigate to the folder where whips.py installed or add it to your path. It should be the /bin folder corresponding to the /lib folder where your python packages are installed. Invoke it as:

```
whips.py --help
```

. 4) Follow the on-screen instructions, adding each of the required parameters. If you need help with the projection attributes or the output function attributes, invoke the built-in help as:

```
whips.py --AttributeHelp <function_name>
```

For detailed explanations of all parameters and attributes, see the "Parameter Details" section below.

. 5) Invoke whips.py once for each output file you'd like to create. Note that the software creates output files with only a single timestep, so you'll need to invoke the command once for each timestep (e.g. if you want a month with timesteps every day, you'll probably want to write a shell script that calls the command once for each day)

. 6) Additionally, you may create a grid file for your chosen grid by including the --includeGrid flag followed by the path to the desired filename. A file containing the gridcells used by the projection will be written to this location.

. 7) Concatenate your outputs if desired (the authors recommend the NCO operators at http://nco.sourceforge.net/ if you're using a netCDFoutput format) and carry on!

Example commands

For clarity and readability, line continuation characters are used to place each attribute on a separate line. It is not required to break up attributes like this, but the command line needs to see the invocation as a single command, so if you want to break it onto multiple lines, you must use line-continuation characters.

1. Process MOPITT level 2 CO data, (Version 5)

- processes a single file
- Uses a 36km lambert conic conformal grid centered over North America
- Writes out a 2D, 3D, and 4D parameter from the file `whips.py`

```
--directory /where/you/have/input/files \
        --fileList
MOP02T-20050101-L2V10.1.1.prov.hdf \
        --filetype MOPITT_CO_NASA_HDF_V5 \
        --gridProj lcc2par \
        --mapFunc point_in_cell \
        --outDirectory where/you/want/output \
        --outFileName descriptive_name.nc \
        --verbose True \
        --interactive True \
        --projAttrs xOrig:-2916000 yCell:36000\
        refLon:-97 refLat:40 nCols:162 nRows:126
stdPar2:45 \
        stdPar1:33 xCell:36000 earthRadius:6370000\
        yOrig:-2268000"inFieldNames:Time,Retrieved\
CO Mixing Ratio Profile,Retrieved CO Surface Mixing
Ratio" \
        outFieldNames:time,COprof,COsurf \
        logNormal:False,True,True \
        timeStart:00:00:00_01-01-2005 \
        timeStop:23:59:59_01-01-2005 \
        timeComparison:UTC fillVal:-9999.0\
        solZenAngCutoff:85 dayTime:True
```

2. Process OMI level 2 DOMINO NO2 data, (Version 2)

• Uses a 36km lambert conic conformal grid centered over North America
• explicitly specifies 3 input files
• includes an output grid file (this would be possible in any example, but is only shown in this one)
• Writes out a 2D, and 3D parameter from the file `whips.py \`

```
    --directory /where/you/have/input/files \
    --filetype OMI_NO2_KNMI_HDF_v2_0_postFeb2006 \
    --fileList \
OMI-Aura_L2-OMDOMINO_2011m0901t1502-o37926_v003-2011m1012t1
21342.he5 \
OMI-Aura_L2-OMDOMINO_2011m0901t1641-o37927_v003-2011m1012t1
21458.he5 \
OMI-Aura_L2-OMDOMINO_2011m0901t1820-o37928_v003-2011m1012t1
21613.he5 \
    --gridProj lcc2par \
    --mapFunc regional_intersect \
    --outDirectory /where/you/want/output \
    --outFileName some_descriptive_name.nc \
    --verbose True \
    --interactive False \
    --includeGrid /where/you/want/gridfile/output \
    --projAttrs xOrig:-48 yCell:36 refLon:-97 \
refLat:40 nCols:30 nRows:32 stdPar2:45 stdPar1:33 \
xCell:36 earthRadius:6370 yOrig:-552 \
inFieldNames:Time,AveragingKernel,TroposphericVert\
icalColumn\
outFieldNames:time,avKern,tropVCD \
timeStart:00:00:00_09-01-2011 \
timeStop:23:59:59_09-01-2011 timeComparison:UTC\
fillVal:-9999 cloudFractUpperCutoff:0.3 \
solarZenAngUpperCutoff:85 includePixelCount:False
```

3. Process OMI level 2 NASA NO2 data, (Version 3)

- Uses a 36km lambert conic conformal grid
- writes out two fields to an ouput file
- considers two input files
- looks at all the cornerfiles in the directory

```
 whips.py \

        --directory /where/you/have/input/files \
        --fileList \
OMI-Aura_L2-OMNO2_2011m0430t1440-o36120_v003-2011m0501t043\
317.he5 \
OMI-Aura_L2-OMNO2_2011m0430t1619-o36121_v003-2011m0501t061\
955.he5 \
        --filetype OMI_NO2_NASA_HDF_v3 \
        --gridProj lcc2par \
        --mapFunc regional_intersect \
        --outDirectory /where/you/want/output \
        --outFileName some_file_name.nc \
        --verbose True \
        --interactive False \
        --projAttrs \

        cornerDir:/directery/where/you/keep/cornerf\
        iles \
          cornerFileList: \
          stdPar1:33 stdPar2:45 refLat:40\
          refLon:-97 xOrig:-2916000 \
          yOrig:-2268000 xCell:36000 yCell:36000 \
          nRows:126 nCols:162 earthRadius:6370000\
          inFieldNames:ColumnAmountNO2Trop,Time \
          outFieldNames:tropVCD,time \
               timeComparison:local \
         timeStart:00:00:00_04-30-2011 \
         timeStop:23:59:59_04-30-2011 \
         cloudFractUpperCutoff:.3 \
         solarZenAngUpperCutoff:85 fillVal:-9999.0\
         includePixelCount:False
```

4. Process OMI level 2 NASA HCHO data, (Version 3)

- Uses a 36km lambert conic conformal grid
- writes out two fields to an ouput file
- considers two input files
- looks at all the cornerfiles in the directory

```
whips.py \

       --directory /where/you/have/input/files \
       --fileList \
OMI-Aura_L2-OMHCHO_2010m0101t0005-o29063_v003-2014m0626t16\
5731.he5 \
OMI-Aura_L2-OMHCHO_2010m0101t0144-o29064_v003-2014m0626t16\
5656.he5 \
       --filetype OMI_HCHO_NASA_HDF_v3 \
       --gridProj lcc2par \
       --mapFunc regional_intersect \
       --outDirectory /where/you/want/output \
       --outFileName some_file_name.nc \
       --verbose True \
       --interactive False \
       --projAttrs \

       cornerDir:/directery/where/you/keep/cornerf\
       iles \
        cornerFileList: \
        stdPar1:33 stdPar2:45 refLat:40\
        refLon:-97 xOrig:-2916000 \
        yOrig:-2268000 xCell:36000 yCell:36000 \
        nRows:126 nCols:162 earthRadius:6370000\
        inFieldNames: ColumnAmount,Time \
        outFieldNames:HCHOColumn,time \
            timeComparison:local \
       timeStart:00:00:00_04-30-2011 \
       timeStop:23:59:59_04-30-2011 \
       cloudFractUpperCutoff:.3 \
       solarZenAngUpperCutoff:85 fillVal:-9999.0\
       includePixelCount:False
```

5. Process OMI level 2 NASA SO2 data, (Version 3)

- Uses a 36km lambert conic conformal grid
- writes out two fields to an ouput file
- considers two input files
- looks at all the cornerfiles in the directory

```
 whips.py \

       --directory /where/you/have/input/files \
       --fileList \
OMI-Aura_L2-OMSO2_2010m0101t0005-o29063_v003-2014m0626t16\
5731.he5 \
OMI-Aura_L2-OMSO2_2010m0101t0144-o29064_v003-2014m0626t16\
5656.he5 \
       --filetype OMI_SO2_NASA_HDF_v3 \
       --gridProj lcc2par \
       --mapFunc regional_intersect \
       --outDirectory /where/you/want/output \
       --outFileName some_file_name.nc \
       --verbose True \
       --interactive False \
       --projAttrs \

       cornerDir:/directery/where/you/keep/cornerf\
       iles \
        cornerFileList: \
        stdPar1:33 stdPar2:45 refLat:40\
        refLon:-97 xOrig:-2916000 \
        yOrig:-2268000 xCell:36000 yCell:36000 \
        nRows:126 nCols:162 earthRadius:6370000\
        inFieldNames: ColumnAmountSO2_PBL,Time \
        outFieldNames:SO2PBLColumn,time \
             timeComparison:local \
       timeStart:00:00:00_04-30-2011 \
       timeStop:23:59:59_04-30-2011 \
       cloudFractUpperCutoff:1 \
       solarZenAngUpperCutoff:85 fillVal:-9999.0\
       includePixelCount:False
```

6. Process MODIS level 2 NASA AOD data, (Version 1)

- Uses a 36km lambert conic conformal grid centered over the US
- writes out two fields to an ouput file
- considers all input files in the directory
- includes the number of level 2 pixels averaged for each level 3 pixel

```
whips.py \

        --directory /where/you/have/input/files \
        --filetype MODIS_AOD_NASA_Collection_5 \
        --gridProj lcc2par \
        --mapFunc point_in_cell \
        --outDirectory /where/you/want/output \
        --outFileName some_descriptive_name.nc \
        --verbose True \
        --interactive True \
        --projAttrs \
          xOrig:-2916000 yOrig:-2268000 \
          xCell:36000 yCell:36000 \
          nRows:126 nCols:162 \
          refLat:40 refLon:-97 stdPar1:33 stdPar2:45\
          earthRadius:6370000 \
          --outFuncAttrs \
          timeStart:00:00:00_07-03-2008 \
          timeStop:23:59:59_07-03-2008 \
          timeComparison:UTC \
          fillVal -9999.0\
         inFieldNames:Corrected_Optical_Depth_Land,\
         Effective_Optical_Depth_Average_Ocean \
         outFieldNames:AOD_Land,AOD_Ocean \
         includePixelCount:True
```

Example Commands – Text input files

The three text input file examples below are equivalent to their counterparts in the section above.

1. Process MOPITT level 2 CO data, (Version 5)

```
-----------------------------------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
BEGIN
DIRECTORY = /where/you/have/input/files
FILELIST =  MOP02T-20050101-L2V10.1.1.prov.hdf
FILETYPE = MOPITT_CO_NASA_HDF_V5
GRIDPROJ = lcc2par
MAPFUNC =  point_in_cell
VERBOSE = True
INTERACTIVE = True
. Projection Attributes
xOrig = -2916000
yCell = 36000
refLon = -97
refLat = 40
nCols = 162
nRows = 126
stdPar2 = 45
stdPar1 = 33
xCell = 36000
earthRadius = 6370000
yOrig = -2268000
. output function attributes
inFieldNames = Time,Retrieved CO MixingRatio
Profile,Retrieved
CO Surface Mixing Ratio
outFieldNames = time,COprof,COsurf
logNormal = False,True,True
timeStart = 00:00:00_01-01-2005
timeStop = 23:59:59_01-01-2005
timeComparison = UTC
fillVal = -9999.0
solZenAngCutoff = 85
dayTime = True
```

```
OUTDIRECTORY =  /where/you/want/output
OUTFILENAME = descriptive_name.nc
END

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
whips.py --inFromFile nameOfAboveFile.txt
```

2. Process OMI level 2 DOMINO NO2 data, (Version 2)

~~~~~~
BEGIN
DIRECTORY = /where/you/have/input/files
FILETYPE = OMI_NO2_KNMI_HDF_v2_0_postFeb2006
FILELIST =
OMI-Aura_L2-OMDOMINO_2011m0901t1502-o37926_v003-
2011m1012t121342.he5
OMI-Aura_L2-OMDOMINO_2011m0901t1641-
o37927_v003-2011m1012t121458.he5 OMI-Aura_L2-
OMDOMINO_2011m0901t1820-o37928_v003-2011m1012t12161
3.he5
GRIDPROJ = lcc2par
MAPFUNC = regional_intersect
VERBOSE = True
INTERACTIVE = False
OUTDIRECTORY = /where/you/want/output
OUTFILENAME = some_descriptive_name.nc
INCLUDEGRID =
/where/you/want/output/grid/full/path.nc
. Proj attrs
xOrig = -48
yCell = 36
refLon = -97
refLat = 40
nCols = 30
nRows = 32
stdPar2 = 45
stdPar1 = 33
xCell = 36
earthRadius = 6370
yOrig = -552
. Output attrs
inFieldNames =
Time,AveragingKernel,TroposphericVerticalColumn
outFieldNames = time,avKern,tropVCD
timeStart = 00:00:00_09-01-2011
timeStop = 23:59:59_09-01-2011
timeComparison = UTC
fillVal = -9999
~~~~~~

```
cloudFractUpperCutoff = 0.3
solarZenAngUpperCutoff = 85
includePixelCount = False
END
```

~~~~~~

```
whips.py --inFromFile nameOfAboveFile.txt
```
~~~~~~

3. Process OMI level 2 NASA NO2 data, (Version 3)

~~~~~~
BEGIN
. Flags
DIRECTORY = /where/you/have/input/files
FILETYPE = OMI_NO2_NASA_HDF_v3
FILELIST =
OMI-Aura_L2-OMNO2_2011m0430t1440-o36120_v003-
2011m0501t043317.he5
OMI-Aura_L2-OMNO2_2011m0430t1619-
o36121_v003-2011m0501t061955.he5
GRIDPROJ = lcc2par
MAPFUNC = regional_intersect
OUTDIRECTORY = /where/you/want/output
OUTFILENAME = some_file_name.nc
VERBOSE = True
INTERACTIVE = False
. Parser attributes
cornerDir =/directory/where/you/keep/cornerfiles
cornerFileList =
. Grid attributes
stdPar1 = 33
stdPar2 = 45
refLat = 40
refLon = -97
xOrig = -2916000
yOrig = -2268000
xCell = 36000
yCell = 36000
nRows = 126
nCols = 162
earthRadius = 6370000
. Output function attributes
inFieldNames = ColumnAmountNO2Trop,Time
outFieldNames = tropVCD,time
timeComparison = local
timeStart = 00:00:00_04-30-2011
timeStop = 23:59:59_04-30-2011
cloudFractUpperCutoff = .3
solarZenAngUpperCutoff = 85
~~~~~~

```
fillVal = -9999.0
includePixelCount = False
END
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
whips.py --inFromFile nameOfAboveFile.txt
```

4. Process OMI level 2 NASA HCHO data, (Version 3)

~~~~~~
```
BEGIN
. Flags
DIRECTORY = /where/you/have/input/files
FILETYPE = OMI_HCHO_NASA_HDF_v3
FILELIST =
OMI-Aura_L2-OMHCHO_2010m0101t0005-o29063_v003-2014m
0626t165731.he5
OMI-Aura_L2-OMHCHO_2010m0101t0144-o29064_v003-2014m
0626t165656.he5
GRIDPROJ = lcc2par
MAPFUNC = regional_intersect
OUTDIRECTORY = /where/you/want/output
OUTFILENAME = some_file_name.nc
VERBOSE = True
INTERACTIVE = False
. Parser attributes
cornerDir =/directory/where/you/keep/cornerfiles
cornerFileList =
. Grid attributes
stdPar1 = 33
stdPar2 = 45
refLat = 40
refLon = -97
xOrig = -2916000
yOrig = -2268000
xCell = 36000
yCell = 36000
nRows = 126
nCols = 162
earthRadius = 6370000
. Output function attributes
inFieldNames = ColumnAmount,Time
outFieldNames = HCHOColumn,time
timeComparison = local
timeStart = 00:00:00_04-30-2011
timeStop = 23:59:59_04-30-2011
cloudFractUpperCutoff = .3
solarZenAngUpperCutoff = 85
```
~~~~~~

```
fillVal = -9999.0
includePixelCount = False
END
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
whips.py --inFromFilenameOfAboveFile.txt
```

5. Process OMI level 2 NASA SO2 data, (Version 3)

~~~~~~
```
BEGIN
. Flags
DIRECTORY = /where/you/have/input/files
FILETYPE = OMI_SO2_NASA_HDF_v3
FILELIST =
OMI-Aura_L2-OMSO2_2010m0101t0005-o29063_v003-2014m
0626t165731.he5
OMI-Aura_L2-OMSO2_2010m0101t0144-o29064_v003-2014m
0626t165656.he5
GRIDPROJ = lcc2par
MAPFUNC = regional_intersect
OUTDIRECTORY = /where/you/want/output
OUTFILENAME = some_file_name.nc
VERBOSE = True
INTERACTIVE = False
. Parser attributes
cornerDir =/directory/where/you/keep/cornerfiles
cornerFileList =
. Grid attributes
stdPar1 = 33
stdPar2 = 45
refLat = 40
refLon = -97
xOrig = -2916000
yOrig = -2268000
xCell = 36000
yCell = 36000
nRows = 126
nCols = 162
earthRadius = 6370000
. Output function attributes
inFieldNames = ColumnAmountSO2PBL,Time
outFieldNames = SO2PBLColumn,time
timeComparison = local
timeStart = 00:00:00_04-30-2011
timeStop = 23:59:59_04-30-2011
cloudFractUpperCutoff = 1
solarZenAngUpperCutoff = 85
```
~~~~~~

```
fillVal = -9999.0
includePixelCount = False
END
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
whips.py --inFromFile nameOfAboveFile.txt
```

6. Process MODIS level 2 NASA AOD data, (Version 1)

~~~~~~
```
BEGIN
. Flags
DIRECTORY = /where/you/have/inputfiles
FILETYPE = MODIS_AOD_NASA_Collection_5
GRIDPROJ = lcc2par
MAPFUNC = point_in_cell
VERBOSE = True
INTERACTIVE = True
OUTDIRECTORY = /where/you/want/output
OUTFILENAME = some_file_name.nc
. Projection attributes
stdPar1 = 33
stdPar2 = 45
refLat = 40
refLon = -97
xOrig = -2916000
yOrig = -2268000
xCell = 36000
yCell = 36000
nRows = 126
nCols = 162
earthRadius = 6370000
. Output function attributes
inFieldNames =
Corrected_Optical_Depth_Land,Effective_Optical_Dept
h_Average_Ocea
n
outFieldNames = AOD_Land,AOD_Ocean
timeComparison = UTC
timeStart = 00:00:00_07-03-2008
timeStop = 23:59:59_07-03-2008
fillVal = -9999.0
includePixelCount = True
END
```
~~~~~~
```
whips.py --inFromFile nameOfAboveFile.txt
```

Parameter Details

Each input attribute is explained here. Note that each output function has a separate set of required inputs and that the --outFuncAttrs is therefore broken down by output function. Make sure you are referencing the details for the attributes relevant to the function you want to use.

`--help`

REQUIRED: NO DEFAULT: N/A

- Display the onscreen help message and exit the program.

`--directory /path/to/input/directory`

REQUIRED: NO

DEFAULT: the current working directory at time of invocation

- The input directory that the program will search for whatever input files are specified. If those files are not found in this directory, the program's behavior is governed by the value of the --interactive flag

`--fileList file1 [file2] [file3] ...`

REQUIRED: NO

DEFAULT: The list of all files in --directory (non-recursive)

- The list of files that the program should attempt to process for output. Most output functions (with the exception of the function designed for MOPITT CO) are designed to accept an arbitrary number of inputs and only use that data which fits the requirements.

- Processing files without relevant information (IE files from the wrong day) can be computationally expensive. It is strongly advised that this parameter be used to include only relevant files whenever possible.

- Locations of data (current as of 11/14/14)

- MOPITT - <http://eosweb.larc.nasa.gov/HPDOCS/datapool/>
- NASA OMI - <http://mirador.gsfc.nasa.gov/>
- KNMI OMI - <http://www.temis.nl/airpollution/no2.html>
- MODIS AOD - <http://ladsweb.nascom.nasa.gov/data/search.html>

```
--filetype {HDFmopittl2_generic,
MOPITT_CO_NASA_HDF_V5,HDFnasaomil2_generic,
OMI_NO2_NASA_HDF_v1_2,HDFknmiomil2_generic,
OMI_NO2_KNMI_HDF_v2_0_preFeb2006,
OMI_NO2_KNMI_HDF_v2_0_postFeb2006,
OMI_NO2_NASA_HDF_v3, OMI_HCHO_NASA_HDF_v3,
OMI_SO2_NASA_HDF_v3, HDFmodisl2_generic,
MODIS_AOD_NASA_Collection_5}
```

REQUIRED: YES
DEFAULT: N/A

▪The type of file we're attempting to read in. Must be one of the options listed above. This must match the format of the file (currently only the standard level 2 files for each of the above listed instruments/retrievals are supported).

▪Some filetypes require additional "parser parameters". These can be passed in at the command line under the --projAttrs flag and should be formatted just like any projection attribute or output function attribute for text file input.

HDFmopittl2_generic – Generic input filetype for NASA's level 2 MOPITT CO retrieval

Default output function:
unweighted_filtered_MOPITT_avg_netCDF

Default parameters provided: N/A Additional attributes required: N/A MOPITT_CO_NASA_HDF_V5 – Input filetype for NASA's level 2 MOPITT CO retrieval (version 5)

Default output function:
unweighted_filtered_MOPITT_avg_netCDF
Default parameters provided:
time
longitude
solZenAng
surfTypeField
colMeasField
outUnits
dimLabels
dimSizes
Additional attributes required: N/A

HDFknmiomil2_generic – The generic filetype for the OMI NO2 data as processed by KNMI (the DOMINO retrieval)

Default output function:
OMNO2e_netCDF_avg
Default parameters provided:
N/A
Additional attributes required:
N/A

OMI_NO2_KNMI_HDF_v2_0_preFeb2006 – The filetype for OMI NO2 as processed by KNMI (version 2.0 of the DOMINO retrieval). All input files should be before the shift in vertical layers that took place in February 2006.

Default output function:
OMNO2e_netCDF_avg
Default parameters provided:
overallQualFlag
cloudFrac
solarZenithAngle
time

longitude
pixIndXtrackAxis
outUnits
extraDimLabel
extraDimSize

Addiontal attributes required:
N/A

OMI_NO2_KNMI_HDF_v2_0_postFeb2006 – The filetype for OMI NO2 as processed by KNMI (version 2.0 of the DOMINO retrieval). All input files should be after the shift in vertical layers that took place in February 2006.

Default output function:
OMNO2e_netCDF_avg

Default parameters provided:
overallQualFlag
cloudFrac
solarZenithAngle
time
longitude
pixIndXtrackAxis
outUnits
extraDimLabel
extraDimSize

Addiontal attributes required:
N/A

HDFnasaomil2_generic – The generic filetype for the OMI NO2 data as processed by NASA (the OMNO2 product)

Default output function:
OMNO2e_netCDF_avg

Default parameters provided:

N/A

Additional parameters required:

cornerDir - the directory containing the auxillary corner files that the parser needs if the selected map function makes use of them.

cornerFileList - A list of the corner files in cornerDir that should be searched for corner files that match the input file. Should be comma-separated list of arbitrary length. Set to empty string (right hand side of equals sign or colon blank for input file or command line, respectively) to use all files in the cornerdir. Order does not matter, files will be matched based on orbit number.

OMI_NO2_NASA_HDF_v1_2 – The filetype for OMI NO2 data as processed by NASA (version 1.2 of the OMNO2 product).

Default output function:

OMNO2e_netCDF_avg

Default parameters provided:

overallQualFlag
cloudFrac
solarZenithAngle
time
longitude
pixIndXtrackAxis
outUnits
extraDimLabel
extraDimSize

Additional parameters required:

cornerDir - the directory containing the auxillary corner files that the parser needs if the selected map function makes use of them.

cornerFileList - A list of the corner files in cornerDir that should be searched for corner files that match the input file. Should be comma-separated list of arbitrary length. Set to empty string (right hand side of equals sign or colon blank for input file or command line, respectively) to use all files in the cornerdir. Order does not matter, files will be matched based on orbit number.

OMI_NO2_NASA_HDF_v3 – The filetype for OMI NO2 data as processed by NASA (version 3 of the OMNO2 product).

Default output function:

OMNO2e_netCDF_avg_v3

Default parameters provided:

overallQualFlag
cloudFrac
solarZenithAngle
time
longitude
pixIndXtrackAxis
outUnits
extraDimLabel
extraDimSize

Additional parameters required:

cornerDir - the directory containing the auxillary corner files that the parser needs if the selected map function makes use of them.

cornerFileList - A list of the corner files in cornerDir that should be searched for corner files that match the input file. Should be comma-separated list of arbitrary length.

Set to empty string (right hand side of equals sign or colon blank for input file or command line, respectively) to use all files in the cornerdir. Order does not matter, files will be matched based on orbit number.

OMI_HCHO_NASA_HDF_v3 – The filetype for OMI HCHO data as processed by NASA (version 3 of the OMHCHO product).

Default output function:
OMIHCHO_netCDF_avg

Default parameters provided:
overallQualFlag
cloudFrac
solarZenithAngle
time
longitude
pixIndXtrackAxis
outUnits
extraDimLabel
extraDimSize

Additional parameters required:
cornerDir - the directory containing the auxillary corner files that the parser needs if the selected map function makes use of them.
cornerFileList - A list of the corner files in cornerDir that should be searched for corner files that match the input file. Should be comma-separated list of arbitrary length. Set to empty string (right hand side of equals sign or colon blank for input file or command line, respectively) to use all files in the cornerdir. Order does not matter, files will be matched based on orbit number.

OMI_SO2_NASA_HDF_v3 – The filetype for OMI SO2 data as processed by NASA (version 3 of the OMSO2 product).

Default output function:
OMSO2_netCDF_avg_v3

Default parameters provided:
overallQualFlag
cloudFrac
solarZenithAngle
time
longitude
pixIndXtrackAxis
outUnits
extraDimLabel
extraDimSize

Additional parameters required:
cornerDir - the directory containing the auxillary corner files that the parser needs if the selected map function makes use of them.
cornerFileList - A list of the corner files in cornerDir that should be searched for corner files that match the input file. Should be comma-separated list of arbitrary length. Set to empty string (right hand side of equals sign or colon blank for input file or command line, respectively) to use all files in the cornerdir. Order does not matter, files will be matched based on orbit number.

HDFmodisl2_generic – The generic filetype for MODIS AOD data as processed by NASA. Support for QA-mean averaging will be added in a future update.

Default output function:
MODIS_simp_avg
Default parameters supplied:
N/A
Additional parameters required:
N/A

MODIS_AOD_NASA_Collection_5 – The filetype for level 2 MODIS Aerosol Optical Depth data as processed by NASA (Collection 5)

Default output function:
MODIS_simp_avg
Default parameters provided:
time
extraDimSize
extraDimLabels

Additional parameters required:
N/A

`--gridProj {latlon, lcc2par}`
REQUIRED: YES

DEFAULT: N/A

The grid projection used to define the target grid (the grid that we wish to regrid our data to). Must be one of the above options. Further details on these options are as follows:

latlon - The Plate Caree projection, also known as the "unprojected" projection. x and y are mapped directly to longitude and latitude, respectively.

REQUIRED PARAMETERS:

xCell - The size of a gridcell in the x (longitude) direction. In degrees.

yCell - The size of a gridcell in the y (latitude) direction. In degrees.

xOrig - The longitude of the lower-left corner of the domain

yOrig - The latitude of the lower-left corner of the domain

nRows - The number of rows in the grid. nCols - The number of columns in the grid.

lcc2par - The Lambert Conic Conformal projection (2 parallel construction). x and y are transformed and scaled, then mapped to latitude and longitude via the projection. The projection parameters must be accurate to get the correct output grid. All required parameters are available in the GRIDDESC file associated with the MM3 modeling system. A description of the GRIDDESC format can be found at:
http://www.baronams.com/products/ioapi/GRIDDESC.html

REQUIRED PARAMETERS:

stdPar1 - One of the 2 standard parallels used to define the Lambert Conic Conformal projection. Must be a valid latitude, in degrees.

stdPar2 - The second standard parallel used to define the Lambert Conic Conformal projection. Set this equal to the same value as stdPar1 if the single-parallel form of the projection is being used. Must be a valid latitude in degrees.

refLat - The reference latitude upon which the projection is centered. This is the YCENT value in the GRIDDESC file. In degrees.

refLon - The reference longitude upon which the projection is centered. This is BOTH the XCENT and PROJ_GAMMA values in the GRIDDESC file. If these values are not identical, do not use this function. In degrees.

xOrig - The location of the origin in projected x coordinates. This is the XORIG value in the GRIDDESC file. In same units as earthRadius.

yOrig - The location of the origin in projected y coordinates. This is the YORIG value in the GRIDDESC file. In same units as earthRadius.

xCell - The x dimension of a cell, in projected coordinates. In the same units as earthRadius. This is the XCELL value in the GRIDDESC file.

yCell - The y dimension of a cell, in projected coordinates. In the same units as earthRadius. This is the YCELL value in the GRIDDESC file.

nRows - The number of rows in the grid.

nCols - The number of columns in the grid.

earthRadius - The assumed radius of the Earth (assumed spherical). Must match units used for xCell and yCell.

`--projAttrs name1:value1 name2:value2 ...`
REQUIRED: YES

DEFAULT: N/A

- The attributes required for the chosen projection. Must have all the required attributes for that projection. The required parameters for each projection are listed under that projection's name above.

- Case-sensitive. Attribute names must EXACTLY match those laid out above.

`--mapFunc {point_in_cell, regional_intersect}`
REQUIRED: YES

DEFAULT: N/A

- The mapping function that will be used to assign pixels to cell(s). Certain datasets have restrictions on which mapping functions are usable.

- Also responsible for computing the "geometric weight" of pixels. That is, this function computes the weight unique to a cell/pixel combination. At present, neither of the functions provide this functionality.

point_in_cell - Maps pixels defined by a single lat/lon (usually the cell center) to whatever grid cell that point lies inside in projected space. This assigns each pixel to a unique grid cell. Grid cells are open on the top and right sides and closed on the left and lower sides (with directions defined according to the projected coordinate system). This function is supported by all currently available input filetypes. NOTE: for filetypes where regional_intersect is available it is strongly recommended to use regional_intersect over point_in_cell.

regional_intersect - Maps pixels (as defined by pairs of geocoordinates that nominally correspond to pixel corners) to ALL gridcells intersected. No geometric weights are calculated. This function is currently supported by HDFnasaomil2, HDFnasaomihchol2, HDFnasaomiv3l2 and HDFknmiomil2 filetypes only. Makes several assumptions:

– Polar discontinuities not encountered

– Projection is NOTglobal

– Grid is rectilinear in projected space.

– Pixelsareconvexpolygons.

global_intersect - Maps pixels (as defined by pairs of geocoordinates that nominally correspond to pixel corners) to ALL gridcells intersected. No geometric weights are calculated. This function is currently supported by HDFnasaomil2, HDFnasaomihchol2, HDFnasaomiv3l2 and HDFknmiomil2 filetypes only. Makes several assumptions:

– Polar discontinuities not encountered

– Grid is rectilinear in projected space.

– Pixels are convex polygons.

– East and west edges of projection lie on a common meridian

--outFunc {OMNO2e_netCDF_avg,OMIHCHO_netCDF_avg, MODIS_simp_avg_netCDF,OMNO2e_netCDF_avg_v3, unweighted_filtered_MOPITT_avg_netCDF}

REQUIRED: NO
DEFAULT: Depends on the value chosen for --filetype

▪The function that computes the output and writes the output file. Functions are given significant freedom, but all current functions take some kind of average and write it to an output file.

▪These functions are frequently designed around a particular instrument or input format. Efforts are made to make them as general as possible, but specialized output functions are only guaranteed (and really should only be used) for the parser types for which they have been designed. To this end, the associated function will be chosen by default for all filetypes, though it can always be overridden with this command.

▪To see which output function is the default for a particular filetype, see the description of the filetype above.

▪In all cases where a fieldname must be given for a parameter it is the short name (the name used to access the field through the parser) that must be given. The fieldnames must correspond to the official field names in the data file. These can usually be found in the data documentation. Documentation for a few commonly processed formats can be found at:

- OMI (KNMI) - <http://www.temis.nl/docs/OMI_NO2_HE5_1.0.2.pdf>

- OMI NO2 (NASA) - <http://disc.sci.gsfc.nasa.gov/Aura/data-holdings/OMI/documents/v003/OMNO2_readme_v003.pdf>

- OMI (NASA) HCHO –

<http://www.cfa.harvard.edu/atmosphere/Instruments/OMI/

PGEReleases/READMEs/OMHCHO_README.pdf>

- MOPITT - <http://www.acd.ucar.edu/mopitt/v5_users_guide_beta.pdf>

OMNO2e_netCDF_avg - Averaging algorithm based on the NASA OMI level 2 to level 3 processing algorithm. Designed for the OMI level 2 filetypes (HDFnasaomil2 and HDFknmiomil2) and the default for those filetypes. Use with other filetypes is of questionable utility.

Outputs results to a netCDF file.

Further details available in the official NASA documentation located at <http://disc.sci.gsfc.nasa.gov/Aura/data-holdings/OMI/omno2e_v003.shtml>

Assumptions:

- Data is at most 3 dimensional.
- Invalid pixels are marked with an overall quality flag.
- Timestamps are in the TAI93format.

OMNO2e_netCDF_avg_v3 - Averaging algorithm based on the NASA OMI level 2 to level 3 processing algorithm. Designed for the OMI NO2 level 2 filetypes (HDFnasaomiv3l2). Use with other filetypes is of questionable utility.

Outputs results to a netCDF file.

Further details available in the official NASA documentation located at <http://disc.sci.gsfc.nasa.gov/Aura/data-holdings/OMI/omno2e_v003.shtml>

Assumptions:

- Data is at most 3 dimensional.
- Invalid pixels are marked with an overall quality flag.
- Pixels affected by row anomalies are marked with an xtrack quality flag.
- Timestamps are in the TAI93format.

OMIHCHO_netCDF_avg - Averaging algorithm based on the NASA OMI level 2 to level 3 processing algorithm. Designed for the OMI HCHO level 2 filetypes (HDFnasaomihchol2). Use with other filetypes is of questionable utility.

Outputs results to a netCDF file.

Further details available in the official NASA documentation located at <http://disc.sci.gsfc.nasa.gov/Aura/data-holdings/OMI/omno2e_v003.shtml>

Assumptions:
- Data is at most 3 dimensional.
- Invalid pixels are marked with main data quality flag.
- Pixels affected by row anomalies are marked with an xtrack quality flag.
- Timestamps are in the TAI93format.

OMISO2_netCDF_avg - Averaging algorithm based on the NASA OMI level 2 to level 3 processing algorithm. Designed for the OMI SO2 level 2 filetypes (HDFnasaomiv3l2). Use with other filetypes is of questionable utility.

Outputs results to a netCDF file.

Further details available in the official NASA documentation located

at<http://disc.sci.gsfc.nasa.gov/Aura/data-holdings/OMI/omno2e_v003.shtml>

Assumptions:

- Data is at most 3 dimensional.
- Invalid pixels are marked with main data quality flag.
- Pixels affected by row anomalies are marked with an xtrack quality flag.
- Timestamps are in the TAI93format.

unweighted_filtered_MOPITT_avg_netCDF – Averaging algorithm based on the NASA algorithm for processing level 2 MOPITT CO data to level 3 MOPITT CO data. Designed for the MOPITT level 2 filetypes (HDFmopittl2 only at present) and is the default for that filetype. Use with other filetypes is discouraged.

Outputs results to a netCDF file.

IMPORTANT: Only 1 input file may be used with this function. Conveniently, NASA currently provides data in 1-day granules.

Further details available in the official NASA documentation: Deeter, Merritt N (2009). MOPITT (Measurements Pollution in the Troposphere) Validated Version 4 Product Users Guide. Available from <http://www.acd.ucar.edu/mopitt/products.shtml>

Assumptions/caveats:

- All fields are filtered based on the number of valid layers present in the specified column field. Including 2D fields.
- Timestamps are in the TAI93 format.

MODIS_simp_avg_netCDF – Averaging algorithm based on the NASA algorithm for processing level 2 MODIS AOD data to level 3 MODIS AOD data. Designed for the MODIS level 2 filetypes (HDFmodisl2 and Collection 5) and is the default output function for those filetypes. Use with other filetypes is discouraged.

Outputs results to a netCDF file. Further details available in the official NASA documentation:
Hubanks, et al. (2008) MODIS Atmosphere L3 Gridded Product
Algorithm Theoretical Basis Document. Available from <http://modis-atmos.gsfc.nasa.gov/_docs/L3_ATBD_2008_12-04.pdf>
Assumptions/caveats:
Does not factor in QA weights. A more general QA-weighted output function will be added in a future release.
– Timestamps are in the TAI93 format.

--outFuncAttrs name1:value1 name2:value2

REQUIRED: YES

DEFAULT: N/A

▪The attributes required for the chosen output function. Must have all required attributes for that output function. The required parameters for each projection are listed above.

▪If one of the "value" elements contains whitespace, enclose the entire name:value pair in double quotes. For example:

the:full_monty <- okay

"the:full monty" <- okay

the:full monty <- not okay

- In many cases, a comma delimited list is requested. Make sure that elements of the list are not separated by spaces. For example:

  pythons:EricIdle,JohnCleese <- okay

  pythons:EricIdle, JohnCleese " <- not okay

  pythons:Eric Idle,John Cleese"<- okay

  "pythons:Eric Idle, John Cleese"<- not okay

- Case-sensitive. Attribute names must EXACTLY match those laid out below.

- Below are the required parameters for each existing output function. { } contain usable/recommended parameters for applicable filetypes. These are based on the current versions of the data at the time of writing and should bedouble-checked:

OMNO2e_netCDF_avg

overallQualFlag - The name of the field containing the overall quality flag for the pixels. This flag should be true (1) for invalid pixels and false (0) for valid pixels. { OMI KNMI – TroposphericColumnFlag OMI NASA NO2 – vcdQualityFlags OMI NASA HCHO - MainDataQualityFlag }

cloudFrac - The name of the field containing the cloud fractions. { OMI KNMI - CloudFraction OMI NASA - CloudFraction }

solarZenithAngle - The name of the field the solar zenith angles in degrees.{ OMI KNMI - SolarZenithAngle OMI NASA - SolarZenithAngle }

time - The name of the field containing the timestamps. Timestamps are assumed to be in the TAI-93 format. {OMI KNMI – Time OMI NASA - Time}

longitude - The name of the field containing the longitudes at cell centers. Longitudes should be in degrees east. { OMI KNMI – Longitude OMI NASA - Longitude }

inFieldNames - The names of the fields desired to be output. Input as comma delimited list.

outFieldNames - The names of the output variables (even if they are to be the same as input variables). Should be a comma- delimited list co-indexed to inFieldNames

outUnits - The units of the variables to be written out. Should be a comma-delimited list co-indexed to inFieldNames

extraDimLabel - Label for the extra dimension (should the variable have an extra dimension). Ignored in the case of a 2D variable. Should be a comma-delimited list co-indexed to inFieldNames

extraDimSize - The size of the extra dimensions (should the variable have an extra dimension). For 2D variables, must be set to 0. (zero) Should be a comma-delimited list co-indexed to inFieldNames.

timeComparison - Must be set to either "local" or "UTC". Determines how the file timestamps are compared to the start/stop time. If set to "local", then the file timestamps are converted to local time on a pixel-by-pixel basis (using longitude to estimate time zone) before being compared to time boundaries. If set to "UTC" the file timestamps (which are assumed to be in UTC) are compared against the start/stop time directly.

timeStart - The earliest time for which data should be recorded into the output file. All times in input files before this time will be filtered out. Must be in the format:

hh:mm:ss_MM-DD-YYYY

timeStop - The latest time for which data should be recorded into the output files. All times in input files after this time will be filtered out. Must be in the format:

hh:mm:ss_MM-DD-YYYY

cloudFractUpperCutoff - The maximum cloud fraction to allow before excluding pixel from average. Suggested value from NASA is 0.3

solarZenAngUpperCutoff - The maximum solar zenith angle to allow before excluding pixel from average. Suggested value from NASA is 85. Must be in degrees.

pixIndXtrackAxis - The dimension order (0 based) of the "cross- track" dimension (whichever dimension has size 60). For all currently known cases set equal to 1 (depends on the construction of the parser function. If you rewrite the parser, check this).

fillVal - The value to use as a fill value in the output netCDF file. This value will replace any missing or invalid output values.

includePixelCount - if set to “True”, the output file will include the field “ValidPixelCount” which will have a count of the valid pixels for each cell. If set to “False” this extra field will not be included.

unweighted_filtered_MOPITT_avg_netCDF –

time - The name of the field containing timestamps. Timestamps are assumed to be in the TAI-93 format. { MOPITT - Time }

longitude - The name of the field containing the longitudes at cell centers. Longitudes should be in degrees east. { MOPITT - Longitude }

inFieldNames - The names of the fields desired to be output.

Input as comma delimited list.

outFieldNames - The names of the output variables (even if they are to be the same as input variables). Should be a comma- delimited list co-indexed to inFieldNames.

outUnits - The units of the variables to be written out. Should be a comma-delimited list co-indexed to inFieldNames.

logNormal - List of boolean strings that specify how to take the averages of the corresponding fields. If the string is "True" that field is averaged assuming a lognormal distribution. If the string is "False" that field is averaged assuming a normal distribution. Official documentation (linked above) has further information on when log-average is appropriate. Should be a comma-delimited list co-indexed to inFieldNames

dimLabels - List of names of the extra dimensions in the output file. Must be a forwardslash-delimited list of comma- delimited lists of labels. Fields with no extra dimensions may be left blank. For example, if there are four inFields, the first and third of which have no extra dimensions, the second of which has one ("foo"), and the fourth has two ("foo" and "bar"), the dimLabels entry should look like this:

/foo//foo,bar

The outer (forwardslash-delimited) list must be co-indexed to inFieldNames.

dimSizes - List of the sizes of the extra dimensions in the output file. Must be a forwardslash-delimited list of comma- delimited lists of integers. Fields with no extra dimensions may be left blank. For example, if there are four inFields, the first and third of which have no extra dimensions, the second of which has one (which has length

4), and the fourth has two (which have lengths four and five, respectively), the dimSizes entry should look like this:

/4//4,5

The outer (forwardslash-delimited list must be co-indexed to inFieldNames and each inner (comma-delimited) list should be the same size as the corresponding sublist in dimLabels.

timeStart - The earliest time for which data should be recorded into the output file. All times before this time in the input file(s) will be filtered out. Must be in the format:

hh:mm:ss_MM-DD-YYYY

timeStop - The latest time for which data should be recorded into the output file. All times after this time in the input file(s) will be filtered out. Must be in the format:

hh:mm:ss_MM-DD-YYYY

timeComparison - Must be set to either "local" or "UTC". Determines how the file timestamps are compared to the start/stop time. If set to "local", then the file timestamps are converted to local time on a pixel-by-pixel basis (using longitude to estimate time zone) before being compared to time boundaries. If set to "UTC" the file timestamps (which are assumed to be in UTC) are compared against the start/stop time directly.

fillVal - The value to use as a fill value in the output netCDF file. This value will replace any missing or invalid output values.

SolZenAngCutoff - The solar zenith angle that defines the day to night transition (we use the SZA to separate day and night pixels, which should not be averaged together). The geometric value here would be 90. Recommended value is

85. In degrees.

solZenAng - The name of the field containing the solar zenith angle (in degrees).{ MOPITT - Solar Zenith Angle }

dayTime - Boolean variable that indicates whether the output file should contain values from day or night. If set to "True" the output file will have daylight values. If set to "False" the output file will have night values.

surfTypeField - The name of the field containing the surface type index.{ MOPITT - Surface Index }

colMeasField - The name of the field containing the column measurement that will be used to determine how many valid layers are present in the cell. This field must be 4 dimensional, with the first extra dimension being the level and the first element of the second extra dimension containing NaN's at the appropriate levels.{ MOPITT - Retrieved CO Mixing Ratio Profile }

MODIS_simp_avg_netCDF –

timeStart - The earliest time for which data should be recorded into the output file. All times before this time in the input file(s) will be filtered out. Must be in the format: hh:mm:ss_MM-DD-YYYY

timeStop - The latest time for which data should be recorded into the output file. All times after this time in the input file(s) will be filtered out. Must be in the format: hh:mm:ss_MM_DD-YYYY

timeComparison - Must be set to either “local” or “UTC”. Determines how the file timestamps are compared to the start/stop time. If set to “local”, timestamps are converted to local time on a pixel-by-pixel basis (using longitude to estimate time zone) before being compared to time

boundaries. If set to “UTC” the file timestamps (which are assumed to be in UTC) are compared against the start/stop time directly.

Time - The name of the field containing timestamps. Timestamps are assumed to be in the TAI-93 format. { MODIS - Scan_Start_Time }

fillVal - The value to use as a fill value in the output netCDF file. This value will replace any missing or invalid output values

inFieldNames - The names of the fields desired to be output. Input as a comma-delimited list.

outFieldNames - The names of the output variables (even if they are to be the same as the input variables). Should be a comma-delimited list co-indexed to inFieldNames.

outUnits - The units of the variables to be written out. Should be a comma-delimited list co-indexed to inFieldNames.

extraDimSize - List of the sizes of the extra dimensions in the output file. Must be a comma-delimited list of integers. Fields with no extra dimensions should be left blank. For example, if there are four inFields, the first and third of which have no extra dimensions, the second of which has one (which has length four), and the fourth has one (which has length five), the extraDimSize entry should look like this: ,4,,5 The list must be co-indexed to inFieldNames.

extraDimLabels - List of names of the extra dimensions in the output file. Must be a comma-delimited list of strings. Fields with no extra dimensions should be left blank. For example, if there are four inFields, the first and third of which have no extra dimensions, the second of which has one (“foo”), and the fourth has one (“bar”), the

extraDimLabels entry should look like this: ,foo,,bar The list must be co-indexed to inFieldNames.

--outDirectory /path/to/output/directory

REQUIRED: YES
DEFAULT: N/A
The output directory to which the output file(s) should be written. Make sure that you have write permissions to this directory (the program will complain and quit out if you do not).

--outFileName FileName

REQUIRED: NO
DEFAULT: output1
The name of the output file itself. User is responsible for adding any file extensions here (IE .nc if it's a netCDF, .txt if it's an ASCII). Output will be written in outDirectory under this name.

--includeGrid GridFileName

REQUIRED: NO
DEFAULT: N/A
Supply this flag along with the absolute path to a file to which to write out the latitudes and longitudes of the gridcells defined by the selected projection.

--verbose {True,False}

REQUIRED: NO
DEFAULT: True
Determines how much command-line output the software provides while running. The default behavior (--verbose True) provide command line updates for most major subprocesses inside the software. Setting "False" here will cause the software to be completely silent while running.

--interactive {True,False}

REQUIRED: NO
DEFAULT: False
Determines how the program will handle invalid/nonexistent files. Under the default behavior (--interactive False) the software will automatically ignore any file it can't process and continue processing any other files in --fileList. If set to True, exectuion will be suspended and the user will be given several options when an invalid file is encountered.

--AttributeHelp ProjectionName/OutputFunctionName/FileType [...]

REQUIRED: NO
DEFAULT: N/A
Prints a message explaining the required attributes for a given output function or projection and then exits the program.
Inputting a filetype outputs the required attributes for default output function associated with that filetype.

--inFromFile FileName

REQUIRED: NO
DEFAULT: N/A
(New to version 1.1) Allows the use of a text input file as an alternate input format.
The format of the text input file must match that laid out in section X.X of this user guide. See the README for more on the format of this file

Worked Example

We are going to process and plot OMI NO2 (KNMI retrieval) data for September 1st 2011. For this exercise, we will input the processing command directly at the command line, but you can create a script to do so just as easily, or construct a text input file and run WHIPS with ----inFromFile instead.

The first step is to open the terminal and ensure that whips.py is installed and working properly. This can be determined by opening the terminal application and entering the following at the command line:

```
whips.py --help
```

You should get a print-out of the built-in help for WHIPS. This is one of the best resources if you find yourself confused about a particular argument.

Now that we know the program is working, let's start building the command we'll use to process our data. Because the command is so long, we can use the \ character to enter it on multiple lines, but for simplicity sake of this example, we will exclude it. Thus, we begin with the program name itself:

```
whips.py
```

The first thing the program needs is the directory in which it will find the input files, which we have already downloaded. We give it that information with the following command:

```
--directory /Users/nasa/whips_example/data/
```

We also need to tell WHIPS what kind of files to expect so it knows how to open and process them. We call this the "filetype" and

there is a specific filetype associated with all the input formats WHIPS understands. According to README.txt, the filetype for Level 2 OMI data as processed by KNMI from February 2006 onwards is "OMI_NO2_KNMI_HDF_v2_0_postFeb2006". We input this as:

```
--filetype OMI_NO2_KNMI_HDF_v2_0_postFeb2006
```

We've told the program where to look for files and how to open them. Now we should tell it which files we want it to open. This can be done with a command structured as follows:

```
   --fileList
   OMI-Aura_L2-OMDOMINO_2011m0901t1820-o37928_v003-2011m101
2t121613.he5
   OMI-Aura_L2-OMDOMINO_2011m0901t1641-o37927_v003-2011m101
2t121458.he5
   OMI-Aura_L2-OMDOMINO_2011m0901t1502-o37926_v003-2011m101
2t121342.he5
```

That's it for telling the program about the input files, now we have to tell it exactly how to process them. We'll begin by defining the grid on which we want the output data. We can specify the type of grid projection we want with the following:

```
--gridProj lcc2par
```

But that isn't enough to fully specify the grid – WHIPS needs more information about how the grid is laid out. We call these the "projection attributes".

In physical terms, our target grid is a 36 km grid over the Midwest. We will be using the Lambert Conic Conformal projection, assuming an Earth radius of 6370000 m. We chose 33°N and 45°N as our "true" parallels, and reference the projection against the point 40°N, 97°W. However, we wish to the grid's origin to lie at (-48km, -552 km) relative to the center of the projection. Our cells, as mentioned, will be 36km x 36 km. Our grid will be 30 cells wide and 32 cells tall.

Translated into the language of WHIPS, that grid can be described as follows. Note that we convert the earth radius to kilometers to match the units of our cell sizes and origin location.

```
      --projAttrs earthRadius:6370 stdPar1:33
stdPar2:45
        refLat:40 refLon:-97 xOrig:-48 yOrig:-552
   xCell:36 yCell:36 nRows:42 nCols:36
```

WHIPS needs to know how to interpolate the pixels onto the grid we just specified. In this case, because we have information available on the size and shape of the pixels, we'll get the best data quality by applying the "regional intersect" method. This can be specified as:

```
--mapFunc regional_intersect
```

WHIPS needs to know how to process the pixels into a "Level 3" product once they've been gridded. As was the case when we specified the grid, WHIPS needs more specific information, which we call the "output function attributes". The attributes required are unique to each satellite product, and are documented within README.txt. We tell WHIPS to look for output function attributes by entering the following:

```
--outFuncAttrs
```

Let's start by specifying which fields we'd like WHIPS to process and output. If we're interested in comparing to a model, we'll want at minimum the tropospheric column, the timestamp, and the averaging kernel. We'll need to tell WHIPS the names of these fields, which we can find in the documentation KNMI provides on their website at <http://www.temis.nl/docs/OMI_NO2_HE5_2.0_2011.pdf>.

```
inFieldNames:Time,AveragingKernel,TroposphericVerti
calColumn
```

Those might not be the names that you want in your output file, so WHIPS requires the user to specify corresponding output names as well. If you like the standard input names, feel free to list them again. Note that you should list output names (and all other parameters corresponding to specific fields) in the **same order** as the input field names.

```
outFieldNames:time,avgKern,tropVCD
```

WHIPS also needs a user defined fill value along with more parameters whose recommended values are defined in the README.txt:

```
fillVal:-9999 cloudFractUpperCutoff:0.3
solarZenAngUpperCutoff:85
```

Now we need to specify the time period we would like to look at, and the format of that time:

```
timeComparison:UTC
timeStart:00:00:00_09-01-2011
timeStop:23:59:59_09-01-2011
```

Next, we need to define the directory and file name we would like our data written to, and we need a couple additional settings for the running of the program itself. We also need to specify the name of the grid file, so that the grid definitions will be output (to increase efficiency of WHIPS and to avoid the same data reoccurring in multiple files, the output files do not incorporate grid definitions).

```
--outDirectory/Users/nasa/whips_example/output
            --outFileName OMI_DOMINO_20110901.nc
            --includeGrid \
   /Users/nasa/whips_example/OMI_DOMINO_GridFile.nc
            --verbose True
            --interactive True
```

After you have copied all of the parameters into the terminal, you may now hit enter, and wait for the program to run. Once WHIPS is finished, we can make a plot of the data.