MISR Toolkit
Users Guide

Multi-angle
Imaging
Spectro-
Radiometer

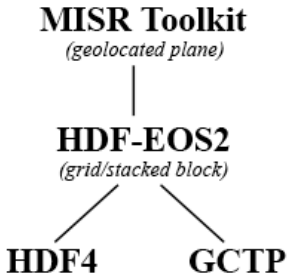
December 2013

Brian Rheingans
Jason Matthews
Charles Thompson
Sebastian Val

Jet Propulsion Laboratory, California Institute of Technology

JPL
Jet Propulsion Laboratory
California Institute of Technology

MISR Toolkit Users Guide



1) Introduction

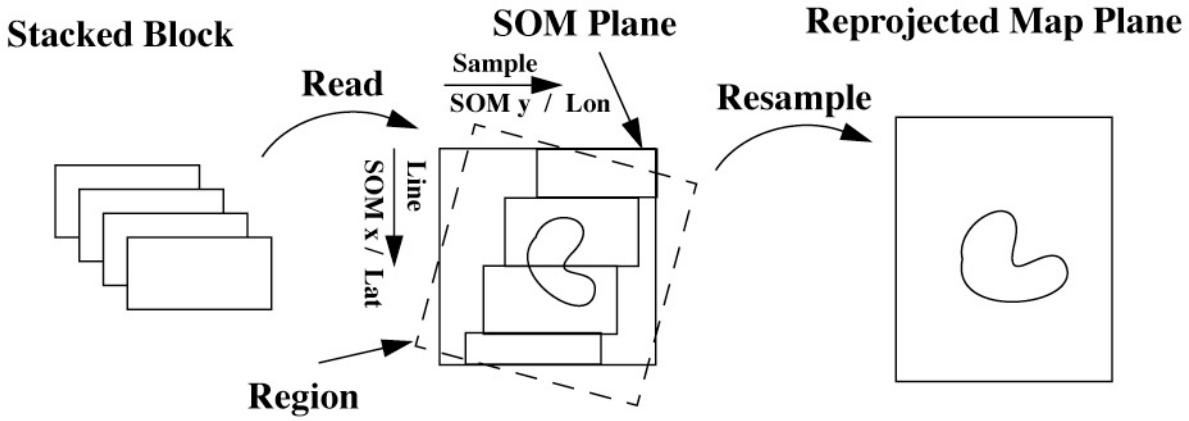
1. Purpose

The purpose of the MISR Toolkit is to provide a simplified programming interface to access MISR L1B2, L2 and some ancillary data products. These products are available in two forms: project standard and conventional. The project standard MISR products are in a HDF-EOS2 grid file format, but with an added “stacked-block” dimension. This block paradigm can make project standard MISR products difficult to read. To alleviate this situation a conventional MISR product was created to adhere to the HDF-EOS2 grid file format standard and remove the block paradigm. The conventional MISR products are also in a HDF-EOS2 grid file format and better follow the HDF-EOS2 conventions without the “stacked-block” dimension.

Either product form still requires a fair amount of work to access MISR data when using HDF-EOS2, such as block stitching, unpacking and unscaling of data and science parameters. The MISR Toolkit makes accessing MISR data of either form easy. It allows you the ability 1) to specify regions to read based on geographic location and extent or the more traditional path and block range; 2) to map between path, orbit, block, time range and geographic location; 3) to automatically stitch, unpack and unscale MISR data; 4) to perform coordinate conversions between lat/lon, SOM x/y, block/line/sample and line/sample of a data plane. This means geolocation can be computed instantly without referring to an ancillary data set lookup using a few MISR toolkit function calls.

This document only describes the C interface which provides the fundamental functionality. Other languages will be supported and described in other documents. Eventually, map reprojections will be supported as described in the concept diagram below.

2. Concept Diagram



3. Terminology

1. Product

   A product is any MISR ancillary, L1B2 or L2 file that is of the HDF-EOS2 grid file format that stores MISR imagery or science retrievals. There are two forms: project standard and conventional. Project standard MISR product files are generated at the DAAC using MISR operational production software. Conventional MISR product files can be requested when ordering MISR data and are converted upon delivery. The MISR Toolkit can read both forms.

2. File

   A file is synonymous with a product. MISR data is stored in files with a hdf extension. A file name is used to tell the toolkit what product to operate on. A file contains one or more grids.

3. Grid

   A grid is an HDF-EOS2 concept which describes geolocated data. It contains a set of projection equations (or references to them) along with their relevant parameters. Together, these relatively few pieces of information define the location of all points in the grid. The equations and parameters can then be used to compute the latitude and longitude for any point in the grid. A grid describes only one map projection, but many grids of different map projections may exist in a file. MISR, however, uses one map projection for all Level 1 and Level 2 products: the Space Oblique Mercator. A grid contains one or more fields.

4. Field

   Fields in a grid data set are rectilinear arrays of two or more dimensions. Most commonly, they are simply two dimensional rectangular arrays representing a spatial coverage of the globe. Each field contains data of similar scientific nature which must share the same data type. The data fields are related to each other by common geolocation. That is, a single set of geolocation information is used for all data fields within one grid data set. Also, a field can contain zero or more extra-dimensions beyond the spatial line and sample dimensions. These extra-dimensions may represent multiple bands, cameras, particle types, etc. It is important to note that the MISR Toolkit reads everything into a two dimensional data plane. Therefore, it is necessary to index a slice of the multi-dimensional fields. A zero-based indexed bracket notation on the fieldname is used when reading a slice of a multi-dimensional field.

   For example:

   RegBestEstimateSpectralOptDepth[0] = Blue band
   RegBestEstimateSpectralOptDepth[1] = Green band
   RegBestEstimateSpectralOptDepth[2] = Red band
   RegBestEstimateSpectralOptDepth[3] = Nir band

   This also applies to more than one extra-dimensions, so just add more bracketed indices. There are routines to query a field for the number, size, and name of any extra-dimensions. The MISR Data Production Specification document should be referenced for more information.

5. Path

   A path is an entire orbital revolution. The Terra satellite follows 233 distinct orbits that repeat every 16 days; each of those distinct repeating orbits is called a path and they are numbered from 1 to 233. MISR takes data in the daylight portion of the orbital path. The Terra satellite does not actually go directly over the poles. Path 37 always goes over, for example, Los Angeles.

6. Orbit

   An orbit is one Terra satellite revolution around the planet, numbered sequentially since launch. Orbit number continually increase where as path repeats. Therefore each path maps to multiple orbits. Also, orbit implies time because orbit increases with time. For example, path 37 over Los Angeles contains many orbits at different times since launch. The MISR Toolkit provides functions to map between path, orbit and time.

7. Block

   The total extent of each path that ever receives daylight is divided into 180 blocks. This is what constitutes the “stacked-block” concept of the project standard MISR product. Block 1 is on the spacecraft ascending side of the North Pole. Block number increases along the path through the North Polar region, down the descending side, and through the South Polar region, ending with Block 180 on the ascending side of the South Pole. In any given orbit, the daylight portion covers about 144 of those blocks. The 180 blocks is to allow for seasonal variation or summer and winter solstices.

8. Line

   Each block or data plane (any image for that matter) contains one or more lines of data divided up into samples. With respect to a 2-dimensional array; a row would be the same as a line.

9. Sample

   Each line is divided into samples. With respect to a 2-dimensional array a column would be the same as a sample. The MISR Toolkit provides functions to map between line and sample and geographic latitude/longitude, Space Oblique Mercator X/Y and even MISR specific block/line/sample.

10. Region

    A region is a MISR Toolkit specific concept and not a MISR product concept. In order to read MISR data using the Toolkit you must define a region of interest. There are several ways to define a region: 1) path and block range; 2) upper left and lower right geographic latitude/longitude corners; 3) center geographic latitude/longitude and an extent in kilometer, meters, degrees or pixels. A region only needs to be defined once to read any MISR data product that intersects that region. There are also functions to map between regions, paths and blocks.

11. Data Plane

    A data plane is a 2-dimensional array of data returned by the MtkReadData function. It is of arbitrary data type and dimension, which is determined by the file/grid/field read. It's relation to a region is only by approximate coverage. The data plane, in fact, is a grid of SOM map projected data, defined by the HDF-EOS2 grid projection parameters and map projection equations. In fact, a map structure is defined by MtkReadData that describes the grid and map projection parameters of the data plane. There are MISR Toolkit functions that map between data plane line/sample, SOM x/y and geographic latitude/longitude.

12. Space Oblique Mercator (SOM)

    Space Oblique Mercator is a modified cylindrical projection with the map surface defined by the satellite orbit. It was designed especially for continuous mapping of satellite imagery. It is basically conformal (preserves angles), especially in the region of the satellite scan. The ground-track of the satellite, a curved line on the globe, is shown as a curved line on the map and is continuously true to scale as the orbit continues. All meridians and parallels are curved lines, except meridians at each polar approach. The MISR Toolkit provides functions to map between the SOM x/y coordinate system and geographic latitude/longitude, data plane line/sample and MISR block/line/sample.

13. Packed degree, minutes, seconds

    Packed degrees, minutes, seconds or dms is a way to store these three values in a floating point number. It is of the form ddd0mm0ss.ssss.

2) How to obtain the MISR Toolkit

1. Website

The MISR Toolkit can be obtained from the Open Channel Foundation. The web address is http://www.openchannelsoftware.org. Search for "MISR" in the Quick Application Search field to access the MISR Visualization and Analysis Tools page. Select MISR Toolkit. Open Channel Foundation requires a login and has a license agreement process. Please refer to the Open Channel Foundation web page for details.

2. Library Dependencies

The MISR Toolkit depends on four other libraries which are *not* obtained via the Open Channel Foundation. They are HDF-EOS2, HDF4, zlib and libjpeg. The MISR file format is HDF-EOS2. More information on how to obtain HDF-EOS2 can be found at http://www.hdfeos.org. The HDF, zlib and libjpeg libraries can be obtained from http://hdf.ncsa.uiuc.edu.

Download and installation instructions for these additional libraries are provided in the MISR Toolkit bundled documentation. Please download the MISR Toolkit from Open Channel Foundation and refer to the accompanying instructions for library version, download location and installation instructions.

The Python interface also depends on the NumPy Python module and can can be downloaded from http://numpy.org.

3. Supported Architectures and Languages

Linux32 Fedora Core 19 or later

- C functions (source only)
- Python bindings (source only)

Linux64 Fedora Core 19 or later

- C functions (source only)
- IDL bindings (source only)
- Python bindings (source only)

MacOS X 10.6.1 or later (Intel)

- C functions (source only)
- IDL bindings (source only)
- Python bindings (source only)

Windows XP 32-bit

- C functions (dll & source)
- IDL bindings (dll & source – pre-built DLL require IDL 8.2)
- Python bindings (dll & source)

3) MISR Toolkit Routines

MISR Toolkit Routine Summary Table

Category	**C Routine Name**	***Description***
Util	MtkVersion	Reports MISR Toolkit version
	MtkDataBufferAllocate	Allocates 2D data plane buffer of specified type and size
	MtkDataBufferFree	Frees a 2D data plane buffer
	MtkDataBufferAllocate3D	Allocates 3D data buffer of specified type and size
	MtkDataBufferFree3D	Frees a 3D data buffer
	MtkStringListFree	Frees a string list
	MtkDateTimeToJulian	Convert date and time (ISO 8601) to Julian date
	MtkJulianToDateTime	Convert Julian date to date and time (ISO 8601)
FileQuery	MtkFileType	Retrieves MISR product type of a file
	MtkFileLGID	Retrieves MISR local granule ID (true product name)
	MtkFileVersion	Retrieves MISR file version
	MtkFillValueGet	Retrieves fill value of a file/grid/field
	MtkFileAttrList	Lists file attributes of a file
	MtkFileAttrGet	Retrieves file attributes of a file
	MtkGridAttrList	Lists grid attributes of a file/grid
	MtkGridAttrGet	Retrieves grid attributes of a file/grid

Category	**C Routine Name**	***Description***
	MtkFileToPath	Retrieves path of a file
	MtkFileToOrbit	Retrieves orbit of a file
	MtkFileToBlockRange	Retrieves block range of a file
	MtkFileToGridList	Retrieves grid list of file
	MtkFileGridToFieldList	Retrieves field list of a file/grid
	MtkFileGridToNativeFieldList	Retrieves native field list of a file/grid (excludes derived fields)
	MtkFileGridFieldToDimList	Retrieves dimension list of a file/grid/field
	MtkFileGridFieldCheck	Checks validity of a file/grid/field
	MtkFileGridFieldToDataType	Retrieves the data type of a file/grid/field
	MtkFileGridToResolution	Retrieves resolution of a file/grid
	MtkFileCoreMetaDataRaw	Read core metadata from a MISR product file into a buffer
	MtkFileCoreMetaDataQuery	Query file for core metadata parameter
	MtkFileCoreMetaDataGet	Get parameter from file core metadata
	MtkCoreMetaDataFree	Free core metadata structure
	MtkMakeFilename	Constructs a MISR filename from it's components
	MtkFindFileList	Searches a directory for MISR file from it's components (Nto available on Windows)
	MtkFileBlockMetaList	Lists block metadata of a file
	MtkFileBlockMetaFieldList	Lists block metadata fields of a file
	MtkFileBlockMetaFieldRead	Retrieves block metadata fields from a file
	MtkTimeMetaRead	Retrieves time metadata from L1B2 file for use with MtkPixelTime
UnitConv	MtkDdToDegMinSec	Converts decimal degrees to degrees, minutes, seconds
	MtkDdToDms	Converts decimal degrees to packed dms
	MtkDdToRad	Converts decimal degrees to radians
	MtkDegMinSecToDd	Converts degrees, minutes, seconds to decimal degrees
	MtkDegMinSecToDms	Converts degrees, minutes, seconds to packed dms
	MtkDegMinSecToRad	Converts degrees, minutes, seconds to radians
	MtkDmsToDd	Converts packed dms to decimal degrees
	MtkDmsToDegMinSec	Converts packed dms to degrees, minutes, seconds
	MtkDmsToRad	Converts packed dms to radians
	MtkRadToDd	Converts radians to decimal degrees
	MtkRadToDegMinSec	Converts radians to degrees, minutes, seconds
	MtkRadToDms	Converts radians to packed dms
CoordQuery	MtkBlsToLatLon	Converts block/line/sample to lat/lon
	MtkBlsToLatLonAry	Converts an array of block/line/sample to lat/lon
	MtkBlsToSomXY	Converts block/line/sample to SOM x/y
	MtkBlsToSomXYAry	Converts an array of block/line/samples to SOM x/y
	MtkLatLonToBls	Converts lat/lon to block/line/sample
	MtkLatLonToBlsAry	Converts an array of lat/lon to block/line/sample
	MtkSomXYToBls	Converts SOM x/y to block/line/sample
	MtkSomXYToBlsAry	Converts an array of SOM x/y to block/line/sample
	MtkLatLonToSomXY	Converts lat/lon to SOM x/y
	MtkLatLonToSomXYAry	Converts an array of lat/lon to SOM x/y

Category	**C Routine Name**	***Description***
	MtkSomXYToLatLon	Converts SOM x/y to lat/lon
	MtkSomXYToLatLonAry	Converts an array of SOM x/y to lat/lon
	MtkPathToProjParam	Retrieves MISR projection parameters for a given path
	MtkPathBlockRangeToBlockCorners	Computes block corner lat/lon coordinates for a path and block range
	MtkPixelTime	Computes pixel time given a SOM x/y coordinate and time metadata
MapQuery	MtkLSToLatLon	Converts data plane line/sample to lat/lon
	MtkLSToLatLonAry	Converts an array of data plane line/sample to lat/lon
	MtkLSToSomXY	Converts data plane line/sample to SOM x/y
	MtkLSToSomXYAry	Converts an array of data plane line/sample to SOM x/y
	MtkLatLonToLS	Converts lat/lon to data plane line/sample
	MtkLatLonToLSAry	Converts an array of lat/lon to data plane line/sample
	MtkSomXYToLS	Converts SOM x/y to data plane line/sample
	MtkSomXYToLSAry	Converts an array of SOM x/y to data plane line/sample
	MtkCreateLatLon	Creates latitude and longitude data plane buffers given mapinfo
OrbitPath	MtkLatLonToPathList	Retrieves a path list that crosses a given lat/lon
	MtkRegionToPathList	Retrieves a path list that crosses a given region
	MtkRegionPathToBlockRange	Retrieves the block range of a given region and path
	MtkOrbitToPath	Retrieves the path of a given orbit
	MtkTimeToOrbitPath	Retrieves an orbit/path of a given time
	MtkTimeRangeToOrbitList	Retrieves an orbit list over a given time range
	MtkPathTimeRangeToOrbitList	Retrieves an orbit list over a given path and time range
	MtkOrbitToTimeRange	Retrieves the time range of a given orbit
SetRegion	MtkSetRegionByPathBlockRange	Sets a region by path and block range
	MtkSetRegionByUlcLrc	Sets a region by upper left and lower right lat/lon
	MtkSetRegionByLatLonExtent	Sets a region by center lat/lon and a specified extent by specifying the units of extent
	MtkSnapToGrid	Snaps a region to a grid of a given path and resolution
ReadData	MtkReadData	Reads a region of MISR data given file/grid/field and a region
	MtkReadBlock	Reads a block of MISR data given file/grid/field and block number.
	MtkReadBlockRange	Read a block range of MISR data given file/grid/field and start block/end block into a 3D buffer
	MtkReadRaw	Reads a region of MISR data given file/grid/field and a region without unpacking and/or unscaling
WriteData	MtkWriteBinFile	Writes a raw binary file and info file given a data buffer and mapinfo
	MtkWriteBinFile3D	Writes a raw binary file and info file given a 3D data buffer
	MtkWriteEnviFile	Writes an IDL ENVI file given a data buffer and mapinfo
ReProject	MtkCreateGeoGrid()	Creates a regularly spaced geographic 2-D grid
	MtkResampleCubicConvolution()	Resamples source data using interpolation by cubic convolution
	MtkResampleNearestNeighbor()	Resamples source data using interpolation by cubic convolution
	MtkTransformCoordinates()	Transforms latitude/longitude coordinates into line/sample coordinates
Regression	MtkApplyRegression()	Applies regression to given data
	MtkDownSample()	Downsamples data by averaging pixels

Category	C Routine Name	*Description*
	MtkLinearRegressionCalc()	Uses linear regression to fit data
	MtkRegressionCoeffAllocate()	Allocates buffer to contain regression coefficients
	MtkRegressionCoeffCalc()	Calculates linear regression coefficients for translating values
	MtkRegressionCoeffFree()	Frees memory for regression coefficients
	MtkResampleRegressionCoeff()	Resamples regression coefficients at each pixel
	MtkSmoothData()	Smooths the given array with a boxcar average
	MtkUpsampleMask()	Upsamples a mask by nearest neighbor sampling

4) MISR Toolkit Programming Models

Detailed documentation for all Mtk routines are bundle with the source tarball and are accessible using a browser. There are also command line utilities in the src directory that demonstrate the usage of each function.

****** IMPORTANT INFO ******

In all the examples below the user should check the return status of the Mtk routines for MTK_SUCCESS or 0. Every routine below returns MTKt_status.

****** IMPORTANT INFO ******

With respect to the CoordQuery and MapQuery routines, line and sample are floating point data types because the map coordinates lat/lon and som x/y don't always map to a pixel center, so sometimes rounding is necessary. An integral line and sample refer to the pixel center, therefore they should be rounded to an integer index before using as an index into the data buffer.

****** IMPORTANT INFO ******

Some fields have additional dimensions beyond the line and sample dimensions of the data plane. These dimension names and sizes can be queried using the MtkFileGridFieldToDimList() routine. To read a data plane (or more specifically a slice) through this multi-dimensional field, you use a bracket index notation to specify the additional dimensions. For example, LandHDRF has two additional dimension, Nband (0-3) and Ncamera (0-8). To access the red band and nadir camera data plane (or slice), you would use LandHDRF[2][5]. Additional dimension indexing is always 0-based.

Some Examples:

1. Given geographic lat/lon, find intersecting paths, find orbits for one of these paths between a given time range, construct MISR product filenames.

```
double lat_dd = 32.2, lon_dd = -114.5
int pathcnt, *pathlist

MtkLatLonToPathList(lat_dd, lon_dd, &pathcnt, &pathlist)

start_time = “2005-01-25T00:00:00Z” // YYYY-MM-DDThh:mm:ssZ
end_time = “2005-01-27T23:59:59Z”
path = pathlist[pathcnt/2]    // Pick the center path from pathlist
int orbitcnt, *orbitlist

MtkPathTimeRangeToOrbit(path, start_time, end_time, &orbitcnt, &orbitlist)

char *filename
MtkMakeFilename(“/data”, ”GRP_ELLIPSOID”, “AF”, path, orbit[0], “F03_0024”,
                &filename)

free(pathlist)
free(orbitlist)
free(filename)
```

2. Given a geographic lat/lon and an extent in kilometers, define a region of interest, find intersecting paths and orbits.

```
double lat_dd = 32.2, lon_dd = -114.5
double lat_extent = 20, lon_extent = 10
MTKt_Region region = MTKT_REGION_INIT

MtkSetRegionByLatLonExtent(lat_dd, lon_dd, lat_extent, lon_extent, “km”, &region)

int pathcnt, *pathlist

MtkRegionToPathList(region, &pathcnt, &pathlist)

int orbitcnt, *orbitlist
path = pathlist[pathcnt/2]
starttime = “2004-01-20T02:03:04Z”   /* YYYY-MM-DDThh:mm:ss */
endtime = “2004-02-20T02:03:04Z”

MtkPathTimeRangeToOrbitList(path, starttime, endtime, &orbitcnt, &orbitlist)

int startblock, endblock

MtkRegionPathToBlockRange(region, path &startblock, &endblock)

free(pathlist)
free(orbitlist)
```

3. Given the region of interest above, read L2AS LandHDRF[3][4] (band 3, camera 4, (0-based)) into a data plane, and query map plane coordinates and LandHDRF data values.

```
char filename[] = “MISR_AM1_AS_LAND_P037_O029058_F06_0017.hdf”
char gridname[] = “SubregParamsLnd”
char fieldname[] = “LandHDRF[3][4]”
MTKt_DataBuffer databuf = MTKT_DATABUFFER_INIT
```

```
MTKt_MapInfo mapinfo = MTKT_MAPINFO_INIT

MtkReadData(filename, gridname, fieldname, region, &databuf, &mapinfo)

float line = 4, sample = 5
double lat, lon, somx, somy

MtkLSToLatLon(mapinfo, line, sample, &lat, &lon)
MtkLSToSomXY(mapinfo, line, sample, &somx, &somy)

databuf.data.f[round(line)][round(sample)] // .f due to floating point LandHDRF

MtkDataBufferFree(&databuf)
```

4. Given the region of interest above, read NRGB radiance data into data planes from an L1B2, and query map plane coordinates and NRGB radiance values. Note that for nadir camera (AN) all the bands are at the same resolution (275m). The off-nadir cameras, only the red band is at 275m resolution and the other bands are 1100m. Therefore, it is necessary to keep track of the appropriate mapinfo structure to be used in the MapQuery routines. It is possible to use the one mapinfo structure for all bands if they are the same resolution (or re-binned to the same resolution.) You can check the resolution by looking at the value in redmapinfo.resolution.

```
char filename[] = “MISR_AM1_GRP_ELLIPSOID_GM_P037_P029058_AN_F04_0006.hdf”
MTKt_DataBuffer red = MTKT_DATABUFFER_INIT
MTKt_DataBuffer grn = MTKT_DATABUFFER_INIT
MTKt_DataBuffer blu = MTKT_DATABUFFER_INIT
MTKt_DataBuffer nir = MTKT_DATABUFFER_INIT
MTKt_MapInfo redmapinfo = MTKT_MAPINFO_INIT
MTKt_MapInfo grnmapinfo = MTKT_MAPINFO_INIT
MTKt_MapInfo blumapinfo = MTKT_MAPINFO_INIT
MTKt_MapInfo nirmapinfo = MTKT_MAPINFO_INIT

MtkReadData(filename, “RedBand”, “Red Radiance”, region, &red, &redmapinfo)
MtkReadData(filename, “GreenBand”, “Green Radiance”, region, &grn, &grnmapinfo)
MtkReadData(filename, “BlueBand”, “Blue Radiance”, region, &blu, &blumapinfo)
MtkReadData(filename, “NIRBand”, “NIR Radiance”, region, &nir, &nirmapinfo)

float line = 4, sample = 5
double lat, lon, somx, somy

MtkLSToLatLon(redmapinfo, line, sample, &lat, &lon)
MtkLSToSomXY(redmapinfo, line, sample, &somx, &somy)

red.data.f[round(line)][round(sample)]  // .f due to floating point radiance
grn.data.f[round(line)][round(sample)]  // round line and sample to use as indices
blu.data.f[round(line)][round(sample)]
nir.data.f[round(line)][round(sample)]

double lat2 = 32.801543, lon2 = -115.636011
float line2, sample2

MtkLatLonToLS(redmapinfo, lat2, lon2, line2, sample2)
red.data.f[round(line2)][round(sample2)]

MtkDataBufferFree(&red)
MtkDataBufferFree(&grn)
MtkDataBufferFree(&blu)
MtkDataBufferFree(&nir)
```

5. Given a MISR product filename and a block range, define a region of interest. Use this region to read MISR data as above.

```
char filename[] = “MISR_AM1_GRP_ELLIPSOID_GM_P037_P029058_AN_F04_0006.hdf”
int startblock = 34, endblock = 65
int path
MTKt_Region region = MTKT_REGION_INIT

MtkFileToPath(filename, &path)
MtkSetRegionByPathBlockRange(path, startblock, endblock, &region)

MtkReadData(filename, “RedBand”, “Red Radiance”, region, &red, &redmapinfo)
MtkReadData(filename, “GreenBand”, “Green Radiance”, region, &grn, &grnmapinfo)
MtkReadData(filename, “BlueBand”, “Blue Radiance”, region, &blu, &blumapinfo)
MtkReadData(filename, “NIRBand”, “NIR Radiance”, region, &nir, &NIRmapinfo)

MtkDataBufferFree(&red)
MtkDataBufferFree(&grn)
MtkDataBufferFree(&blu)
MtkDataBufferFree(&NIR)
```

6. FileQuery routines (please refer to bundled documentation for more routines).

```
char *filename, *gridname, *fieldname
int path, orbit, startblock, endblock, ngrids, nfields, ndims, resolution
int *gridlist, *dimsize
char **fieldlist, **dimlist

MtkFileToPath(filename, &path)
MtkFileToOrbit(filename, &orbit)
MtkFileToBlockRange(filename, &startblock, &endblock)
MtkFileGridToResolution(filename, gridname, &resolution)
MtkFileToGridList(filename, &ngrids, &gridlist)
MtkFileGridToFieldList(filename, gridname, &nfields, &fieldlist)
MtkFileGridFieldToDimlist(filename, gridname, fieldname, &ndims, &dimlist[],
                          &dimsize)
free(gridlist)
free(dimsize)
MtkStringListFree(nfields, &fieldlist)
MtkStringListFree(ndims, &dimlist)
```

7. OrbitPath routines (please refer to bundled documentation for more routines).

```
double lat_dd, lon_dd
int path, npaths, orbit, norbits, startblock, endblock
int *pathlist, *orbitlist
char datetime[MTKd_DATETIME_LEN]
char starttime[MTKd_DATETIME_LEN], endtime[MTKd_DATETIME_LEN]
MTKt_Region region = MTKT_REGION_INIT

MtkLatLonToPathList(lat_dd, lon_dd, &npaths, &pathlist)
MtkOrbitToPath(orbit, &path)
MtkPathTimeRangeToOrbitList(path, startime, endtime, &norbits, &orbitlist)
MtkTimeRangeToOrbitList(starttime, endtime, &norbits, &orbitlist)
MtkTimeToOrbitPath(datetime, &orbit, &path)
MtkRegionToPathList(region, &npaths, &pathlist)
MtkRegionPathToBlockRange(region, path, &startblock, &endblock)
MtkOrbitToTimeRange(orbit, &start_time, &end_time)

free(pathlist)
free(orbitlist)
```

5) Compiling and Linking Instructions

To compile and link the routines `foo.c`

```
#include "MisrToolkit.h"
#include "MisrError.h"
#include <stdio.h>

int main() {

  MTKt_status status;
  int result;
```

```
  int path = 37;
  int resolution = 275;
  int block = 60;
  float line = 256;
  float sample = 1024;
  double lat_dd, lon_dd;
  int b;
  float l, s;
  int latdeg, londeg, latmin, lonmin;
  double latsec, lonsec;

  status = MtkBlsToLatLon(path, resolution, block, line, sample,
                          &lat_dd, &lon_dd);
  if (status != MTK_SUCCESS) return 1;

  status = MtkLatLonToBls(path, resolution, lat_dd, lon_dd, &b, &l, &s);
  if (status != MTK_SUCCESS) return 1;

  status = MtkDdToDegMinSec(lat_dd, &latdeg, &latmin, &latsec);
  if (status != MTK_SUCCESS) return 1;

  status = MtkDdToDegMinSec(lon_dd, &londeg, &lonmin, &lonsec);
  if (status != MTK_SUCCESS) return 1;

  printf("path = %d\n", path);
  printf("resolution = %d\n", resolution);
  printf("block, line, sample = %d, %6.1f, %6.1f\n", block, line, sample);
  printf("lat_dd, lon_dd = %f, %f\n", lat_dd, lon_dd);
  printf("lat deg, min, sec = %d:%02d:%5.2f\n", latdeg, latmin, latsec);
  printf("lon deg, min, sec = %d:%02d:%5.2f\n", londeg, lonmin, lonsec);
  printf("b, l, s = %d, %6.1f, %6.1f\n", b, l, s);

  result = bar(lat_dd, lon_dd);
  if (result) return 1;

  return 0;
}
```

and `bar.c`

```
#include "MisrToolkit.h"
#include "MisrError.h"
#include <stdio.h>

int bar( double lat, double lon ) {

  MTKt_status status;
  int pathcnt;
  int *pathlist;
  int i, j;
  int orbitcnt;
  int *orbitlist;
                                          /* YYYY-MM-DDThh:mm:ssZ */
  char *starttime = "2002-02-02T02:00:00Z"; /* 2002-02-02 02:00:00 UTC */
  char *endtime = "2002-05-02T02:00:00Z";   /* 2002-05-02 02:00:00 UTC */

  printf("starttime = %s\nendtime = %s\n", starttime, endtime);

  status = MtkLatLonToPathList(lat, lon, &pathcnt, &pathlist);
  if (status != MTK_SUCCESS) return 1;

  printf("Pathlist = ");
  for (i = 0; i < pathcnt; i++) {
    printf("%d ", pathlist[i]);
  }
  printf("\n");

  for (i = 0; i < pathcnt; i++) {
    status = MtkPathTimeRangeToOrbitList(pathlist[i], starttime, endtime,
                                         &orbitcnt, &orbitlist);
    if (status != MTK_SUCCESS) return 1;
    printf("Orbitlist for Path %d = ", pathlist[i]);
    for (j = 0; j < orbitcnt; j++) {
      printf("%d ", orbitlist[j]);
    }
    printf("\n");
  }
  return 0;
}
```

Compile and link

```
source <hdfeosdir>/bin/<arch>/hdfeos_env.csh
source $MTK_INSTALLDIR/bin/Mtk_c_env.csh
gcc $MTK_CFLAGS -c foo.c
gcc $MTK_CFLAGS -c bar.c
gcc -o baz foo.o bar.o $MTK_LDFLAGS
```

Executing `./baz`

```
path = 37
resolution = 275
block, line, sample = 60,  256.0, 1024.0
lat_dd, lon_dd = 38.130890, -110.848248
lat deg, min, sec = 38:07:51.21
lon deg, min, sec = -110:50:53.69
```

```
b, l, s = 60,  256.0, 1024.0
starttime = 2002-02-02T02:00:00Z
endtime = 2002-05-02T02:00:00Z
Pathlist = 35 36 37 38 39
Orbitlist for Path 35 = 11379 11612 11845 12078 12311 12544
Orbitlist for Path 36 = 11481 11714 11947 12180 12413
Orbitlist for Path 37 = 11350 11583 11816 12049 12282 12515
Orbitlist for Path 38 = 11452 11685 11918 12151 12384
Orbitlist for Path 39 = 11321 11554 11787 12020 12253 12486
```

6) MISR Toolkit Routine Reference

Bundled with the MISR Toolkit in the doc directory are automatically generated web pages which provides up-to-date function call interface definitions, structure definitions, return values, usage examples, etc. Included are both Python and IDL binding documentation.

7) Command Line Utilities

MISR Toolkit Command Line Utilities Summary Table

Utility	***Usage***
MtkVersion	`MISR Toolkit Version 1.2.0`

MtkDateTimeToJulian	`Usage: MtkDateTimeToJulian <Date and Time>`
	`  Date and Time in ISO 8601 format.`
	`Example: MtkDateTimeToJulian 2002-05-02T02:00:00Z`

MtkJulianToDateTime	`Usage: MtkJulianToDateTime <Julian Date>`
	`  Julian Date >= 1721119.5`
	`Example: MtkJulianToDateTime 2453728.27313`

MtkBlsToLatLon	`Usage: bin/MtkBlsToLatLon <--help> | <--path=path> <--res=resolution> <--bls=block,line,sample>`
	
	`Where: --path=path is the path number`
	`  --res=resolution is the resolution in meters`
	`  --bls=block,line,sample is block, line, and sample`
	`  --help is this info`
	
	`Example: MtkBlsToLatLon --path=1 --res=1100 --bls=22,101,22`

MtkLatLonToBls	`Usage: bin/MtkLatLonToBls <--help> | <--path=path> <--res=resolution> [<--dd=lat_dd,lon_dd> | <--rad=lat_r,lon_r> | <--dms=lat_dms,lon_dms>]`
	
	`Where: --path=path is the path number`
	`  --res=resolution is the resolution in meters`
	`  --dd=lat_dd,lon_dd is lat,lon in decimal degrees`
	`  --rad=lat_r,lon_r is lat,lon in radians`
	`  --dms=lat_dms,lon_dms is lat,lon in packed degrees, minutes and seconds`
	`  --help is this info`
	
	`Example: MtkLatLonToBls --path=1 --res=1100 --dd=82.740690,-3.310459`
	`  MtkLatLonToBls --path=1 --res=1100 --rad=1.444098,-0.057778`
	`  MtkLatLonToBls --path=1 --res=1100 --dms=82044026.490000,-3018037.650000`

MtkBlsToSomXY	`Usage: bin/MtkBlsToSomXY <--help> | <--path=path> <--res=resolution> <--bls=block,line,sample>`
	
	`Where: --path=path is the path number`
	`  --res=resolution is the resolution in meters`
	`  --bls=block,line,sample is block, line, and sample`
	`  --help is this info`
	
	`Example: MtkBlsToSomXY --path=1 --res=1100 --bls=22,101,22`

MtkSomXYToBls	`Usage: bin/MtkSomXYToBls <--help> | <--path=path> <--res=resolution> <--somxy=som_x,som_y>`

Utility	*Usage*
	Where: --path=path is the path number --res=resolution is the resolution in meters --somxy=som_x,som_y is SomX and SomY --help is this info Example: MtkSomXYToBls --path=1 --res=1100 --somxy=10529200.016621,622600.018066
MtkLatLonToSomXY	Usage: bin/MtkLatLonToSomXY <--help> | <--path=path> [<--dd=lat_dd,lon_dd> | <--rad=lat_r,lon_r> | <--dms=lat_dms,lon_dms>] Where: --path=path is the path number --dd=lat_dd,lon_dd is lat,lon in decimal degrees --rad=lat_r,lon_r is lat,lon in radians --dms=lat_dms,lon_dms is lat,lon in packed degrees, minutes and seconds --help is this info Example: MtkLatLonToSomXY --path=1 --dd=82.740690,-3.310459 MtkLatLonToSomXY --path=1 --rad=1.444098,-0.057778 MtkLatLonToSomXY --path=1 --dms=82044026.490000,- 3018037.650000
MtkSomXYToLatLon	Usage: ./MtkSomXYToLatLon <--help> | <--path=path> <-- somxy=som_x,som_y> [--rad | --dms] Where: --path=path is the path number --somxy=som_x,som_y is SomX and SomY --rad display output in radians --dms display output in degrees minutes seconds --help is this info Example: MtkSomXYToLatLon --path=1 --somxy=10529200.016621,622600.018066 MtkSomXYToLatLon --path=1 --somxy=10529200.016621,622600.018066 --rad MtkSomXYToLatLon --path=1 --somxy=10529200.016621,622600.018066 --dms
MtkCreateLatLon	Usage: ./MtkCreateLatLon <--help> | --path=<path_number> --res=<resolution> | [--setregion-path-blockrange=<path,start_blk,end_blk> | --setregion-ulclrc=<ulclat,ulclon,lrclat,lrclon> | --setregion-latlon-extent=<lat,lon,latext,lonext,extent_units>] --binfilename=<Lat/Lon Binary Output File> Where: --setregion-path-blockrange=path,start_blk,end_blk is path, start block, end block. --setregion-ulclrc=ulclat,ulclon,lrclat,lrclon is Upper Left Corner Lat, Lon and Lower Right Corner Lat and Lon. --setregion-latlon-extents=lat,lon,latext,lonext,extent_units is Lat, Lon in degrees and Extent in specified units. --binfilename=file is the output file. Note: The parameter extent_units is a case insensitive string that can be set to one of the following values: 1) degrees, deg, dd for degrees; 2) meters, m for meters; 3) kilometers, km for kilometers; and 4) 275m, 275 meters, 1.1km, 1.1 kilometers for pixels of a specified resolution per pixel. Example 1: MtkCreateLatLon --path=37 --res=1100 --setregion-path- blockrange=37,45,50 --binfilename=out.bin Example 2: MtkCreateLatLon --path=37 --res=1100 --setregion-latlon- extent=38,-111,3000,300,km --binfilename=out.bin Example 3: MtkCreateLatLon --path=37 --res=1100 --setregion-latlon- extent=38,-111,2000,300,1100m --binfilename=out.bin Example 4: MtkCreateLatLon --path=37 --res=1100 --setregion- ulclrc=51.5,-112,24,-109 --binfilename=out.bin
MtkPixelTime	Usage: ./MtkPixelTime <--help> | --hdffilename=<L1B2 Product File> --somxy=somx,somy

Utility	*Usage*
	Where: --hdffilename=file is a MISR L1B2 Product File. --somxy=som_x,som_y is SomX and SomY. Example: MtkPixelTime --hdffilename=../Mtk_testdata/in/MISR_AM1_GRP_ELLIPSOID_GM_P037_O002905 8_AA_F03_0024.hdf --somxy=10529200.016621,622600.018066
MtkDdToDegMinSec	Usage: ./MtkDdToDegMinSec <Decimal Degrees> Example: MtkDdToDegMinSec 130.08284167
MtkDegMinSecToDd	Usage: MtkDegMinSecToDd <--help> | <--deg=degrees> <--min=minutes> <--sec=seconds> Where: --deg=degrees is the number of degrees --min=minutes is the number of minutes --sec=seconds is the number of seconds --help is this info Example: MtkDegMinSecToDd --deg=130 --min=4 --sec=58.23
MtkDdToDms	Usage: MtkDdToDms <Decimal Degrees> Example: MtkDdToDms 130.08284167
MtkDmsToDd	Usage: MtkDmsToDd <--help> | <--dms=packed dms> Where: --dms=packed dms is packed degrees minutes seconds --help is this info Example: MtkDmsToDd --dms=130004058.23
MtkDmsToDegMinSec	Usage: MtkDmsToDegMinSec <--help> | <--dms=packed dms> Where: --dms=packed dms is packed degrees minutes seconds --help is this info Example: MtkDmsToDegMinSec --dms=130004058.23
MtkDegMinSecToDms	Usage: MtkDegMinSecToDms <--help> | <--deg=degrees> <--min=minutes> <--sec=seconds> Where: --deg=degrees is the number of degrees --min=minutes is the number of minutes --sec=seconds is the number of seconds --help is this info Example: MtkDegMinSecToDms --deg=130 --min=4 --sec=58.23
MtkRadToDd	Usage: MtkRadToDd <--help> | <--rad=radians> Where: --rad=radians is the number of radians --help is this info Example: MtkRadToDd --rad=2.270373886
MtkDdToRad	Usage: MtkDdToRad <Decimal Degrees> Example: MtkDdToRad 130.08284167
MtkRadToDegMinSec	Usage: MtkRadToDegMinSec <--help> | <--rad=radians> Where: --rad=radians is the number of radians --help is this info Example: MtkRadToDegMinSec --rad=2.270373886
MtkDegMinSecToRad	Usage: MtkDegMinSecToRad <--help> | <--deg=degrees> <--min=minutes> <--sec=seconds> Where: --deg=degrees is the number of degrees --min=minutes is the number of minutes --sec=seconds is the number of seconds --help is this info Example: MtkDegMinSecToRad --deg=130 --min=4 --sec=58.23
MtkRadToDms	Usage: MtkRadToDms <--help> | <--rad=radians> Where: --rad=radians is the number of radians

Utility	*Usage*
	--help is this info Example: MtkRadToDms --rad=2.270373886
MtkDmsToRad	Usage: MtkDmsToRad <--help> | <--dms=packed dms> Where: --dms=packed dms is packed degrees minutes seconds --help is this info Example: MtkDmsToRad --dms=130004058.23
MtkPathToProjParam	Usage: MtkPathToProjParam <path> <resolution meters>
MtkLatLonToPathList	Usage: MtkLatLonToPathList <--help> | <--dd=lat_dd,lon_dd> | <--rad=lat_r,lon_r> | <--dms=lat_dms,lon_dms> Where: --dd=lat_dd,lon_dd is lat,lon in decimal degrees --rad=lat_r,lon_r is lat,lon in radians --dms=lat_dms,lon_dms is lat,lon in packed degrees, minutes and seconds --help is this info Example: MtkLatLonToPathList --dd=-75.345,169.89 MtkLatLonToPathList --rad=-1.315,2.965 MtkLatLonToPathList --dms=-75020042.000,169053024.000
MtkRegionToPathList	Usage: MtkRegionToPathList <--help> | [--setregion-path-blockrange=<path,start_blk,end_blk> | --setregion-ulclrc=<ulclat,ulclon,lrclat,lrclon> | --setregion-latlon-extent=<lat,lon,latext,lonext,extent_units>] Where: --setregion-path-blockrange=path,start_blk,end_blk is path, start block, end block. --setregion-ulclrc=ulclat,ulclon,lrclat,lrclon is Upper Left Corner Lat, Lon and Lower Right Corner Lat and Lon. --setregion-latlon-extents=lat,lon,latext,lonext,extent_units is Lat, Lon in degrees and Extent in specified units. Note: The parameter extent_units is a case insensitive string that can be set to one of the following values: 1) degrees, deg, dd for degrees; 2) meters, m for meters; 3) kilometers, km for kilometers; and 4) 275m, 275 meters, 1.1km, 1.1 kilometers for pixels of a specified resolution per pixel. Example 1: MtkRegionToPathList --setregion-path-blockrange=37,45,75 Example 2: MtkRegionToPathList --setregion-latlon-extent=38,-111,3000,300,km Example 3: MtkRegionToPathList --setregion-latlon-extent=38,-111,2000,300,1100m Example 4: MtkRegionToPathList --setregion-ulclrc=51.5,-112,24,-109
MtkOrbitToPath	Usage: MtkOrbitToPath <orbit> Orbit >= 1000 Example: MtkOrbitToPath 12115
MtkPathBlockRangeToBlockCorners	Usage: MtkPathBlockRangeToBlockCorners <--help> | --path=<Path Number> --startblock=<Start Block> --endblock=<End Block> Where: --path=path_num is the path number. --startblock=start_block is starting block. --endblock=end_block is ending block. Example 1: MtkPathBlockRangeToBlockCorners --path=37 --startblock=35 --endblock=40
MtkRegionPathToBlockRange	Usage: MtkRegionPathToBlockRange <--help> | [--setregion-path-blockrange=<path,start_blk,end_blk> | --setregion-ulclrc=<ulclat,ulclon,lrclat,lrclon> | --setregion-latlon-extent=<lat,lon,latext,lonext,extent_units>] --path=<Path>]

Utility	*Usage*
	Where: --setregion-path-blockrange=path,start_blk,end_blk is path, start block, end block. --setregion-ulclrc=ulclat,ulclon,lrclat,lrclon is Upper Left Corner Lat, Lon and Lower Right Corner Lat and Lon. --setregion-latlon-extents=lat,lon,latext,lonext,extent_units is Lat, Lon in degrees and Extent in specified units. Note: The parameter extent_units is a case insensitive string that can be set to one of the following values: 1) degrees, deg, dd for degrees; 2) meters, m for meters; 3) kilometers, km for kilometers; and 4) 275m, 275 meters, 1.1km, 1.1 kilometers for pixels of a specified resolution per pixel. Example 1: MtkRegionPathToBlockRange --path=37 --setregion-path-blockrange=37,45,75 Example 2: MtkRegionPathToBlockRange --path=37 --setregion-latlon-extent=38,-111,3000,300,km Example 3: MtkRegionPathToBlockRange --path=37 --setregion-latlon-extent=38,-111,2000,300,1100m Example 4: MtkRegionPathToBlockRange --path=37 --setregion-ulclrc=51.5,-112,24,-109
MtkPathTimeRangeToOrbitList	Usage: MtkPathTimeRangeToOrbitList <--help> | <--path=path> <--start=YYYY-MM-DDThh:mm:ssZ> <--end=YYYY-MM-DDThh:mm:ssZ> Where: --path=path is the path number --start=YYYY-MM-DDThh:mm:ssZ --end=YYYY-MM-DDThh:mm:ssZ --help is this info Example: MtkPathTimeRangeToOrbitList --path=78 --start=2002-02-02T02:00:00Z --end=2002-05-02T02:00:00Z
MtkTimeRangeToOrbitList	Usage: MtkTimeRangeToOrbitList <--help> | <--start=YYYY-MM-DDThh:mm:ssZ> <--end=YYYY-MM-DDThh:mm:ssZ> Where: --start=YYYY-MM-DDThh:mm:ssZ --end=YYYY-MM-DDThh:mm:ssZ --help is this info Example: MtkTimeRangeToOrbitList --start=2002-02-02T02:00:00Z --end=2002-05-02T05:00:00Z
MtkTimeToOrbitPath	Usage: MtkTimeToOrbitPath <time> Where: <time>=YYYY-MM-DDThh:mm:ssZ Time must be on or after 2000-03-03 00:00:00 UTC Example: MtkTimeToOrbitPath 2002-02-02T02:00:00Z
MtkOrbitToTimeRange	Usage: MtkOrbitToTimeRange <Orbit Number> Orbit Number >= 1000 Example: MtkOrbitToTimeRange 24372
MtkFileToPath	Usage: MtkFileToPath <MISR Product File>
MtkFileToOrbit	Usage: MtkFileToOrbit <MISR Product File>
MtkFileToBlockRange	Usage: MtkFileToBlockRange <MISR Product File>
MtkFileGridToResolution	Usage: ./MtkFileGridToResolution <MISR Product File> <Grid Name>
MtkFileToGridList	Usage: MtkFileToGridList <MISR Product File>
MtkFileGridToFieldList	Usage: MtkFileGridToFieldList <MISR Product File> <Grid Name>
MtkFileGridToNativeFieldList	Usage: MtkFileGridToNativeFieldList <MISR Product File> <Grid Name>
MtkFileGridFieldToDimList	Usage: MtkFileGridFieldToDimList <MISR Product File> <Grid Name> <Field Name>
MtkFileVersion	Usage: MtkFileVersion <MISR Product File>
MtkFileLGID	Usage: MtkFileLGID <MISR Product File>

Utility	*Usage*
MtkFileAttrList	`Usage: MtkFileAttrList <MISR Product File>`
MtkFileAttrGet	`Usage: MtkFileAttrGet <MISR Product File> <Attribute Name>`
MtkGridAttrList	`Usage: MtkGridAttrList <MISR Product File> <Grid Name>`
MtkGridAttrGet	`Usage: MtkGridAttrGet <MISR Product File> <Grid Name> <Attribute Name>`
MtkFileCoreMetaDataQuery	`Usage: MtkFileCoreMetaDataQuery <MISR Product File>`
MtkFileCoreMetaDataGet	`Usage: MtkFileCoreMetaDataGet <MISR Product File> <Parameter Name>`

MtkMakeFilename	`Usage: MtkMakeFilename <--help> | <--dir=directory> <--prod=product> [ <--cam=camera> ] <--path=path> <--orbit=orbit> <--ver=version>`
	
	`Where: --dir=directory is base directory to append to file name.`
	`--prod=product is the product to search for.`
	`--cam=camera is the camera.`
	`--path=path is the path number.`
	`--orbit=orbit is the orbit number.`
	`--ver=version is the version number.`
	`--help is this info`
	
	`Example: MtkMakeFilename --dir=data --prod=GRP_TERRAIN_GM --cam=DA --path=123 --orbit=12345 --ver=F03_0024`
	`MtkMakeFilename --dir=data --prod=TC_ALBEDO --path=012 --orbit=12345 --ver=F04_0007`

MtkFindFileList	`Usage: MtkFindFileList <--help> | <--dir=directory> <--prod=product> [ <--cam=camera> ] <--path=path> <--orbit=orbit> <--ver=version>`
	
	`Where: --dir=directory is top level directory to search.`
	`--prod=product is the product to search for.`
	`--cam=camera is the camera.`
	`--path=path is the path number.`
	`--orbit=orbit is the orbit number.`
	`--ver=version is the version number.`
	`Note: All of the above options support regular expressions.`
	`--help is this info`
	
	`Example: MtkFindFileList --dir=. --prod="GRP.*" --cam=DA --path=".*" --orbit=".*" --ver=F03_0024`
	`MtkFindFileList --dir=. --prod="TC.*" --path=037 --orbit=".*" --ver=".*"`

MtkReadData	`Usage: bin/MtkReadData <--help> |`
	`[--entire-file |`
	`--setregion-path-blockrange=<path,start_blk,end_blk> |`
	`--setregion-ulclrc=<ulclat,ulclon,lrclat,lrclon> |`
	`--setregion-latlon-extent=<lat,lon,latext,lonext,extent_units> ]`
	`--hdffilename=<Input File>`
	`--gridname=<Grid Name>`
	`--fieldname=<Field Name>`
	`--binfilename=<Binary Output File>`
	
	`Where: --entire-file queries the hdffile for path and block range.`
	`--setregion-path-blockrange=path,start_blk,end_blk is path, start block, end block.`
	`--setregion-ulclrc=ulclat,ulclon,lrclat,lrclon is Upper Left Corner Lat, Lon and Lower Right Corner Lat and Lon.`
	`--setregion-latlon-extents=lat,lon,latext,lonext,extent_units is Lat, Lon in degrees and Extent in specified units.`
	`--hdffilename=file is a MISR Product File.`
	`--gridname=grid_name is the name of the grid.`
	`--fieldname=field_name is the name of the field.`
	`--binfilename=file is the output file.`
	
	`Note: The parameter extent_units is a case insensitive string that can be set to one of the following values:`
	
	`1) degrees, deg, dd for degrees;`
	`2) meters, m for meters;`
	`3) kilometers, km for kilometers; and`
	`4) 275m, 275 meters, 1.1km, 1.1 kilometers for pixels of a specified resolution per pixel.`
	
	`Example 1: MtkReadData --setregion-path-blockrange=37,45,75 --hdffilename=../Mtk_testdata/in/MISR_AM1_GRP_ELLIPSOID_GM_P037_O029058_AA_F03_0024.hdf --gridname=BlueBand --fieldname="Blue Radiance/RDQI"`

Utility	*Usage*
	--binfilename=out.bin Example 2: MtkReadData --setregion-latlon-extent=38,-111,3000,300,km --hdffilename=../Mtk_testdata/in/MISR_AM1_GRP_ELLIPSOID_GM_P037_O002905 8_AA_F03_0024.hdf --gridname=BlueBand --fieldname="Blue Radiance/RDQI" --binfilename=out.bin Example 3: MtkReadData --setregion-latlon-extent=38,- 111,2000,300,1100m --hdffilename=../Mtk_testdata/in/MISR_AM1_GRP_ELLIPSOID_GM_P037_O002905 8_AA_F03_0024.hdf --gridname=BlueBand --fieldname="Blue Radiance/RDQI" --binfilename=out.bin Example 4: MtkReadData --setregion-ulclrc=51.5,-112,24,-109 --hdffilename=../Mtk_testdata/in/MISR_AM1_GRP_ELLIPSOID_GM_P037_O002905 8_AA_F03_0024.hdf --gridname=BlueBand --fieldname="Blue Radiance/RDQI" --binfilename=out.bin Example 5: MtkReadData --entire-file --hdffilename=../Mtk_testdata/in/MISR_AM1_GRP_ELLIPSOID_GM_P037_O002905 8_AA_F03_0024.hdf --gridname=BlueBand --fieldname="Blue Radiance/RDQI" --binfilename=out.bin
MtkReadBlockRange	Usage: MtkReadBlockRange <--help> | [--entire-file | --hdffilename=<Input File> --gridname=<Grid Name> --fieldname=<Field Name> --startblock=<Start Block> --endblock=<End Block> --binfilename=<Binary Output File> Where: --entire-file queries the hdffile for the block range. --hdffilename=file is a MISR Product File. --gridname=grid_name is the name of the grid. --fieldname=field_name is the name of the field. --startblock=start_block is starting block. --endblock=end_block is ending block. --binfilename=file is the output file. Example 1: MtkReadBlockRange --hdffilename=../Mtk_testdata/in/MISR_AM1_GRP_ELLIPSOID_GM_P037_O002905 8_AA_F03_0024.hdf --gridname=BlueBand --fieldname="Blue Radiance/RDQI" --startblock=35 --endblock=40 --binfilename=out.bin Example 2: MtkReadBlockRange --entire-file --hdffilename=../Mtk_testdata/in/MISR_AM1_GRP_ELLIPSOID_GM_P037_O002905 8_AA_F03_0024.hdf --gridname=BlueBand --fieldname="Blue Radiance/RDQI" --binfilename=out.bin
MtkMisrToEnvi	Usage: ./MtkMisrToEnvi <--help> | [--entire-file | --setregion-path-blockrange=<path,start_blk,end_blk> | --setregion-ulclrc=<ulclat,ulclon,lrclat,lrclon> | --setregion-latlon-extent=<lat,lon,latext,lonext,extent_units>] --hdffilename=<Input File> --gridname=<Grid Name> --fieldname=<Field Name> --envifilename=<Envi Output File> Where: --entire-file queries the hdffile for path and block range. --setregion-path-blockrange=path,start_blk,end_blk is path, start block, end block. --setregion-ulclrc=ulclat,ulclon,lrclat,lrclon is Upper Left Corner Lat, Lon and Lower Right Corner Lat and Lon. --setregion-latlon-extents=lat,lon,latext,lonext,extent_units is Lat, Lon in degrees and Extent in specified units. --hdffilename=file is a MISR Product File. --gridname=grid_name is the name of the grid. --fieldname=field_name is the name of the field. --envifilename=file is the output envi file. Note: The parameter extent_units is a case insensitive string that can be set to one of the following values: 1) degrees, deg, dd for degrees; 2) meters, m for meters; 3) kilometers, km for kilometers; and

Utility	*Usage*
	4) 275m, 275 meters, 1.1km, 1.1 kilometers for pixels of a specified resolution per pixel. Example 1: MtkMisrToEnvi --setregion-path-blockrange=37,45,75 --hdffilename=../Mtk_testdata/in/MISR_AM1_GRP_ELLIPSOID_GM_P037_O029058_AA_F03_0024.hdf --gridname=BlueBand --fieldname="Blue Radiance/RDQI" --envifilename=out Example 2: MtkMisrToEnvi --setregion-latlon-extent=38,-111,3000,300,km --hdffilename=../Mtk_testdata/in/MISR_AM1_GRP_ELLIPSOID_GM_P037_O029058_AA_F03_0024.hdf --gridname=BlueBand --fieldname="Blue Radiance/RDQI" --envifilename=out Example 3: MtkMisrToEnvi --setregion-latlon-extent=38,-111,2000,300,1100m --hdffilename=../Mtk_testdata/in/MISR_AM1_GRP_ELLIPSOID_GM_P037_O029058_AA_F03_0024.hdf --gridname=BlueBand --fieldname="Blue Radiance/RDQI" --envifilename=out Example 4: MtkMisrToEnvi --setregion-ulclrc=51.5,-112,24,-109 --hdffilename=../Mtk_testdata/in/MISR_AM1_GRP_ELLIPSOID_GM_P037_O029058_AA_F03_0024.hdf --gridname=BlueBand --fieldname="Blue Radiance/RDQI" --envifilename=out Example 5: MtkMisrToEnvi --entire-file --hdffilename=../Mtk_testdata/in/MISR_AM1_GRP_ELLIPSOID_GM_P037_O029058_AA_F03_0024.hdf --gridname=BlueBand --fieldname="Blue Radiance/RDQI" --envifilename=out
MtkFileType	Usage: bin/MtkFileType <--help> | --hdffilename=<Input File> Where: --hdffilename=file is a MISR Product File. Example: MtkFileType --hdffilename=../Mtk_testdata/in/MISR_AM1_GRP_ELLIPSOID_GM_P037_O029058_AA_F03_0024.hdf
MtkFillValueGet	Usage: bin/MtkFillValueGet <--help> | --hdffilename=<Input File> --gridname=<Grid Name> --fieldname=<Field Name> Where: --hdffilename=file is a MISR Product File. --gridname=grid_name is the name of the grid. --fieldname=field_name is the name of the field. Example: MtkFillValueGet --hdffilename=../Mtk_testdata/in/MISR_AM1_GRP_ELLIPSOID_GM_P037_O029058_AA_F03_0024.hdf --gridname=BlueBand --fieldname="Blue Radiance/RDQI"