Madrigal documentation - v2.6

Madrigal Database v2.6 Documentation - Contents

Home	Access data

A PDF version of Madrigal 2.6 documentation in a single file

Brief history of Madrigal

Madrigal is a database of ground-based measurements and models of the Earth's upper atmosphere and ionosphere. It is closely related to the Coupling, Energetics and Dynamics of Atmospheric Regions (CEDAR) program, which is devoted to the characterization and understanding of the atmosphere above about 60 km, with emphasis on the various processes that determine the basic structure and composition of the atmosphere, and on the mechanisms that couple different atmospheric regions. Instruments developed or upgraded under CEDAR include interferometers, spectrometers, imagers, lidars and medium, high-frequency and incoherent scatter radars. Models developed with CEDAR support or incorporating CEDAR data include first-principle global circulation models, and empirical models of temperature, density and winds. The success of CEDAR has been due, in large measure, to its ability to encourage collaborative efforts coalescing observations, theory and modeling. The CEDAR community includes about 800 scientists and students from around the world.

From the inception of the CEDAR program in 1988, there has been a great concern among the members of the CEDAR community to make the data collected by the CEDAR instruments easily accessible for joint studies. Consequently, a high priority was placed on establishing a repository for CEDAR data and model results. An incoherent scatter radar database had been established at the National Center for Atmospheric Research (NCAR) in 1985, and this evolved into the CEDAR Database in 1989. By the end of 1997, it had grown to include data from 44 instruments and 16 models. Over 200 users have requested information from the Database.

A central element of the CEDAR Database is a standard data format. This format evolved from the format used by the earlier incoherent scatter database, which in turn evolved from an earlier version of Madrigal developed at the MIT Haystack Observatory in 1980. Haystack continues to maintain and develop Madrigal as an open-source project with community contributions. The basic Madrigal data format is the same as at NCAR, and data files are easily exchanged between the two sites, but Madrigal has a significantly different emphasis. The function of NCAR in the CEDAR Data Base is first of all to archive a comprehensive collection of CEDAR data. Madrigal, on the other hand, has evolved into a robust, World Wide Web based system capable of managing and serving archival and real-time data from a wide variety of CEDAR instruments. The Madrigal database is focused on data that is being actively maintained by the groups that produced the data. With the Madrigal database, the site owner stores only their own data, which they can add to or update at any time. However, because the Madrigal database shares its metadata with all other Madrigal sites, users browsing any Madrigal site can search for data at any other Madrigal site.

CEDAR is now poised to realize the full scientific potential anticipated at its inception. CEDAR's scientific directions and future operational requirements were detailed in 1997 in the CEDAR Phase III Report. The report states that "CEDAR must continue to capitalize on the latest advancements in order to accomplish the best science most effectively. Where practical and advantageous, remote continuous autonomous operation of CEDAR instruments ought to be encouraged. Computer-aided collaborations, CEDAR Database use, and CEDAR program information should be available to the CEDAR community over the World Wide Web. Computer-aided collaborations and real-time access to data and predictive models should be encouraged." Madrigal is a key component to meeting these goals.

Madrigal data are arranged into "experiments", which may contain data files, images, documentation, links, etc. Almost all Millstone Hill Radar experiments since 1976, over 900 experiments in all, as well as many measurements by other instruments, are now available on-line. Each experiment has a unique URL, which serves to map the database contents into the World Wide Web, and is the fundamental mechanism through which the database can be distributed among multiple servers. An key feature of Madrigal is its seamless integration of archival and real-time data. A realtime file on Madrigal is accessed in exactly the same way as any archival file.

Madrigal has been installed at several locations in addition to Millstone Hill, including EISCAT, SRI International, and Jicamarca. The inventories of experiments available at each installation are available to the other installations through a cgi script. Typically each installation retrieves inventories from the other sites and updates a combined experiment list once a day. Users can view the combined inventory and seamlessly retrieve data regardless of where the data is actually stored.

Madrigal is an open source project with a central developer's web site and a central repository. A complete CVS archive of all Madrigal software, including software which is not included with the standard distribution, is available at the developer's web site. There is also an Open Madrigal mailing list and developer's forum. Any group wishing to install Madrigal to distribute their instrument's data is welcome to - see the www.OpenMadrigal.org web site for details. The central repository link is a place where all data from all Madrigal sites is archived. It can also be used to access data, although accessing the local sites is preferred.


Brief overview of Madrigal | Doc home | Madrigal home

Previous: Doc Home **Up:** Doc home **Next:** What's new in Madrigal 2.6?


What's new in Madrigal 2.6? | Doc home | Madrigal home

Previous: Brief History **Up:** Doc home **Next:** Madrigal user's guide

What's new in Madrigal 2.6?

Madrigal 2.6 Release - November 2011

New simple local web user interface

A new easy to use interface to access local Madrigal data was developed by Jicamarca with support from Millstone Hill staff. This interface in now very robust and has been test with data from all Madrigal sites.

HDF5 file download availablity

Users can now download data files as HDF5, in addition to simple column delimited ascii and the more complex CEDAR formats. This is also the result of a Jicamarca/Millstone Hill collaborations.

Users can register interest in experiments or instruments

Users can now register interest in experiments or instruments, and get emails whenever that Madrigal experiment or instrument is updated.

Administrative improvements - can now add external disks to experiments

It is now easy to expand the Madrigal database by mounting additional hard disks to make more room for data.

New metadata - experiment PI's and analyst

Experiments now have direct email links to the experiment PI. These PI's can be updated on an experiment or intrument basis.

Global seach now more robust

The global search UI now generates scripts so you can run the global search right from your computer using either the python remote API, the Matlab remote API, or the IDL remote API.

Madrigal 2.5.2 Release - May 2009

A bug fix release - no new features.

Madrigal 2.5.1 Release - March 2009

This minor revision to Madrigal 2.5 was needed to support using Madrigal as an archiving database. The experiment metadata was expanded to allow archived experiments. Archived experiments are not shared between Madrigal sites, but are visible as local experiments.

Madrigal 2.5 Release - February 2009

Simplification of Web User Interface

Both the Browse for Individual Madrigal Experiments and the Global Madrigal Database Report web interface have been simplified. Searching for instruments under Browse for Individual Madrigal Experiments is now easier through the use of an instrument category selector.

One step file printing available

Under Browse for Individual Madrigal Experiments, users can now choose to print an ascii version of any Madrigal file with one click. With this option they can not include any derived parameters or data filters.

Installation simplified

Autotools is now used to compile all code, significantly reducing the number of parameters in the madrigal.cfg configuration file.

64-bit tested

Madrigal has now been fully tested as a 64-bit application. It is important that all Madrigal installations switch to 64-bit machines by the year 2037, because 32-bit unix cannot handle dates beyond then.

International Reference Ionosphere (IRI) derived parameters now available

All parameters calculated by the International Reference Ionosphere (IRI) model can now be selected as derived parameters.

Additional automatic sharing of metadata added

For administrators: Now when new sites or instruments are added to Madrigal, these metadata files are automatically added to your site.

Experiment level security

Previously, individual Madrigal files could be made public or private. Now entire experiments can be made public, private, or hidden altogether. See the script changeExpStatus.py for details.

Experiment directory naming convention modified

Previous to Madrigal 2.5, all Madrigal experiments had to be stored in directory paths in the form:

```
$MADROOT/experiments/YYYY/<3 letter inst mnemonic>/ddMMMyy[letter],
Example: /opt/madrigal/experiments/1998/mlh/20jan98b.
```

Under this convention, the date of the directory name represented the start date of the experiment, with one letter optionally added. This meant there was a limited number of experiments that could be created for a particular day for a particular experiment. This part of the directory naming convention has been dropped, and now the convention is:

```
$MADROOT/experiments/YYYY/<3 letter inst mnemonic>/*,
Example: /opt/madrigal/experiments/1998/mlh/mlh_exp_234.
```

Madrigal 2.4 Release - February 2006

Simple web UI added

A new web user-interface has been added that allows easy printing and plotting of basic Madrigal data. To make it easy to use, advanced Madrigal features such as derived parameters and filtering of data have been removed.

On-demand plot creation

Madrigal now allows users to create basic scatter plots and pcolor plots versus range or altitude of any measured or derived parameter in a data set.

Logging of user data access

Madrigal now logs user's names, emails, and affiliations whenever data files are directly accessed in a file administrators can access.

Automatic updating of all geophysical data

Madrigal now automatically updates all its internal geophysical files (e.g., Kp, Fof2, Dst, Imf, etc) every time updateMaster is run.

Simple-to-use python module to create and edit Madrigal files

There is now a simple-to-use python module to create and edit Madrigal files.

New administrative scripts to manage Madrigal experiments

Administrators can now add or modify all Madrigal experiments using simple administrative scripts, instead of trying to edit Madrigal metadata files themselves or use the complex genExp script.

Complete documentation rewrite

Madrigal documentation has now been completely rewritten and reorganized into three manuals: one for users, one for administrators, and one for developers.

Automatic graphics conversion

Madrigal will now allow users to select any graphics format they prefer for graphics administrators place in experiments. This feature was contributed by Eiscat.

Update of IGRF/MSIS

The Madrigal derivation engine is now using the IGRF 2010 coefficients, and the MSIS 2000 model.

Limiting of disk space used for global search files

Administrators can now limit the maximum amount of disk space used to store temporary global search files. See the section on editing the madrigal.cfg file in the installation guide.

Madrigal 2.3 Release - March 2004

Remote programming access to Madrigal via web services using any platform

Madrigal now exposes all the information and capabilities it has as web services, which allows easy access to Madrigal from any computer on the internet using any platform (Unix, Windows, Mac, etc). Madrigal's web services are basically cgi scripts with simple output that allows easy parsing of the information. Any language that supports the HTTP standard can then access any Madrigal site. We have written remote API's using python and Matlab, but almost any language could be used. See the section on remote programming access for details of these APIs and the underlying web services.

Note that this approach of remotely accessing Madrigal data has been always possible before by parsing the html output meant to be displayed in a web browser (this general programming method is referred to as "screen scraping"). However, not only is this parsing difficult; but the code often breaks when the user interface is modified in any way. With web services the returned cgi scripts are designed to be both simple to parse and stable.

The web services are not implemented according to the SOAP or XMLRPC standard since not all scripting languages have support for these standards (or for XML parsing). Instead they use the simple approach of returning data requested via a query as a delimited text file. These web services are fully documented here.

Users who want only to write programs to remotely access Madrigal, and not to install a Madrigal server themselves, are now able to download the remote python and Matlab API's from the OpenMadrigal site.

Command-line global search

As an example of remote programming access to Madrigal via web services, an application globalIsprint was written using the python remote API that does a global search of data on any Madrigal site that has installed Madrigal version 2.3. This application is installed as part of Madrigal, and also when the standalone remote python API is installed. It has all the filtering ability of the web-based global search.

Calculate any derivable Madrigal parameter for any time and point(s) in space

By clicking on "Run Models", users can calculate any derived Madrigal parameter (such as magnetic fields, or geophysical parameters) for arbitrary times and ranges of position. Note that this capability is also available as a web service, and through the remote python and Matlab API's.

New derived parameters

- CGM_LAT: **Corrected geomagnetic latitude** (deg)

This parameter gives the location of a point in Corrected geomagnetic latitude. This method uses code developed by Vladimir Papitashvili. For more information on CGM coordinates and this code, click here.

- CGM_LONG: **Corrected geomagnetic longitude** (deg)
  This parameter gives the location of a point in Corrected geomagnetic longitude. This method uses code developed by Vladimir Papitashvili. For more information on CGM coordinates and this code, click here.
- TSYG_EQ_XGSM: **Tsyganenko field GSM XY plane X point** (earth radii)
  This parameter gives the X value in GSM coordinates of where the field line associated with a given input point in space and time crosses the GSM XY plane (the magnetic equatorial plane). GSM stands for Geocentric Solar Magnetospheric System, and its XY plane is the equatorial plane of the earth's magnetic dipole field. The field lines are traced using the Tsyganenko Magnetospheric model, so external effects on the earth's magnetic field such the solar wind are taken into account. This code uses the 2001 Tsyganenko model, which averages solar wind values over the past hour, instead of simply using present values.
- TSYG_EQ_YGSM: **Tsyganenko field GSM XY plane Y point** (earth radii)
  This parameter gives the Y value in GSM coordinates of where the field line associated with a given input point in space and time crosses the GSM XY plane (the magnetic equatorial plane). GSM stands for Geocentric Solar Magnetospheric System, and its XY plane is the equatorial plane of the earth's magnetic dipole field. The field lines are traced using the Tsyganenko Magnetospheric model, so external effects on the earth's magnetic field such the solar wind are taken into account. This code uses the 2001 Tsyganenko model, which averages solar wind values over the past hour, instead of simply using present values.
- TSYG_EQ_XGSM: **Tsyganenko field GSE XY plane X point** (earth radii)
  This parameter gives the X value in GSE coordinates of where the field line associated with a given input point in space and time crosses the GSE XY plane (the equatorial plane). GSE stands for Geocentric Solar Ecliptic System, and its XY plane is the equatorial plane of the earth's rotation. The field lines are traced using the Tsyganenko Magnetospheric model, so external effects on the earth's magnetic field such the solar wind are taken into account. This code uses the 2001 Tsyganenko model, which averages solar wind values over the past hour, instead of simply using present values.
- TSYG_EQ_YGSM: **Tsyganenko field GSE XY plane Y point** (earth radii)
  This parameter gives the Y value in GSE coordinates of where the field line associated with a given input point in space and time crosses the GSE XY plane (the equatorial plane). GSE stands for Geocentric Solar Ecliptic System, and its XY plane is the equatorial plane of the earth's rotation. The field lines are traced using the Tsyganenko Magnetospheric model, so external effects on the earth's magnetic field such the solar wind are taken into account. This code uses the 2001 Tsyganenko model, which averages solar wind values over the past hour, instead of simply using present values.
- BHHMMSS and EHHMMSS: **Start and end time in HHMMSS** (suggested by Mary McCready at SRI)

Bug fixes

The Madrigal C API now no longer aborts when a Cedar file contains cycle marks (Cedar parameter 95) that are not in order. (Reported by Angela Li, SRI)

A problem launching the global search with the python module os.spawnlp was fixed. (Reported by Angela Li, SRI)

Madrigal 2.2 Release - Feb 2003

New derived parameters

- SUNRISE_HOUR - **Ionospheric sunrise** (hour)
  This parameter gives the hour UT that sunrise occurs at that particular point in space that particular day. If that point in space is either in sunlight or in shadow the entire UT day, sunrise_hour will be missing. To find out which, display the Shadow height (SDWHT) parameter. If shadow height is less that the altitude of the point, its in sunlight; if shadow height is greater than the altitude, its in the earth's shadow.
- SUNSET_HOUR - **Ionospheric sunset** (hour)
  This parameter gives the hour UT that sunset occurs at that particular point in space that particular day. If that point in space is either in sunlight or in shadow the entire UT day, sunset_hour will be missing. To find out which, display the Shadow height (SDWHT) parameter. If shadow height is less that the altitude of the point, its in sunlight; if shadow height is greater than the altitude, its in the earth's shadow.
- CONJ_SUNRISE_H - **Magnetic conjugate point sunrise** (hour)
  This parameter gives the hour UT that sunrise occurs at the magnetic conjugate point of the particular point in space that particular day.
- CONJ_SUNSET_H - **Magnetic conjugate point sunset** (hour)
  This parameter gives the hour UT that sunset occurs at the magnetic conjugate point of the particular point in space that particular day.
- SZEN - **Solar zenith angle in measurement vol** (deg)
  This parameter gives the solar zenith angle in degrees. If 0 degrees, the sun is directly overhead. A solar zenith angle of between 90 and 180 degrees does not mean the sun is not visible, due to the finite solid angle of the sun and the altitude the point may be above the earth's surface.
- SZENC - **Conjugate solar zenith angle** (deg)
  This parameter gives the solar zenith angle at the magnetic conjugate point in degrees.
- SDWHT - **Shadow height** (km)
  This parameter gives the height above the earth's surface at which any part of the sun can be seen. It depends only on the time, and on the geodetic latitude and longitude. During the day shadow height will be zero. Since the sun is larger than the earth, the shadow height is always finite. If shadow height is less that the altitude of a given point in space, its in sunlight; if shadow height is greater than the altitude, its in the earth's shadow.
- MAGCONJSDWHT - **Magnetic conjugate shadow height** (km)
  This parameter gives the height above the earth's surface at the magnetic conjugate point's latitude and longitude at which any part of the sun can be seen.
- **10 Interplanetary Magnetic Field parameters**
  Includes field strength in GSM or GSE coordinates, solar wind plasma density, speed, and measuring satellite id. Click on any parameter to see the definition of the two coordinate systems.

Filtering using any parameter

- There are now also free-form filters at the end of the filter section, which allow you to set up filters based on any single parameter or on two parameters either added, subtracted, multiplied, or divided together. For example, you can now filter on Ti, the

ratio Ti/dTi, or gdalt-sdwht (which is positive if the point is in sunlight). See the tutorial for more details.

Better help understanding what each parameter means

- Complex parameters now have full html descriptions accessible from the isprint page. Just click on the parameter name and you'll see the short description. For more complex parameters you'll also see a link to a more detailed explanation.

Improved data output

- If you select only 1D parameters, or derived parameters that depend only on other 1D parameters, isprint will only print a single line per record, making it easier to read.
- All filters used are printed at the beginning of the report. Trivial filters that don't exclude data (such as elevation from 0 to 90 degrees) are ignored.

Better consistency with Cedar standard

- All units are now consistent with the Cedar standard (when displaying Cedar parameters).
- The special Cedar values "missing", "assumed", and "known bad" are differentiated in isprint output, and not all lumped together as "missing" as before.
- Unknown parameter codes displayed with a scale factor of 1.0.

New derived parameters are simple to add

- The isprint web page is now based on the madc library, and has been designed to make it extremely simple to add new derived parameters. See the madc API documentation for details.

	What's new in Madrigal 2.5?	Doc home	Madrigal home

Previous: Brief History **Up:** Doc home **Next:** Madrigal user's guide

	Madrigal user's guide (How do I access Madrigal data?)	Doc home	Madrigal home

Previous: What's new **Up:** Doc home **Next:** Web interface tutorial TOC

Madrigal user's guide (How do I access Madrigal data?)

There are two basic ways to access Madrigal data - through a web browser, or through a programming language such as Matlab or Python. This user's guide consists of a tutorial on using the web interface, and a tutorial and a reference manual for remote data access. The last section also documents some command line interfaces into Madrigal, some of which only run locally on the Madrigal server, and some of which can be run remotely.

Madrigal web tutorial - Table of contents

What is Madrigal?

Madrigal is a robust, World Wide Web based system capable of managing and serving archival and real-time data, in a variety of formats, from a wide range of ground-based instruments. Madrigal is installed at a number of sites around the world. Each site controls their own Madrigal installation, and can add or upgrade their data from their instrument(s) on their site at any time. In addition to storing data in the standard Madrigal/Cedar format, each site can also add additional web-based documentation such as plots or descriptive notes to individual experiments they run.

However, Madrigal also defines standard metadata that all Madrigal sites share. This makes it possible for each Madrigal site to know about all the experiments on all the other sites. From a user's point of view, each Madrigal site allows them to search for data at all the Madrigal sites at once, and simply follow links to the Madrigal site that has the data they are interested in.

Madrigal was originally developed to serve the needs of the incoherent scatter radar community. There are Madrigal sites containing incoherent scatter radar data at Millstone Hill, USA, EISCAT, Sweden, Arecibo, Puerto Rico, SRI International, USA, Cornell University, USA, Jicamarca, Peru, The Institute of Solar-Terrestrial Physics, Russia, and Wuhan Ionospheric Observatory, the Chinese Academy of Sciences. In addition, many of these sites also store data from optical instruments on Madrigal, and the Millstone site holds total electron content data derived from the world-wide network of GPS receivers. Madrigal is designed to hold data from any ground-based instrument that studies atmospheric science.

Madrigal is often compared to the Cedar database. The Cedar database grew out of the existing Madrigal database in the late 1980's. The Cedar database is a central repository and archive for Cedar data, and shares the same data format as Madrigal. Since the Cedar database is a community resource, it guarantees data availability even when individual instruments and groups stop being active. The Madrigal database is focused on data that is being actively maintained by the groups that produced the data. Also, unlike the Cedar database, Madrigal organizes its data by experiments. This allows data to presented over the web in ways specifically suited to individual experiments. The organization of Madrigal data will be reviewed in the next section of this tutorial.

Madrigal also differs from the present Cedar database in that it has derivation engine built into it. This means that you can request many parameters that are not directly recorded in the data file, such as geophysical parameters, magnetic field parameters, model data, and alternative coordinate system parameters. In the web interface, the only difference between measured parameters and derived parameters will be that measured parameters are shown in bold font.

There are two ways to access Madrigal data - through a web browser, or through a programming language for which an application programming interface (API) has been written. This tutorial covers accessing Madrigal data via a web browser. There is also a tutorial covering accessing Madrigal data via API's.

Madrigal is an open source project with a central developer's web site and a central repository. A complete CVS archive of all Madrigal software, including software which is not included with the standard distribution, is available at the developer's web site. There is also an Open Madrigal mailing list and developer's forum. Any group wishing to install Madrigal to distribute their instrument's data is welcome to - see the www.OpenMadrigal.org web site for details. The central repository link is a place where all data from all Madrigal sites is archived. It can also be used to access data, although accessing the local sites is preferred.


What is Madrigal? | Doc home | Madrigal home

Previous: Web tutorial - TOC **Up:** Web tutorial - TOC **Next:** Data Organization

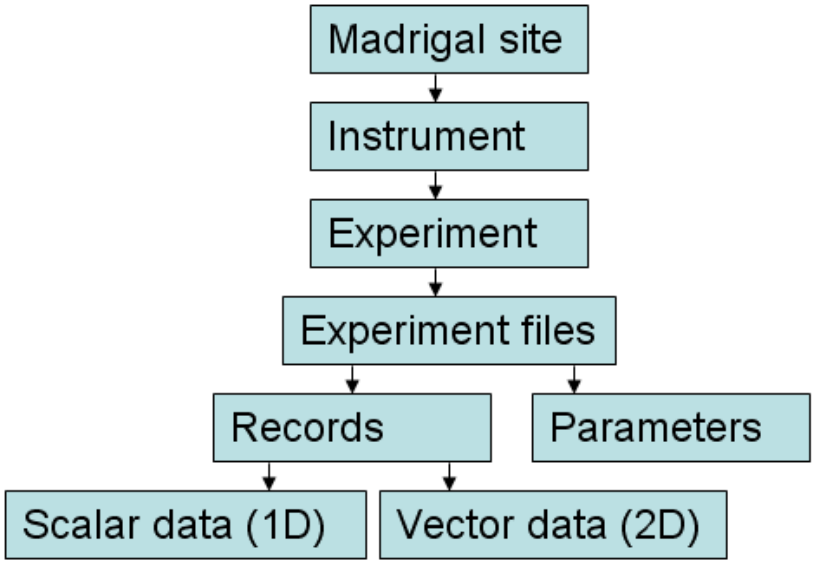
	How does Madrigal organize data?	Doc home	Madrigal home

Previous: What is Madrigal? **Up:** Web Tutorial - TOC **Next:** Access data through the web

How does Madrigal organize data?

Understanding the organization of Madrigal data will help you understand the logic behind the way data is accessed through the web. In this section of the tutorial, a brief overview of the organization of Madrigal data is given, all the way from the highest level (a Madrigal site holding data from one group's instrument(s)) to the lowest level (a single measured value).

A high level view of the Madrigal data model.



Madrigal Site

The highest level of Madrigal is a Madrigal site. A Madrigal site is one particular web site controlled by one particular group, that holds all their own data. At the moment, there are Madrigal sites at Millstone Hill, USA, EISCAT, Sweden, Arecibo, Puerto Rico, SRI International, USA, Cornell University, USA, Jicamarca, Peru, The Institute of Solar-Terrestrial Physics, Russia, and Wuhan Ionospheric Observatory, the Chinese Academy of Sciences. While each Madrigal site stores their own data locally, they also share metadata with all the other sites. This makes it possible for you to search for data at all the Madrigal sites at once no matter which site you visit, and simply follow links to the Madrigal site that has the data you are interested in.

Instrument

The next layer of the Madrigal data model is the instrument. All data in Madrigal is associated with one and only one instrument. Any given Madrigal site will hold data from one or more instruments. Since Madrigal focuses on ground-based instruments, most instruments have a particular location associated with them. However, some Madrigal data is based on measurements from multiple instruments, and so have no particular location. Some examples are "EISCAT Scientific Association IS Radars" which combine data from the multiple EISCAT radars, and "World-wide GPS Receiver Network", which consists of over a thousand individual GPS receivers distributed around the globe.

Experiment

All the data from a given instrument is organized into experiments. An experiment consists of data from a single instrument covering a limited period of time, and, as a rule, is meant to address a particular scientific goal. Madrigal makes the assumption that instruments may be run in different modes, and so the data generated may vary from one experiment to another. By organizing one instrument's data into experiments,

the purpose and limitations of each experiment can be made clearer. In Madrigal, you can navigate to a page that contains a particular experiment for a given instrument. This page may contain notes more fully describing the unique features of this experiment, or may contain plots of the data customized to the type of experiment. This level of organization is not presently maintained in the Cedar database.

For simpler instruments that run constantly in the same mode, it is also possible that all data is put in one experiment. For example, the "DST index" instrument consists of a single experiment that is repeatedly updated.

Experiment Files

The data from a given experiment is stored in one or more experiment files. There are two reasons there may be more than one file for a given experiment. The first is that the experimental data may be analyzed in more than one way, leading to files with different sets of measured parameters. The second is that older, historical files can be kept on-line for reference purposes. By default, you will only access the most recent, default file through the web, unless you choose "Show History Files" when navigating Madrigal.

The format of these files is the Cedar database format, but this is not important since you download from the web ascii format. Note that once to choose a particular file, you will be directed to the Madrigal site that has that file. Madrigal does not share experiment files between site; only higher level metadata about those files.

File parameters

Any given file is made up a series of records holding measured parameters. Note that based on which parameters are in the file, Madrigal will automatically derive a large number of other parameters such as Kp and Magnetic field strength that aren't in the file itself. In the web browser, measured parameters are shown in bold, derived parameters in normal font.

File data

The bottom level of the Madrigal data model is of course the data itself. A Madrigal file is made up of a series of records, each with a start and stop time, representing the integration period of measurement (Madrigal tries to enforce the idea that all measurements take a finite time, but sometimes the start time = the stop time). To get data from a file, simply specify the parameters you want (and optionally, any filters to apply to the data). More details are given later in this tutorial.

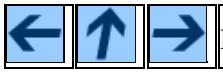
Each Madrigal record has two parts - scalar parameters and vector parameters. For historical reasons these two parts are sometimes called one-dimensional and two-dimensional parameters. Scalar parameters are easy to explain - each scalar parameters has one measurement per record. An example might be the azimuth of a radar making a measurement. Vector parameters have multiple values in a given record. The Cedar file format specifies that all vector parameters must have the same number of measurements. One of more of the vector parameters represent the independent spatial variable(s). For radars this variable is typically range, but latitude, longitude, and altitude could just as easily be used as the three independent spatial variables. The dependent vector variables must all have the same length as the independent variable(s). The independent parameter should never represent time, since the Cedar format specifies that that one record should should cover one period of time.

For example, a radar might store azimuth and elevation as scalar parameters, and range as the independent vector variable. If the electron density and ion temperature are dependent vector variables, and there are ten range measurements, then there must be ten measurements of electron density, and ten measurements of ion temperature. If at certain ranges it is impossible to determine the ion temperature, the Cedar format defines a

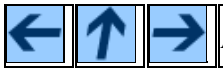
special value to represent missing data to fill the gap.

The Cedar file format defines the physical meaning of almost every parameter to be found in a Cedar file. The only exceptions are parameters defined by individual groups. Any parameter found in a Cedar file that is not defined in the Cedar file format should be fully defined in the header record of the file. See the experiment page for a description of how to view a Cedar file's header record.

Each Cedar parameter can also have an associated error value. This error value can have the special values "missing", "assumed", or "known bad". If an error parameter is "assumed", the implication is that the measured value itself is assumed, and does not represent a measured value. If the error value is "known bad", the measured data is known to have a problem.


How does Madrigal organize data? | Doc home | Madrigal home

Previous: What is Madrigal? **Up:** Web Tutorial - TOC **Next:** Access data through the web


Accessing data through the web | Doc home | Madrigal home

Previous: Data Organization **Up:** Web Tutorial - TOC **Next:** Simple Madrigal data access

Accessing Madrigal data through the web

All Madrigal sites have a similar homepage:

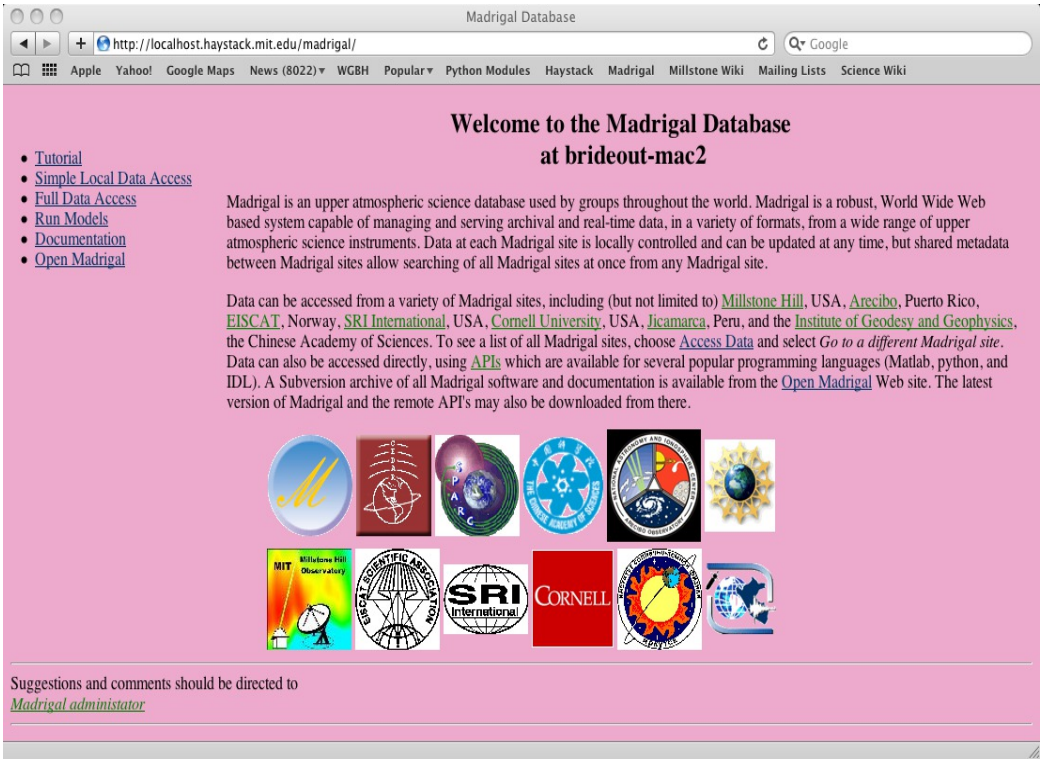
The Madrigal homepage contains a number of links on the left hand side.

Click Simple Local Data Access for simple access to local Madrigal Data.

Click Full Data Access to access the all web interfaces, including searching all Madrigal sites.

Other links on this page include:

- Tutorial: A link to this tutorial on using the web interface
- Run Models: A link to using the Ionospheric models and to using the Madrigal derivation engine to calculate any parameter at any point in time and space.
- Documentation: A link to the main documentation page
- A link to the OpenMadrigal site .



Four methods for accessing Madrigal data

There are four ways to access Madrigal data.

- For a simple-to-use interface for downloading and printing local data without the flexibility of the next three, choose *Simple Local Data Access*
- To look at the data from a particular Madrigal experiment **from any Madrigal site**, choose *Full Data Access*, then *Browse for Individual Madrigal Experiments* .
- To get data in ascii format from a group of Madrigal experiments from the local Madrigal site all at once, choose *Full Data Access*, then *Global Madrigal Search* .
- To plot data from one or more instruments and/or experiments, choose *Full Data Access*, then *Plot Data from Instruments* .

Simple Local Data Access allows users of Madrigal to print and download data easily. In order to make it easy to use, a number of Madrigal's capabilities are not available, including the ability to choose which parameters to print, the ability to display derived parameters, and the ability to filter data.

This tutorial continues with the simple local data access tutorial. To jump ahead to one of the other three more full-featured ways to access Madrigal data, go to Browse for Individual Madrigal Experiments, Global Madrigal Search or Plot Data from Instruments.

Accessing data through the web | Doc home | Madrigal home

Previous: Data Organization **Up:** Web Tutorial - TOC **Next:** Simple Madrigal data access

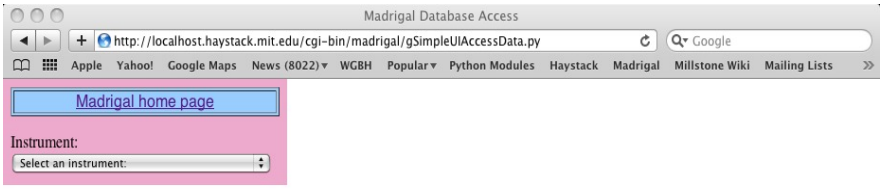
Simple Madrigal Data Access | Doc home | Madrigal home

Previous: Access data through the web **Up:** Web tutorial - TOC **Next:** Browse for experiments

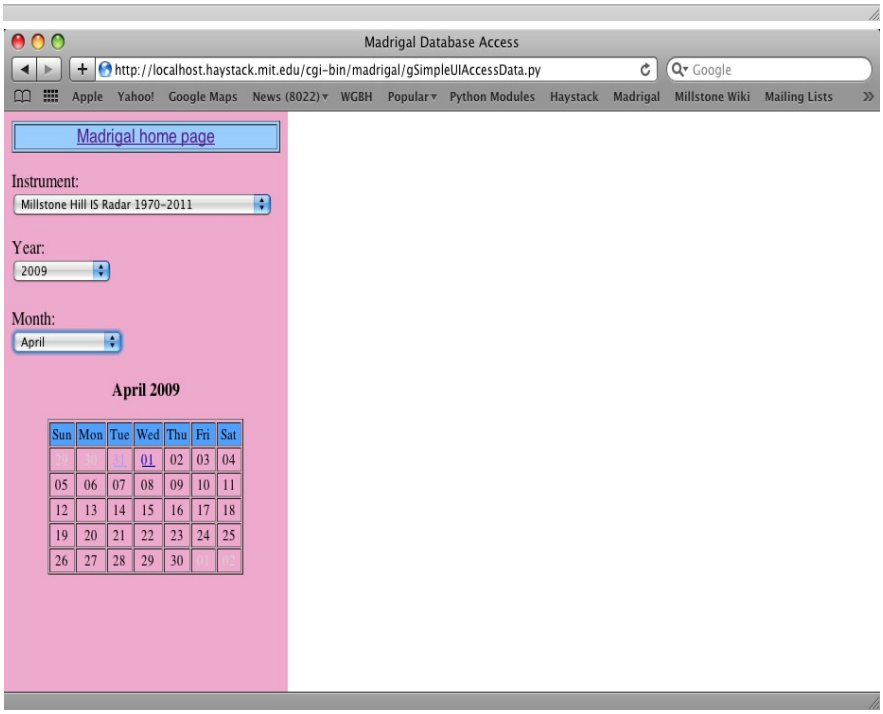
Simple Local Data Access

The simple local data access link allows users of Madrigal to print and download data local Madrigal data easily. In order to make it easy to use, a number of Madrigal's capabilities are not available, including the ability to choose which parameters to print, the ability to display derived parameters, and the ability to filter data. Also, this interface just allows you to browse the local data on this Madrigal server. Use the Browse for Individual Madrigal Experiments interface to search all Madrigal sites for data.

The first step in using the Simple local data access interface is to choose the instrument you want to see data or plots from. This page will list all the instruments for which there is data on the local Madrigal server. You simply click on the instrument you are interested in.

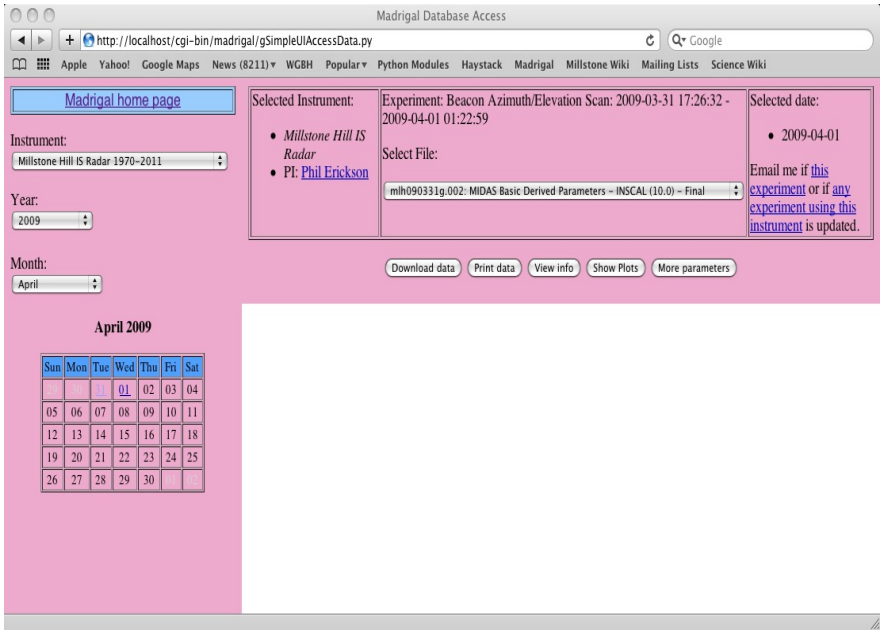


In the frames below you'll see when there is data available for the instrument you selected. You'll choose the year, the month, and then the day.



For the given day, there may be one or more experiments and one or more associated files. You will need to choose if there are more than one. When a file is selected, there are five buttons to choose from:

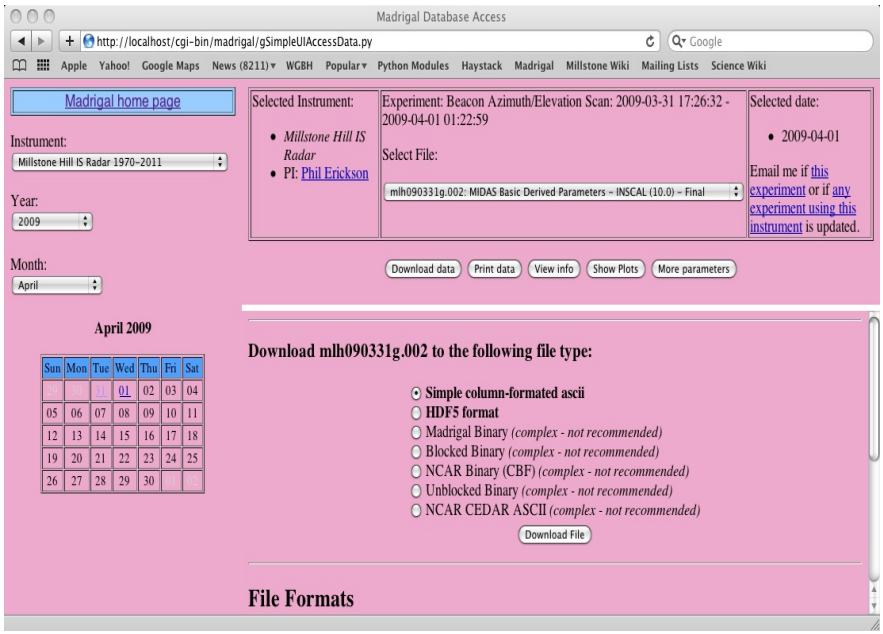
1. Download data
2. Print data
3. View info (show text info in file)
4. Show plots (show plots associated with this experiment)
5. More parameters (show full UI that allows parameter selection, filters, and derived parameters).



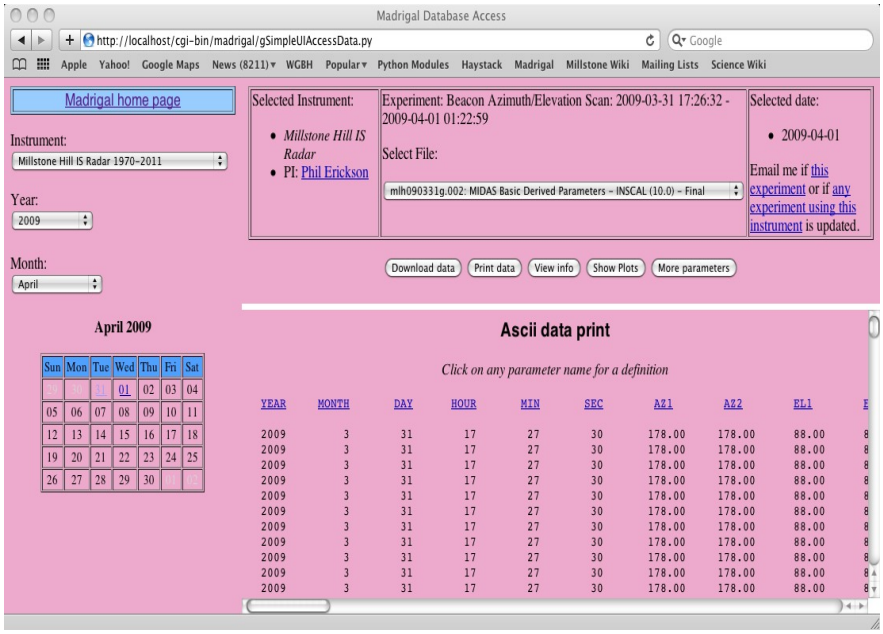
Some of these buttons may be disabled. For example, if there are no plots created for this experiment, *Show plots* will be disabled.

You can also register to receive email updates for this experiment or instrument here.

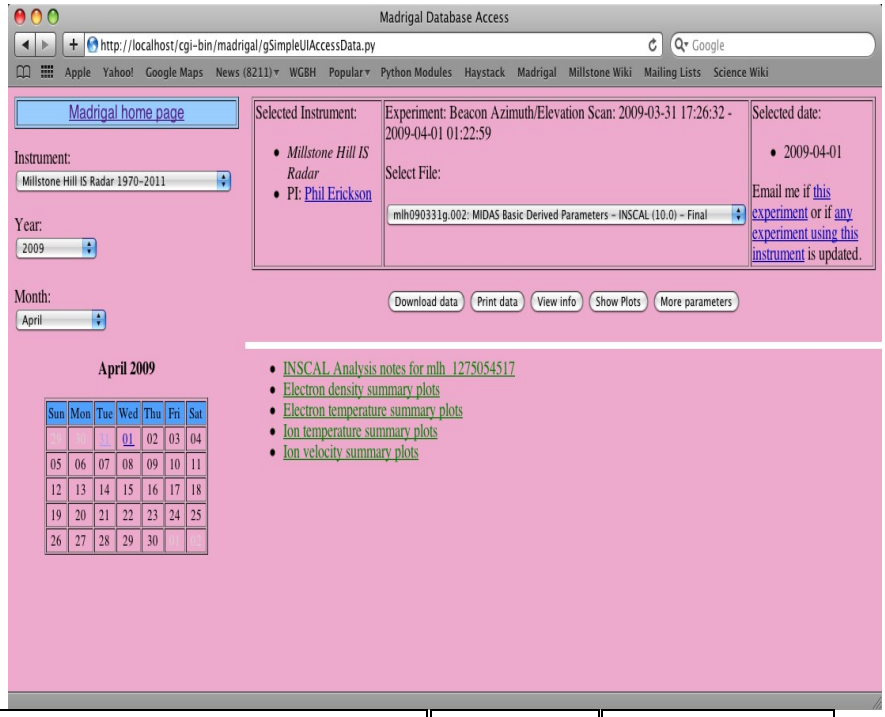
Download data lets you download the data file. The simple column delimited text format or HDF5 format are recommended.



Print data will print all the data in that file in your browser in a simple column delimited text format. All the parameter names are hyperlinks, so you can click on any name to see its full definition.



If you choose *Show plots*, you'll see links to plots created when the data was loaded into Madrigal.



	Simple Madrigal Data Access	Doc home	Madrigal home

Previous: Access data through the web **Up:** Web tutorial - TOC **Next:** Browse for experiments

	Browsing for individual Madrigal experiments	Doc home	Madrigal home

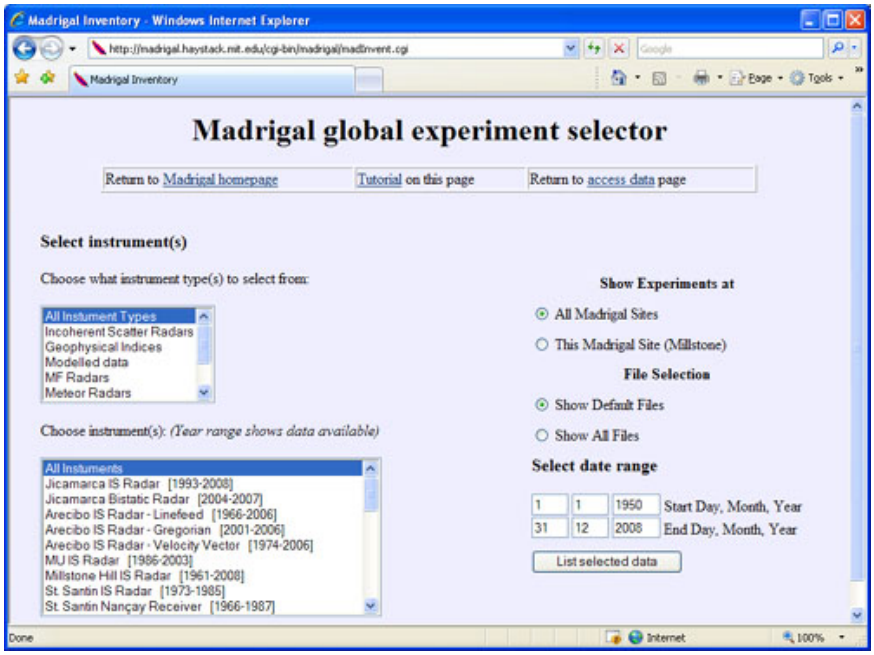
Previous: Simple Madrigal data access **Up:** Access data through the web **Next:** Experiment page

Browsing for individual Madrigal experiments

The second choice from the access data page is *Browse for individual Madrigal experiments*. By selecting this option, you can use the full power of Madrigal to view individual Madrigal experiments. In this part of the web interface, you will be able to display both measured and derived parameters, as well as applying a wide variety of filters to the data. *Unlike the other three choices, this selection allows you to search for data from **all** Madrigal sites, not just the local one.*

The Madrigal experiment selector page is used to select a subset of all possible Madrigal experiments from all Madrigal sites at once.

On the left you can use the top selection to limit what type on instruments you want to select from. You can choose more than one type. The selection box below allows you to select which particular instruments you want The years for which data is available is shown next to each instrument..

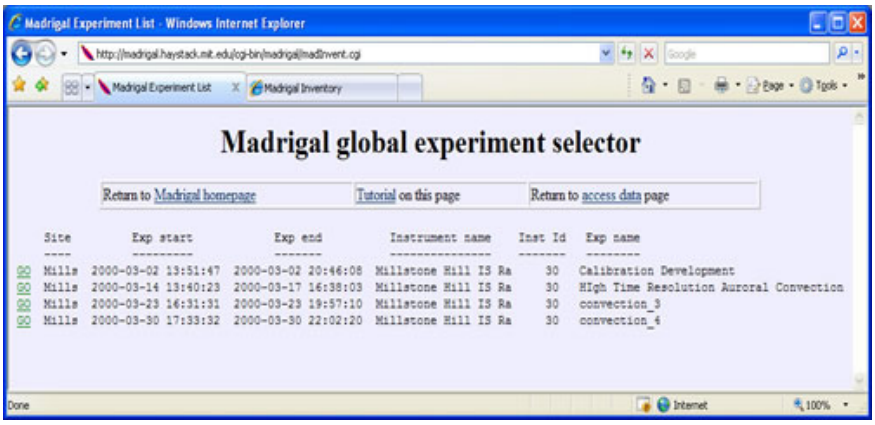


You can also choose to search All Madrigal sites (the default), or just the local Madrigal site. The title will change from *Madrigal global experiment selector* to *Madrigal local experiment selector*. The available instrument types and instruments will also change.

By default, the date range is set to show all experiments. You can limit the date range if you wish. If you limit it so that no experiments are found, you will instead get a page cataloging ***all*** experiments from the instrument(s) you selected, so that you can adjust your date selection appropriately.

As an example, let's say you wanted an experiment at Millstone Hill in March 2000. By selecting all the Millstone Hill IS Radar, and then setting the date filters for just March 2000, you'd get the following page, no matter which Madrigal database you were using.

Each of the four experiments found is described in one row. The Go link on the side will bring you to the experiment page on the correct Madrigal site.



Browsing for individual Madrigal experiments	Doc home	Madrigal home

Previous: Simple Madrigal data access **Up:** Access data through the web **Next:** Experiment page

The Madrigal experiment page	Doc home	Madrigal home

Previous: Browse for experiments **Up:** Access data through the web **Next:** Download file

The Madrigal experiment page

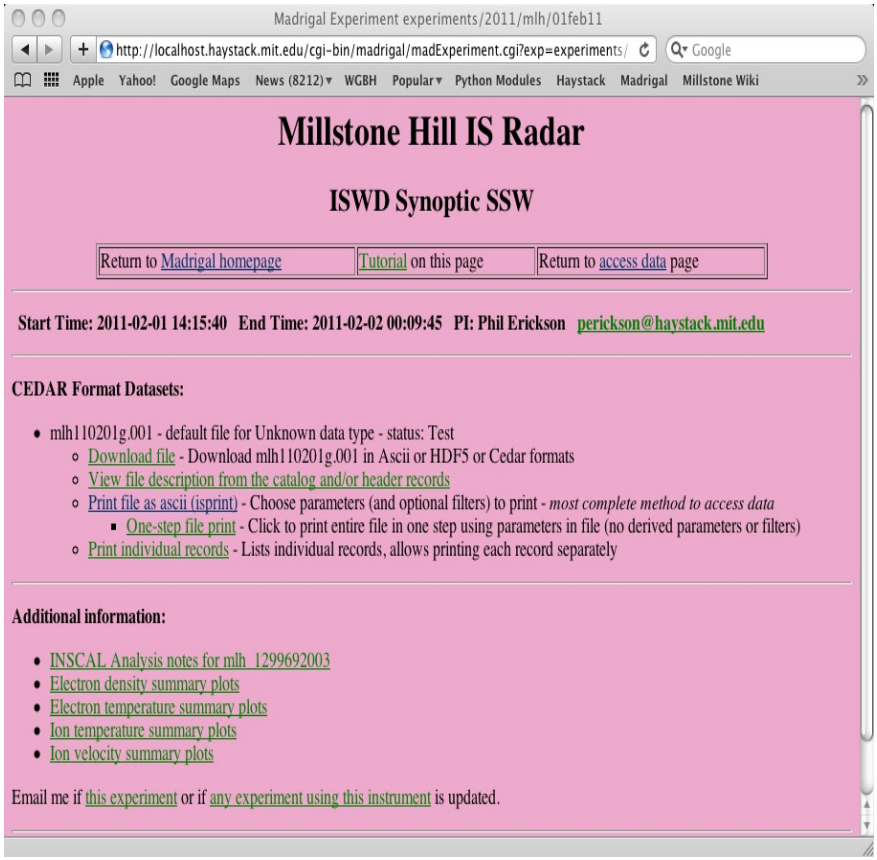
The Madrigal experiment page is the central page that describes a particular Madrigal experiment. It exposes information about the individual Madrigal files contained in an experiment, and also any additional links to auxiliary pages or plots describing the experiment.

For any given Cedar file in the Madrigal experiment, there are a number of ways of viewing the data in the file:

- Download file
- If the file has any catalog or header records, they'll be a link to view them.
- Print file as ascii (isprint) - the standard, full-featured method to access Madrigal data
- One-step file print - a quick way to print just the data using just the parameters in the file itself without filtering.
- Print individual records

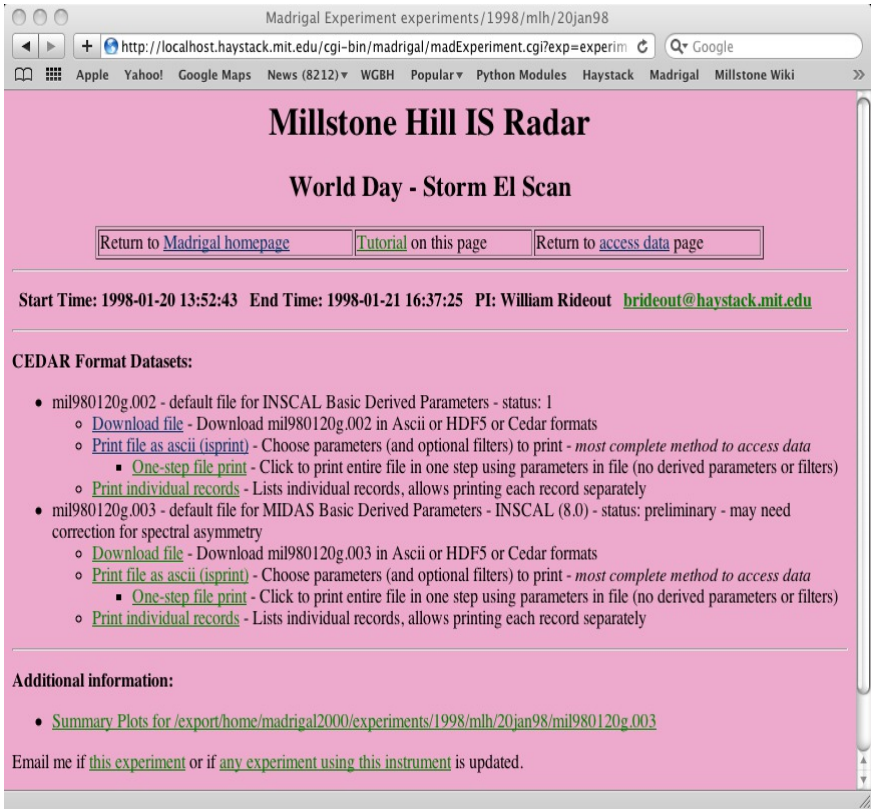
The top of the experiment page shows the instrument name and the experiment name, along with the start and end times of the experiment. There is also a link to the PI's email if you want to ask a question or ask permission to use the data.

Next the individual Cedar-format files are listed. These will normally be the default files unless you selected *Show All Files* while browsing for Madrigal experiments - in that case you may also see historical or alternative files. On the same line as the name of the file is listed the type of data analysis and a status description. For the first file, the type of data analysis is *MIDAS Basic Derived Parameters* and the status description is *Single pulse data*.



Experiment page with multiple files shown.

Each Cedar file will have its own Download file link. This will allow you to download the file itself. It is recommended you choose either simple column-delimited ascii or HDF5 format. The Cedar format files can also be downloaded, but are far more complex.



If the file has any catalog or header records, they'll be a link View file description from the catalog and/or header records to view them.

Next, each Cedar file will have two links,Print file to ascii (isprint)and *One-step file print*. Print file to ascii (isprint) is the most fully-featured way to get ascii listings of the data in any given Cedar file. It will allow you to view both measured and derivable parameters. The data can be filtered in any number of ways. If you will be selecting the same parameters and applying the same filters to more than one file, you can save those settings as a filter. See the Print file to ascii (isprint) on the next page of this tutorial for more information. The One-step file print link does exactly that - it immediately prints out all the data in the file, using only the parameters in the file, and without any data filtering.

Next, each Cedar file will have its own Print individual records link. This interface is meant for a "quick look" at the data in the file or for quickly looking at one individual measurement. It consists of a series of links, one for each record in the file. When clicked, each individual record link will dump that record's data in a standard format. This interface will not show derived parameters; see Print file to ascii (isprint) for that more fully-featured way to view the file. If the site has provided plots for each individual record, you will see links next to each record in the File Summary page. On the Madrigal experiment page, you will see the phrase "*individual record plots available*" if those plots are available. (Madrigal administrators - see the administrator's guide for how to set this up). See the Print individual records page for more information on this interface.

Each Madrigal site can add additional information about any given experiment in any web-accessible format. Links to these additional resources will be found under the *Additional information* heading. In the example shown above, there is a link to *Analysis of ISR Long Duration Experiments*. (Madrigal administrators - see the administrator's guide for how to set this up).

You can also register to receive email updates for this experiment or instrument here.

Finally, anyone who wishes to add notes about this experiment can do so by clicking *Add to these notes*. If, for example, you as a user of the data noted some interesting or questionable data, you can add a note about it. Your note would then show up under the *Notes* heading the next time anyone visited this experiment page.

			The Madrigal experiment page	Doc home	Madrigal home

Previous: Browse for experiments **Up:** Access data through the web **Next:** Download file

	Downloading Madrigal Files	Doc home	Madrigal home

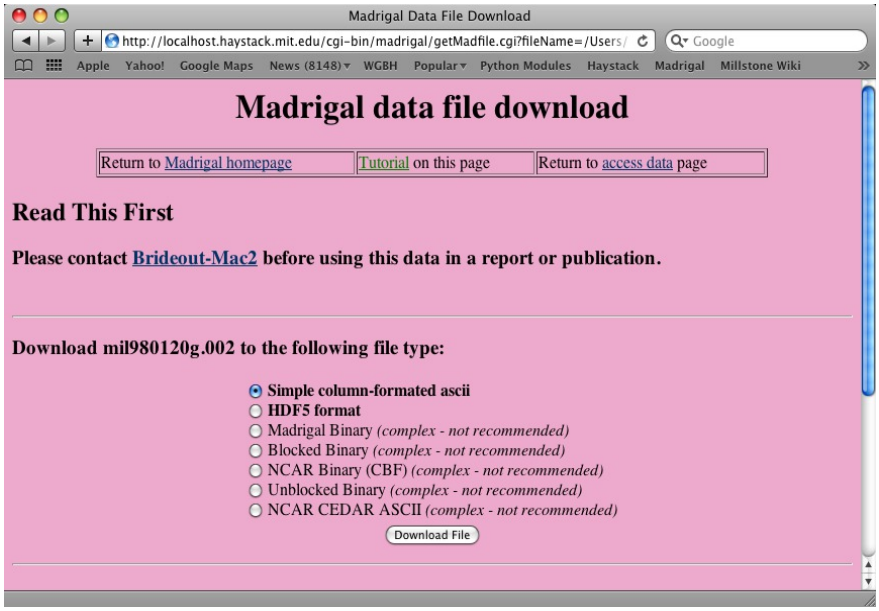
Previous: Experiment page **Up:** Access data through the web **Next:** Data browser

Downloading Madrigal Files

This page allows you to download the Madrigal file directly.

The first part of this page specifies the rules of the road you are expected to follow when using this data.

There are two easy to use formats to choose from: simple column delimited ascii, or HDF5. The more complex CEDAR format files are not recommended unless you already have code to read them.



Downloading Madrigal Files	Doc home	Madrigal home

Previous: Experiment page **Up:** Access data through the web **Next:** Data browser

Print file as ascii	Doc home	Madrigal home

Previous: Download file **Up:** Access data through the web **Next:** Print individual records

Print file as ascii (isprint)

The Madrigal print file page is the main web interface for accessing data from individual Cedar files. It will allow you to view in ascii format both measured and derivable parameters. The data can be filtered in any number of ways. If you will be selecting the same parameters and applying the same filters to more than one file, you can save those settings as a filter.

This page was originally called isprint (for **i**ncoherent **s**catter **print**). Madrigal is now written for any ground-based observation of the upper atmosphere, so this name is simply a historical artifact.

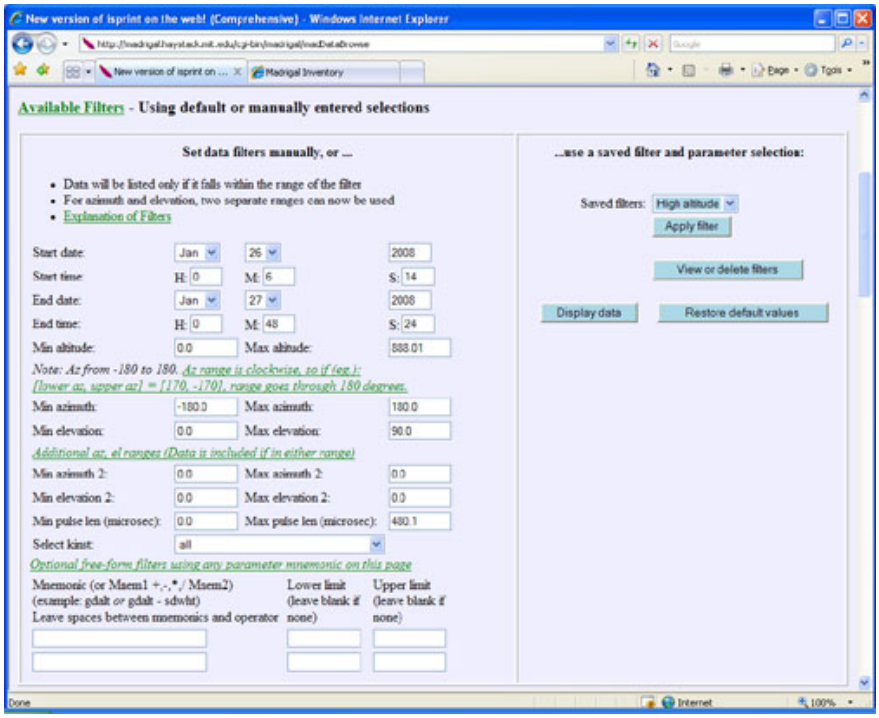
In the data organization section of this tutorial, we discussed that Madrigal stored data as a series of records, with each record representing a single period of time. Within each record were stored scalar measurements (such as a radar azimuth) and vector measurements (such as radar range or electron temperature). The Madrigal print file treats all data as vector data, so if you request both azimuth and range, the identical azimuth data will be repeated for each individual range in a record.

Filters also act on each vector measurement individually. If your filter specifies a limited altitude coverage, some ranges from a given record may appear and others may be removed. Almost all the filters use the idea of a range: if a given parameter is outside of the set range, it is excluded. To use only a minimum or maximum value, simply set the other end of the range to be blank. For example, to see all data with an elevation over 20 degrees, set the minimum to 20 and leave the maximum blank.

The print file page has three sections:

- The filter selection
- The parameter selection
- The output format selection

Filter selection



In this section of the print file, you can set up filters to remove data you do not to appear in the output. When the page is first loaded, all the filter parameters are set to include all the data.

The time filter always appears. The following filters appear only if they make sense for the given file:

- Altitude
- Azimuth
- Elevation
- Pulse length

If data from more than one instrument appears in the file, you can also filter by instrument (this is only the case for instruments made up of combinations of other instruments, such as *Millstone Hill IS Radar*).

Also available is a way for you to filter data based on any parameter or parameters listed on in the parameters section of this page. Here's an example:

Mnemonic (or Mnem1 +,-,*,/ Mnem2) (example: gdalt *or* gdalt - sdwht) Leave spaces between mnemonics and operator	Lower limit (leave blank if none)	Upper limit (leave blank if none)

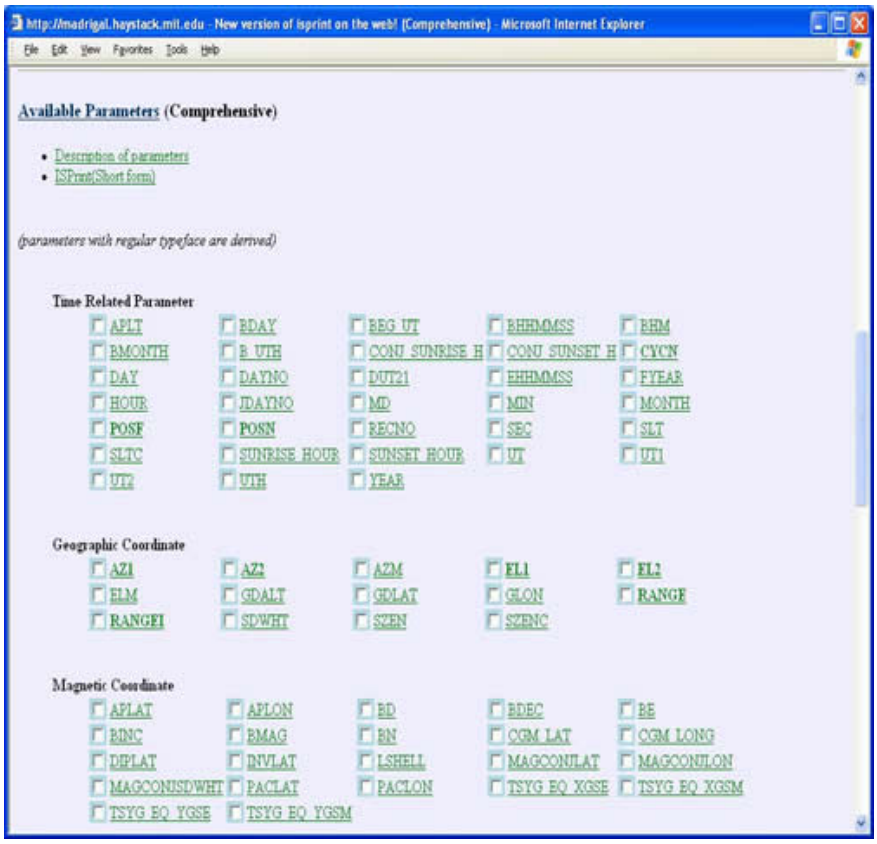
The first filter implies "gdalt - sdwht" must be greater than 0.0. Since sdwht is shadow height (the distance above any point on the earth where the sun is first visible), this filter implies that only data in direct sunlight will be displayed. The second filter says that BMAG (the magnitude of the magnetic field) must be between 0 and 3e-5 Tesla. Note that the meaning and units of any parameter are available by clicking on them.

Note that the filter can be based on any single parameter (such as BMAG above), or any two parameters either added, subtracted, multiplied, or divided (as in the "gdalt - sdwht" example above). Note that if the parameter you enter is missing or cannot be calculated, it will be rejected no matter what the range is, since missing data is never in any range.

In general, all these separate filters are and'ed together, so if data is excluded from any filter it is excluded. There are two exceptions: additional azimuth range, and additional elevation range. The purpose of these two new filters is to provide a little more flexibility in filtering azimuth or elevation than can be provided by a simple range. For example, if you wanted to select azimuth values between 270 and 30 degrees, the simple range approach would not work, since the range does not go through zero. The additional azimuth range added in this release allows this now to be done. The two azimuth ranges are or'ed together: a record is selected if its azimuth is in either range. Elevation works the same way. By default, the additional azimuth and elevation ranges are set to 0 to 0 degrees, so they have no effect on the filter and can be ignored if not needed.

On the right hand side of the filter section are buttons to allow you to apply existing filters, or to save the ones you've created. If you want to reuse the filters and parameters you selected for a different Madrigal experiment, click on the Login button to create a username and password. This username and password will be saved as a cookie on your browser, so you will not need to login each time you use this Madrigal site. You can then use the buttons on the right side of the filter section to save your filter settings and give it a name. When you save the settings, you will decide whether you want your filter to be public (everyone can use it, but not change or delete it) or private (only you can see it or change it). The filter you save will include both your filter settings and the parameters you selected. Only the date filters are not applied when you apply an existing filter, since two different experiments rarely have the same dates.

Parameter selection

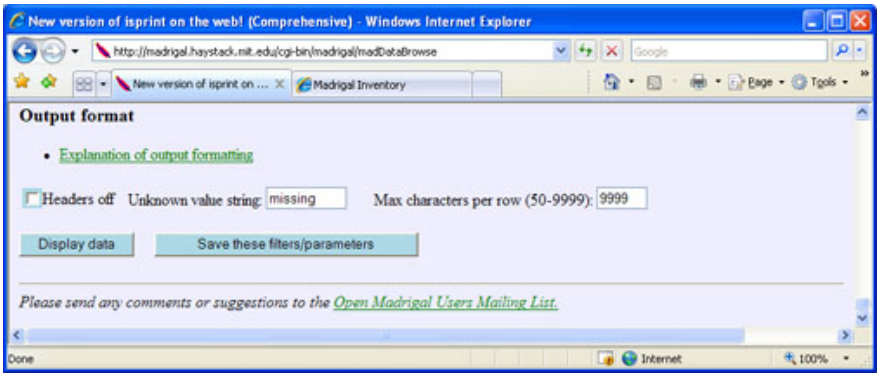


Parameter selection is in this section of the print file, you choose which parameters you want to display. Parameters are grouped into categories. Bold-faced parameters are the measured ones, while the plain text parameters are derived.

Click on any parameter to see its definition, and the units it will be displayed in.

For some parameters, the definition will include a link to even more information about the parameter.

Output format



Finally, in the Output format section, you can:

- Turn headers on or off. Headers are placed at the top of each record by default.
- Change the string that indicates missing values.
- Set the number of characters per row.

Hitting *Display Data* will print the data.

Previous: Download file **Up:** Access data through the web **Next:** Print individual records

	Print individual records	Doc home	Madrigal home

Previous: Print file as ascii **Up:** Access data through the web **Next:** Global search

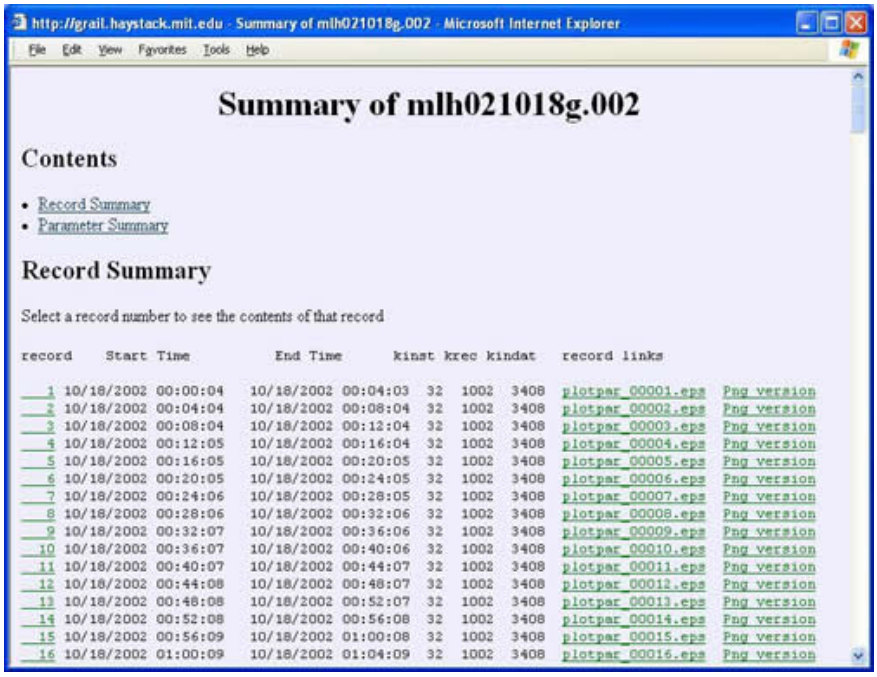
Print individual records page

The print individual records page is meant for looking at individual records in one particular Cedar file. It consists of a series of lines, one for each record in the file. For a fully-featured way of downloading the data that includes both measured and derived parameters, see the Print file as ascii section.

If you click on the record number link, you will see a ascii version of the data in that particular record.

For this file, plots are available for each record. Click on the links to the right of each record to see that record's plot.

At the end of the page is a *Parameter Summary* of all the measured parameters found in the file.



	Print individual records	Doc home	Madrigal home

Previous: Print file as ascii **Up:** Access data through the web **Next:** Global search

	Global Madrigal database report	Doc home	Madrigal home

Previous: Print individual records **Up:** Access data through the web **Next:** Plot data from instruments

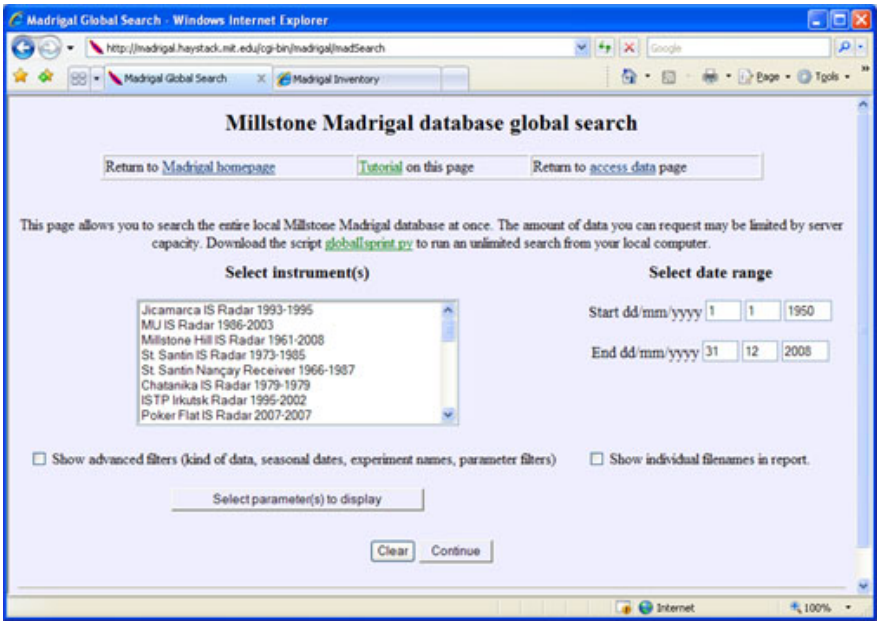
Global Madrigal database report

The Global Madrigal database report is designed to allow searches of the entire local Madrigal Database at once, rather than on only one experiment at a time.These searches can be based on characteristics of the experiment, such as instrument, date, or experiment name, and in addition, can depend on the actual data, such as, whether the ion temperature was above a certain level. This report outputs data in ascii format. This tutorial describes the various way you can select what part of the Madrigal database you want to download. Do do a global search for another Madrigal site, you need to navigate to that site first - this web page only returns data from the local Madrigal site.

This page is meant to be used is conjunction with one of the remote API's: either python, Matlab, or IDL. It will generate commands that you can run from your own local computer using one of these API's to run this search. It is also possible to run the search on the Madrigal server itself, but this is less robust because the amount of data you can request may be limited by server capacity.

Select instrument(s)

This filter allows you to select the instruments you are interested in. You have the option to select more than one instrument.



Experiment dates

This lets you enter the time period you would like to search. It defaults to Jan 1, 1950 to the end of the present year

Show individual filenames in report

Select this to include the name of the individual files just before their data in the report. If not selected, only the data will be included.

Select parameters to display

Use this button to select a list of parameters to be shown in your report. After selecting parameters, you'll be returned to this page, and the parameters you selected will be displayed. UT1, the absolute time is selected by default, but you can always deselect it.

Advanced filters

The following filters are available only if you click the *Show advanced filters* checkbox.

Select kind(s) of data

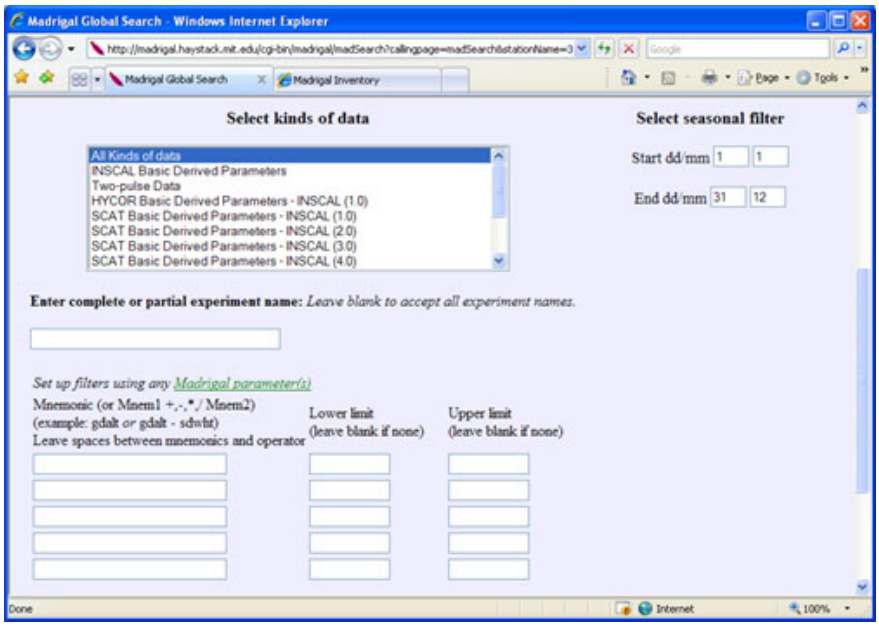
This is a filter that lets you select the kinds of data you would like to search with a particular instrument. A "kind of data" is defined individually at each Madrigal site, and describes the way the raw data was processed to derived the particular Madrigal file. If more than one instrument is selected, a list of the kinds of data for those instruments would be displayed. You can then select which kinds of data you need to in your search. If

you don't know which kinds of data to select, simply select them all..

Select seasonal filter

This filter lets you search through the database in a seasonal or monthly manner. For example, if you need to search for data during the spring time, you enter 3/21 and 6/21 in the start and end date input boxes, respectively. You would then get all experiments between 3/21 and 6/21 that were also in the overall date range entered above. Similarly, if you are only interested in data from November, just enter 11/1 and 11/31, respectively, in the start and end date input boxes. If you do not need this option as a criterion for your search, you can ignore it, since the default will search through the entire year.

Enter complete or partial experiment name



If you want to filter experiments based on experiment name, enter the name of the experiment or partial name. This name-based search is not case sensitive, and you have the option of leaving it blank to get all experiment names.

Select parameters and set up filters

Select parameter(s) to display

Use this button to select a list of parameters to be shown in your report. After selecting parameters, you'll be returned to this page, and the parameters you selected will be displayed. UT1, the absolute time is selected by default, but you can always deselect it.

Set up filters using any Madrigal parameter(s)

Use these parameter filters to apply any filter you want to the data. It doesn't matter whether the parameter is measured (that is, read directly from the file), or derived - any parameter(s) that can be selected for display can be used for a filter. The filters look like the following:

Mnemonic (or Mnem1 +,-,*,/ Mnem2) (example: gdalt *or* gdalt - sdwht) Leave spaces between mnemonics and operator	Lower limit (leave blank if none)	Upper limit (leave blank if none)

These filters work as follows: The first filter implies "gdalt - sdwht" must be greater than 0.0. Since sdwht is shadow height (the distance above any point on the earth where the sun is first visible), this filter implies that only data in direct sunlight will be displayed. The second filter says that BMAG (the magnetic field) must be between 0 and 3e-5 Tesla. Note that the meaning and units of any parameter are available by clicking on them. You'll need to leave spaces between parameters and operators because some parameter names include "+".

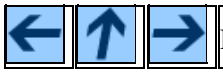
Note that the filter can be based on any single parameter (such as BMAG above), or any two parameters either added, subtracted, multiplied, or divided (as in the "gdalt - sdwht" example above). Leaving either the lower limit or the upper limit blank means there will be either no lower limit or no upper limit. Leaving both blank means the filter is ignored. Note that if the parameter you enter is missing or cannot be calculated, it will be rejected no matter what the range is, since missing data is never in any range.

Continue button

When you select Continue, you will see a summary of all the filter parameters you selected. You can then return to this page to modify any selection, or generate the search command using your API of choice.


Global Madrigal database report | Doc home | Madrigal home

Previous: Print individual records **Up:** Access data through the web **Next:** Plot data from instruments


Plot data from instruments | Doc home | Madrigal home

Previous: Global Madrigal database report **Up:** Access data through the web **Next:** Command line applications

Plot data from instruments

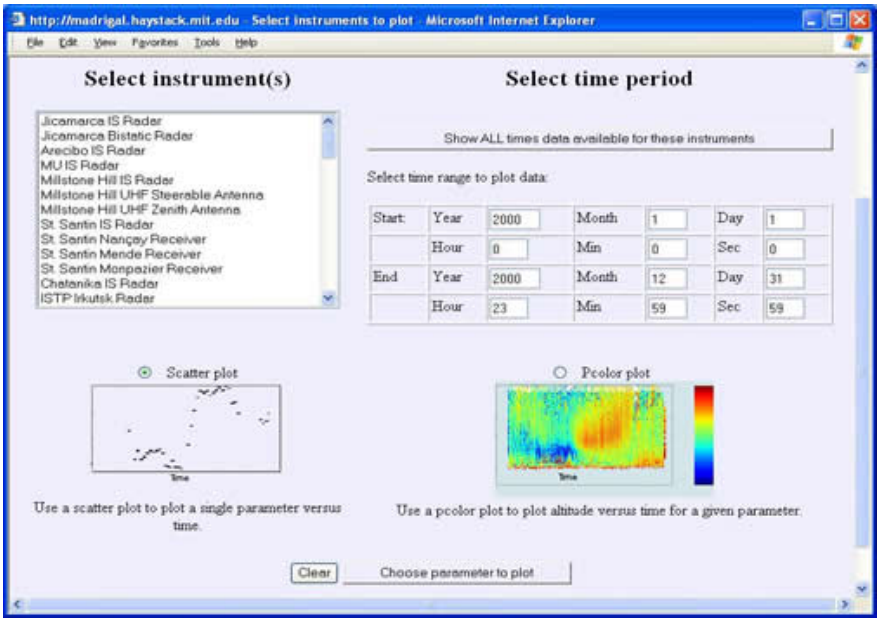
Plot data from instruments allows you to plot data from one or more instruments and/or Madrigal experiments versus time on a single web page. All plots by default will have the same time axis, making it easy to compare data from separate instruments. Plots may be either scatter plots or two-dimensional pcolor plots, where the y axis is altitude. The data can come from more than one Madrigal database, but the source of each plot will be labeled.

Select instrument(s)

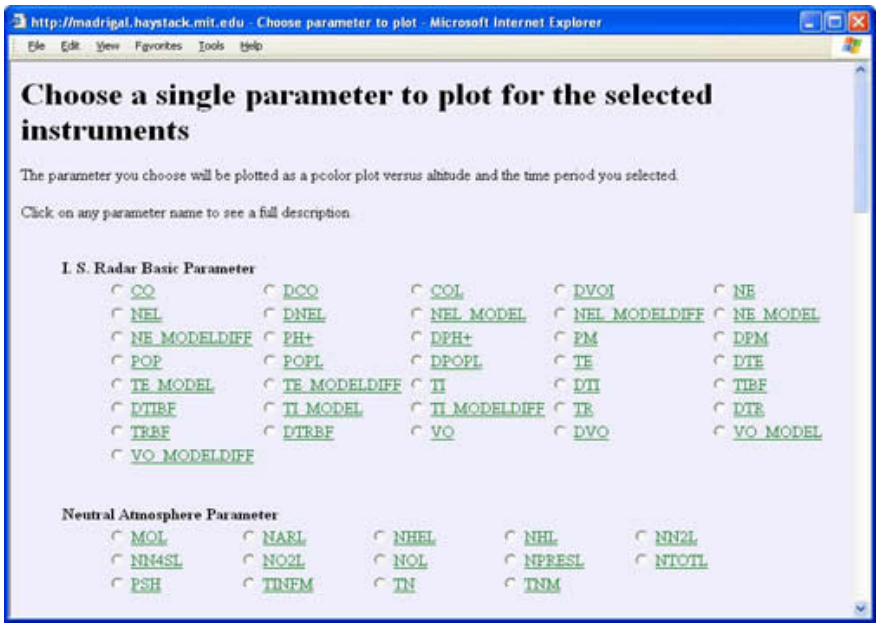
This menu allows you to select the instrument you want to plot data from. If you select more than one instrument, you will generate more than one stacked plot, all with the same axes.

Select time period

The part allows you to select what time period you want to plot. If you don't know when data is available for the instruments you suggested, select the *Show ALL times data available for these instruments* button to see a list. If no data is found for the time period you select, you will automatically be shown the page listing data availability.



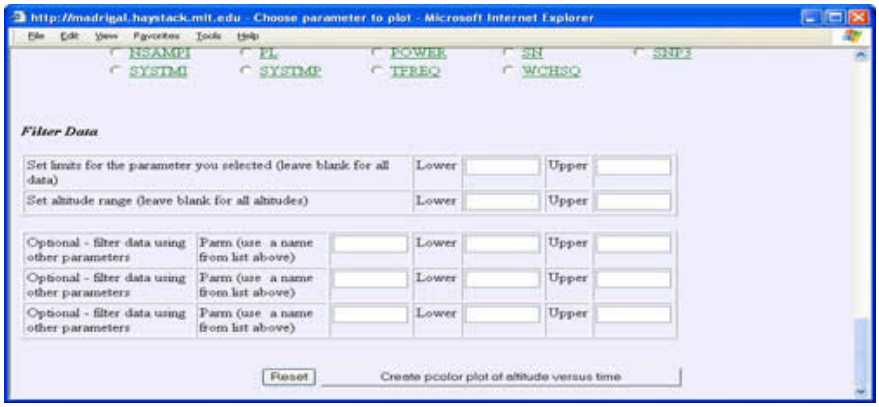
Finally, you choose whether you want a scatter plot or a pcolor plot. In the pcolor plot, the y axis will be altitude, the x axis time, and the value of the parameter you'll select on the next page will be shown on a color scale. When you hit *Choose parameter to plot*, you'll go to the next page.



On this page, you choose a ***single*** parameter to plot. Both the scatter plots and the pcolor plots only plot one parameter at a time. If you select a second parameter, you simply deselect the first one. You do not need to select the x and y axes.

As before, you can see the definition of any parameter by clicking on it .

At the bottom of the page, you can optionally filter the data and set up plot limits. If defaults are okay, just hit the button at the bottom to plot the data.



In the first line, you can set the limits for the parameter you selected. For example, if you selected electron temperature (TE), you might set *Lower*=500 and *Upper*=3000. Leaving either end of the range blank means there is no limit at that end.

If you are creating a pcolor plot, you can also set an altitude filter. If you are creating a scatter plot, this filter will not be there.

You can also filter the data using any other parameter. For example, say you are plotting electron temperature, but only want to see data where the signal-to-noise ratio (SN) is greater than 1. You then set Parm=SN, Lower=1, and Upper=blank in one of the three optional filter lines.

	Plot data from instruments	Doc home	Madrigal home

Previous: Global Madrigal database report **Up:** Access data through the web **Next:** Command line applications

	Command line interface to Madrigal	Doc home	Madrigal home

Previous: Plot data from instruments **Up:** Madrigal user's guide **Next:** Remote access programming tutorial toc

Command line interface to Madrigal

Madrigal contains a number of command line applications that allow access to Madrigal data. Some of these require local access to the files on the Madrigal server, some use the remote API and so can be run from anywhere. This section describes the isprint application in detail, which is relevant both to direct users of the isprint application, and to users of the many API isprint methods that use the same arguments as the isprint command line application.

Command line applications:

Local applications (run only on a madrigal server)

- isprint (arguments are used by many isprint API methods)
- printCedarRecords
- summarizeCedarFile
- translateCedarFile

Remote applications (can be run from anywhere - included in either python or Matlab remote API downloads)

- globalIsprint.py (search over multiple Madrigal experiments - python version)
- globalIsprint.m (search over multiple Madrigal experiments - Matlab version)
- madrigalPColor.py - plot PColor plot of data from a given Madrigal file
- madrigaScatter.py - plot scatter plot of data from a given Madrigal file

Isprint

The application isprint is used to display both measured and derived data from one particular Cedar/Madrigal file. The input file can be any valid Cedar format. It will generate a table of user selected parameters subject to user specified filters. The name isprint original referred to "**I**ncoherent **s**catter **print**" but the application now generically prints any data file in the Cedar format. The engine underlying isprint is the main method for outputting Madrigal data. This application is located in *madroot*/bin.

The isprint application:

- Works with any Cedar file format.
- Treats measured and derived parameters in the same way.
- Accepts any parameter (or two parameters added, subtracted, multiplied, or divided) as a filter, with any number of allowed ranges.
- Keeps track of 1D versus 2D data, and only prints one line of data if only 1D parameters are requested.

To use isprint, the user needs to specify four things:

1. The full filename
2. The parameters desired to be displayed (if any)
3. Any filters to limit the amount of data shown (if any)
4. Any non-default formatting options

The full filename

file= path to file (this argument is required)
Example: file=/opt/madrigal/experiments/1998/mlh/20jan98/mlh980120g.001

The parameters desired to be displayed

Simply enter the desired parameter mnemonic (case-insensitive). They will be displayed in the order entered. If none given, only the header records will be displayed.
Example: azm gdalt Range ti Dti

Any filters to limit the amount of data shown

Time range

date1=mm/dd/yyyy - starting date to be examined. If time1 not given, defaults to 0 UT.
Example: date1=01/20/1998

time1=hh:mm:ss - starting UT time to be examined. If date1 given, is applied to date1. If not, applies on the first day of the experiment.
Example: time1=13:30:00

date2=mm/dd/yyyy - ending date to be examined. If time2 not given, defaults to 0 UT.
Example: date2=01/21/1998

time2=hh:mm:ss - ending UT time to be examined - If date2 not given, refers to date1. If date1 and date2 not given, refers to 1st day.
Example: time2=15:45:00

In the follow arguments ranges are used. If any range value is not given, it may be used to indicate no lower or upper limit (but the comma is always required). Ranges are inclusive of the end points.

Geodetic altitude

z=lower alt limit1, upper alt limit1 [or lower alt limit2 , upper alt limit2 ...] (km)
Example 1: z=100,500 This would limit the geodetic altitude to 100 to 500 km.
Example 2: z=100,200or300,400 This would limit the geodetic altitude to 100 to 200 km or 300 to 400 km.
Example 3: z=,200or300,400 Since the lower limit of the first range is missing, this would limit the geodetic altitude to anything below 200 km or from 300 to 400 km.

Azimuth (from -180 to 180)

az=lower az limit1, upper az limit1 [or lower az limit2 , upper az limit2 ...] (from -180 to 180 degrees)
Example 1: az=100,120 This would limit the azimuth to 100 to 120 degrees.
Example 2: az=-180,-90or90,180 This would limit the azimuth to between -180 and -90 degrees or to between 90 and 180 degrees. Note this allows a filter to go through 180 degrees.

Elevation (from 0 to 90)

el=lower el limit1, upper el limit1 [or lower el limit2 , upper el limit2 ...] (from 0 to 90)
Example 1: el=0,45 This would limit the elevation from 0 to 45 degrees.

Pulse length (in seconds)

plen=lower pl limit1, upper pl limit1 [or lower pl limit2 , upper pl limit2 ...] (pulse len in sec)
Example 1: plen=,5e-4 This would limit the pulse length to 5e-4 seconds or less.

Free form filters

filter=[mnemonic] or [mnemonic1,[+-*/]mnemonic2] , lower limit1 , upper limit1 [or lower limit2 , upper limit2 ...] (any number of filters may be added)
Example: filter=ti,500,1000or2000,3000 Limits the data to points where Ti is between 500 and 1000 degrees or between 2000 and 3000 degrees. Note that the units are always those of the Cedar standard.
Example: filter=gdalt,-,sdwht,0, This filter implies "gdalt - sdwht" must be greater than 0.0. Since sdwht is shadow height (the distance above any point on the earth where the sun is first visible), this filter implies that only data in direct sunlight will be displayed.
Example: filter=ti,/,Dti,100, Limits the data to points where the ratio Ti/dTi is more than 100.

Format options

header=t or f (defaults to header=t, show headers at the beginning of each record)
Example: header=f

badval=bad value string (defaults to "missing")
Example: badval=n/a

assumed=assumed value string (defaults to "assumed")
Example: assumed=-32766

knownbad=known bad value string (defaults to "knownbad")
Example: knownbad=WARNING-BADVALUE

mxchar=maximum characters per line (defaults to no maximum)
Example: mxchar=80

Example: `isprint file=/opt/madrigal/experiments/1998/mlh/20jan98/mil980120g.003 date1=01/20/1998 time1=15:00:00 date2=01/20/1998 time2=16:00:00 z=200,300or500,600 badval=noData filter=gdalt,-,sdwht,0, filter=ti,500,1000 uth gdalt gdlat glon ti te`

This example would show data from mil980120g.003 between 15 and 16 UT on 01/20/1998 where altitude is either between 200 and 300 km or between 500 and 600 km, and where gdalt-sdwht is greater than 0 (point is in sunlight), and where ti is between 500 and 1000. "noData" would be printed if data was not available.

printCedarRecords

printCedarRecord prints specified records of a CEDAR file. The file may be any of the 5 supported CEDAR formats (Madrigal, Blocked Binary, Cbf, Unblocked Binary or ASCII"), and may include any mixture of prologue, header and data records. The format of the file is determined automatically. This application is

located in *madroot*/bin.

Usage: printCedarRecords filename firstRecord lastRecord

summarizeCedarFile

summarizeCedarFile prints a one line per record summary of a CEDAR file. The file may be any of the 5 supported CEDAR formats (Madrigal, Blocked Binary, Cbf, Unblocked Binary or ASCII"), and may include any mixture of prologue, header and data records. The format of the file is determined automatically. This application is located in *madroot*/bin.

Usage: summarize CedarFilefilename

translateCedarFile

This program copies a Cedar file to another file containing the same Cedar records. There are five supported formats:

- 0 - Madrigal
- 1 - Blocked Binary
- 2 - Cbf
- 3 - Unblocked Binary
- 4 - Ascii

The format of the input file is detected automatically.

Usage: translateCedarFile inputFile outputFile outputFileFormat

globallsprint.py

This script runs a global search through Madrigal data from a given URL. It is installed when you install the Remote python API. Because it is a remote program, it can be run from any machine on the internet, using any operating system. It is a python script.

```
Usage:

 globalIsprint --url=<Madrigal url> --parms=<Madrigal parms> --output=<output file> \
       --user_fullname=<user fullname> --user_fullname=<user fullname> \
       --user_fullname=<user fullname> [options]

 where:

 --url=<Madrigal url> - url to homepage of site to be searched
       (ie, http://www.haystack.mit.edu/madrigal/)
       This is required.

 --parms=<Madrigal parms> - a comma delimited string listing the desired Madrigal parameters
       in mnemonic form.  (Example: gdalt,dte,te).  Data will be returned
```

```
      in the same order as given in this string. See
      http://madrigal.haystack.mit.edu/cgi-bin/madrigal/getMetadata and
      choose "Parameter code table" for all possible parameters

--output=<output file name> - the file name to store the resulting data.

--user_fullname=<user fullname> - the full user name (probably in quotes unless your name is S

--user_email=<user email>

--user_affiliation=<user affiliation> - user affiliation.  Use quotes if it contains spaces.

and options are:

--startDate=<MM/DD/YYY> - start date to filter experiments before.  Defaults to allow all expe

--endDate=<MM/DD/YYY> - end date to filter experiments after.  Defaults to allow all experimen

--inst=<instrument list> - comma separated list of instrument codes or names.  See Madrigal do
      using http://madrigal.haystack.mit.edu/cgi-bin/madrigal/getMetadata and
      choose "Instrument Table"
      for this list.  Defaults to allow all instruments. If names are given, the
      argument must be enclosed in double quotes.  An asterick will perform matching as
      in glob.  Examples::

      --inst=10,30

      --inst="Jicamarca IS Radar,Arecibo*"

--expName  - filter experiments by the experiment name.  Give all or part of the experiment na
                   is case insensitive and fnmatch characters * and ? are allowed.  Default i
                   experiment name.

--fileDesc - filter files by their file description string. Give all or part of the file descr
                   is case insensitive and fnmatch characters * and ? are allowed.  Default i
                   file description.



--kindat=<kind of data list> - comma separated list of kind of data codes.  See Madrigal docum
      for this list.  Defaults to allow all kinds of data.  If names are given, the
      argument must be enclosed in double quotes.  An asterick will perform matching as
      in glob. Examples::

      --kindat=3001,13201

      --kindat="INSCAL Basic Derived Parameters,*efwind*,2001"

--filter=<[mnemonic] or [mnemonic1,[+-*/]mnemonic2]>,<lower limit1>,<upper limit1>[or<lower li
      a filter using any measured or derived Madrigal parameter, or two Madrigal parameters ei
      subtracted, multiplied or divided.  Each filter has one or more allowed ranges.  The fil
      data that is in any allowed range.  If the Madrigal parameter value is missing, the filt
      reject that data.  Multiple filter arguments are allowed on the command line.  To skip e
      limit or an upper limit, leave it blank.  Examples::

     filter=ti,500,1000  (Accept when 500 <= Ti <= 1000)

     filter=gdalt,-,sdwht,0,  (Accept when gdalt > shadowheight - that is, point in direct sun
```

```
      filter=gdalt,200,300or1000,1200 (Accept when 200 <= gdalt <= 300 OR 1000 <= gdalt <= 1200

 --seasonalStartDate=<MM/DD> - seasonal start date to filter experiments before.  Use this to s
       year to collect data.  Defaults to Jan 1.  Example:  --seasonalStartDate=07/01 would onl
       experiments after July 1st from each year.

       --seasonalEndDate=<MM/DD> - seasonal end date to filter experiments after.  Use this to
       year to collect data.  Defaults to Dec 31.  Example:  --seasonalEndDate=10/31 would only
       experiments before Oct 31 of each year.

 --showFiles - if given, show file names.  Default is to not show file names.

 --showSummary - if given, summarize all arguments at the beginning.  Default is to not show su

 --includeNonDefault - if given, include all files, including history.  Default is to search on

 --missing=<missing string> (defaults to "missing")

 --assumed=<assumed string> (defaults to "assumed")

 --knownbad=<knownbad string> (defaults to "knownbad")

 --verbose - if given, print each file processed info to stdout.  Default is to run silently.

   Example:

        globalIsprint.py --url=http://www.haystack.mit.edu/madrigal/ --parms='uth,gdalt,ti' --o
            --startDate=01/19/1998 --endDate=01/21/1998 --inst="Millstone*" --verbose --user_fu
            --user_email=brideout@haystack.mit.edu  --user_affiliation=MIT
```

globalIsprint.m

This script runs a global search through Madrigal data from a given URL. It is installed when you install the Remote Matlab API. Because it is a remote program, it can be run from any machine on the internet, using any operating system. It is a Matlab script. Because it uses the Matlab method urlread, which has an upper limit to the amount of data it can read in one call, it is not as robust as globalIsprint.py. It is recommended that globalIsprint.py be used instead if at all possible.

```
function [] = globalIsprint(url, ...
             parms, ...
             output, ...
             user_fullname, ...
             user_email, ...
             user_affiliation, ...
             startTime, ...
             endTime, ...
             inst)
% globalIsprint is a script to search through the entire Madrigal database
% for appropriate data to print in ascii to a file
%
%    Inputs:
%
%        url - url to homepage of site to be searched (Example:
%              'http://www.haystack.mit.edu/madrigal/'
%
%        parms - a comma delimited string listing the desired Madrigal
```

```
%            parameters in mnemonic form.
%            (Example: 'year,month,day,hour,min,sec,gdalt,dte,te').
%            Ascii space-separated ata will be returned in the same
%            order as given in this string. See
%            http://madrigal.haystack.mit.edu/cgi-bin/madrigal/getMetadata
%            "Parameter code table" for all possible parameters.
%
%        output - the local file name to store the resulting ascii data.
%                 (Example: '/tmp/isprint.txt')
%
%        user_fullname - the full user name (Example: 'Bill Rideout')
%
%        user email -  Example: 'brideout@haystack.mit.edu'
%
%        user_affiliation - Example: 'MIT'
%
%        startTime - a Matlab time to begin search at. Example:
%                    datenum('20-Jan-1998 00:00:00') Time in UT
%
%        endTime - a Matlab time to end search at. Example:
%                  datenum('21-Jan-1998 23:59:59') Time in UT
%
%        inst - instrument code (integer).  See
%            http://madrigal.haystack.mit.edu/cgi-bin/madrigal/getMetadata
%            "Instrument Table" for this list. Examples: 30 for Millstone
%            Hill Incoherent Scatter Radar, 80 for Sondrestrom Incoherent
%            Scatter Radar
%
%    Optional inputs
%
%        filters - is the optional filters requested in exactly the form given in isprint
%         command line (example = 'filter=gdalt,,500 filter=ti,500,1000')
%         See:  http://www.haystack.mit.edu/madrigal/ug_commandLine.html for details
%
%        kindats - is an optional array of kindat (kinds of data) codes to accept.
%           The default is an empty array, which will accept all kindats.
%
%        expName - a case insensitive regular expression that matches the experiment
%           name.  Default is zero-length string, which matches all experiment names.
%
%        fileDesc - a case insensitive regular expression that matches the file description.
%           Default is zero-length string, which matches all file descriptions.
%    Returns: Nothing.
%
%    Affects: Writes results to output file
%
%
%
%  Example: globalIsprint('http://www.haystack.mit.edu/madrigal/', ...
%                         'year,month,day,hour,min,sec,gdalt,dte,te', ...
%                         '/tmp/isprint.txt', ...
%                         'Bill Rideout', ...
%                         'brideout@haystack.mit.edu', ...
%                         'MIT', ...
%                         datenum('20-Jan-1998 00:00:00'), ...
%                         datenum('21-Jan-1998 23:59:59'), ...
%                         30);
%
```

madrigalPColor.py

madrigalPColor.py is a command-line application that creates PColor plots from Madrigal. For example, it could be used to plot electron density versus altitude and time for a given Madrigal experiment file.

This application can be run from anywhere, and is included in the python Madrigal remote API download. Use of this module requires matplotlib (matplotlib.sourceforge.net).

Usage:

```
python madrigalPColor.py  --url=<url> --file=<file> --parm=<parm>
       --output=<output> --name=<name> --email=<email> --affiliation=<affiliation>
     [--filter=<filter> --title=<title> --startHour=startHour --endHour=endHour
    --minAlt=minimum_altitude --maxAlt=maximum_altitude --minParm=minimum_parm_value
    --maxParm=maximum_parameter_value]
```

See isprint for details of how filters work.

Example:

```
python madrigalPColor.py  --url=http://madrigal.haystack.mit.edu/madrigal \
       --file=/opt/madrigal/experiments/1998/mlh/20jan98/mlh980120g.002 --parm=nel \
       --output=/tmp/mlh_20jan98.png --name="Bill Rideout" --email=brideout@haystack.mit.edu \
       --affiliation=MIT --filter="filter=elm,80,90 filter=gdalt,,500"
```

madrigalScatter.py

madrigalScatter.py is a command-line application that creates scatter plots from Madrigal. For example, it could be used to Kp versus time for a given Madrigal experiment file.

This application can be run from anywhere, and is included in the python Madrigal remote API download. Use of this module requires matplotlib (matplotlib.sourceforge.net).

Usage:

```
python madrigalScatter.py  --url=<url> --file=<file> --parm=<parm>
   --output=<output> --name=<name> --email=<email> --affiliation=<affiliation>
   [--filter=<filter> --title=<title> --startHour=startHour --endHour=endHour
    --yMin=y_minimum --yMax=y_maximum]
```

See isprint for details of how filters work.

Example:

```
python madrigalScatter.py  --url=http://madrigal.haystack.mit.edu/madrigal \
   --file=/opt/madrigal/experiments/1998/mlh/20jan98/mlh980120g.002 --parm=systmp \
   --output=/tmp/mlh_20jan98.png --name="Bill Rideout" --email=brideout@haystack.mit.edu \
   --affiliation=MIT --filter="filter=elm,80,90"
```

Previous: Plot data from instruments **Up:** Madrigal user's guide **Next:** Remote access programming tutorial toc

 Remote data access programming tutorial - toc | Doc home | Madrigal home

Previous: Command line applications **Up:** Madrigal user's guide **Next:** Remote access - introduction

Madrigal remote data access programming tutorial - Table of Contents

---

Remote data access programming tutorial - Introduction

Madrigal now exposes all the information and capabilities it has as web services, which allows easy access to Madrigal data from any computer on the internet using any platform (Unix, Windows, Mac, etc). Madrigal's web services are basically cgi scripts with simple output that allows easy parsing of the information. Any language that supports the HTTP standard can then access any Madrigal site. We have written remote API's using python and Matlab, but almost any language can be used.

Note that this approach of remotely accessing Madrigal data has been always possible before by parsing the html output meant to be displayed in a web browser (this general programming method is referred to as "screen scraping"). However, not only is this parsing difficult; but the code often breaks when the user interface is modified in any way. With web services the returned cgi scripts are designed to be both simple to parse and stable.

The web services are not implemented according to the SOAP or XMLRPC standard since not all scripting languages have support for these standards (or for XML parsing). Instead they use the simple approach of returning data requested via a query as a delimited text file. Note that the remote access examples written in python and Matlab work on any platform that python and/or Matlab supports.

Use of the remote access methods will be much easier if you understand how Madrigal data is organized. Review the section, "How does Madrigal organize data?" if you have not previously read it.

If you are interested in accessing the capabilities of Madrigal via some other language than Matlab or python, read the following section on Madrigal web services. Also, you might want to read the section on Madrigal web services if you find something you want to add either the remote python or Matlab API's. Otherwise, you can go skip ahead to the section on the remote Matlab API or the remote python API. These API's are available for download from the OpenMadrigal site.

Madrigal web services tutorial

The following section gives a tutorial on the various Madrigal web services in the context of the Madrigal data model. This section is intended to help someone writing their own API to interface with Madrigal. If you simply want to use the existing Matlab or python API's, you can skip ahead to the next sections.

The web services are organized in the same way as the Madrigal data model, from Instrument at the highest level, down to the level of data values. See the Remote data access programming reference manual for complete descriptions of all the Madrigal web services.

Madrigal web services are built using the following cgi scripts:

- getInstrumentsService.py
- getExperimentsService.py
- getExperimentFilesService.py
- getParametersService.py
- isprintService.py
- madCalculatorService.py
- madTimeCalculatorService.py
- radarToGeodeticService.py
- geodeticToRadarService.py
- traceMagneticFieldService.py
- getMetadata

getInstrumentsService.py

The top layer of the Madrigal data model is the instrument. All data in Madrigal is associated with one and only one instrument. Since Madrigal is ground-based focused, an instrument has a particular location associated with it. The following information is available for each instrument:

1. instrument.name Example: *Millstone Hill Incoherent Scatter Radar*
2. instrument.code (unique id) Example: *30*
3. instrument.mnemonic (also unique) (3 char string) Example: *mlh*
4. instrument.latitude Example: *45.0*
5. instrument.longitude Example: *110.0*
6. instrument.altitude Example: *0.015* (km)

To access Instrument metadata, call url:

```
<any Madrigal server>/getInstrumentsService.py
```

For example:

```
http://www.haystack.mit.edu/cgi-bin/madrigal/getInstrumentsService.py
```

will return comma-delimited lines with the six fields above, one for each instrument.

getExperimentsService.py

In the madrigal database system, the data are organized by experiment. An experiment consists of data from a single instrument covering a limited period of time, and, as a rule, is meant to address a particular scientific goal. The following information is available for each instrument:

1. experiment.id (int) Example: *10000111*
2. experiment.url (string) Example: *http://www.haystack.mit.edu/cgi-bin/madtoc/1997/mlh/03dec97*
3. experiment.name (string) Example: *Wide Latitude Substorm Study*
4. experiment.siteid (int) Example: *1*
5. experiment.sitename (string) Example: *Millstone Hill Observatory*
6. experiment.instcode (int) Code of instrument. Example: *30*
7. experiment.instname (string) Instrument name. Example: *Millstone Hill Incoherent Scatter Radar*
8. experiment.start year (int) year of experiment start
9. experiment.start month (int) month of experiment start
10. experiment.start day (int) day of experiment start
11. experiment.start hour (int) hour of experiment start
12. experiment.start minute (int) min of experiment start
13. experiment.start second (int) sec of experiment start
14. experiment.end year (int) year of experiment end
15. experiment.end month (int) month of experiment end
16. experiment.end day (int) day of experiment end
17. experiment.end hour (int) hour of experiment end
18. experiment.end minute (int) min of experiment end
19. experiment.end second (int) sec of experiment end
20. experiment.isLocal (int) 1 if local, 0 if not

To access Experiment metadata for a given instrument and time range, call url:

```
<any Madrigal server>/getExperimentsService.py?code=<instrument code>&startyear=<year>&startmon
&startday=<day>&starthour=<hour>&startmin=<min>&startsec=<sec>&endyear=<year>&endmonth=<month>&
&endhour=<hour>&endmin=<min>&endsec=<sec>&local=<1 or 0>
```

Example: To get all data from instrument Millstone Hill ISR (code=30) during 1998 for all Madrigal servers, url is:

```
http://www.haystack.mit.edu/cgi-bin/madrigal/getExperimentsService.py?code=30&startyear=1998
&startmonth=1&startday=1&starthour=0&startmin=0&startsec=0&endyear=1999&endmonth=1&endday=1
&endhour=0&endmin=0&endsec=0&local=0
```

which will return comma-delimited lines with the 20 fields above, one for each experiment.

getExperimentFilesService.py

The data from a given experiment is stored in one or more experiment files. The format of these files is the Cedar database format, but this is not important since this data is exposed remotely in an ascii format. Note that once we are at the level of experiment files, we need to request the file from the right Madrigal server. The name of correct Madrigal server is given by the experiment.url field, with everything after the "madtoc" indicator replaced by the correct cgi script name (see example below). The following information is available for each experiment file:

1. file.name (string) Example */opt/madrigal/blah/mlh980120g.001*

2. file.kindat (int) Kindat code. Example: *3001*
3. file.kindat desc (string) Kindat description: Example *Basic Derived Parameters*
4. file.category (int) (1=default, 2=variant, 3=history, 4=real-time)
5. file.status (string)(preliminary, final, or any other description)
6. file.permission (int) 0 for public, 1 for private. For now will not return private files.

To access Experiment file metadata for a given experiment, call url:

```
<the correct Madrigal server>/getExperimentFilesService.py?id=<experiment id>
```

where experiment id was returned by getExperimentsService.py, and the correct Madrigal server is experiment.url, truncated before /madtoc/.

Example: to get Experiment File from experiment id = 20001996,experiment url = http://www.eiscat.se/madrigal/cgi-bin/madtoc/1998/mlh/20jan98:

```
http://www.eiscat.se/madrigal/cgi-bin/getExperimentFilesService.py?id=20001996
```

which will return comma-delimited lines with the 6 fields above, one for each experiment file. (Note: this experiment id may change in the future).

getParametersService.py

Any given file is made up a series of records holding parameters. The next cgi scripts returns information about the parameters contained in a given file. The following information is available about the parameters in any experiment file:

1. parameter.mnemonic (string) Example *dti*
2. parameter.description (string) Example: *"F10.7 Multiday average observed (Ott)"*
3. parameter.isError (int) 1 if error parameter, 0 if not
4. parameter.units (string) Example *"W/m2/Hz"*
5. parameter.isMeasured (int) 1 if measured, 0 if derivable
6. parameter.category (string) Example: *"Time Related Parameter"*
7. parameter.isSure (int) - 1 if parameter can be found for every record, 0 if can only be found for some

Note that Madrigal will automatically derive a large number of parameters such as Kp and Magnetic field strength that aren't in the file itself; from the user's point of view these derived parameters appear to be in the file. This is the meaning of column 5: isMeasured. 1 means the data comes from the file itself, 0 means it was derived. To access the parameter information for a given file, call url:

```
<the correct Madrigal server>/getParametersService.py?filename=<full filename>
```

where full filename was returned by getExperimentFilesService.py in column 1, and the correct Madrigal server is again required.

Example: to get the parameters for file mil980120g.001 from the eiscat server:

```
http://www.eiscat.se/madrigal/cgi-bin/getParametersService.py?filename=/usr/local/madroot/exper
```

which will return backslash-delimited lines with the 7 fields above, one for each parameter. Note that backslash is used as a delimiter since commas appear in the parameter descriptions.

isprintService.py

The bottom level of the Madrigal data model is of course the data itself. A Madrigal file is made up of a series of records, each with a start and stop time, representing the integration period of measurement (Madrigal tries to enforce the idea that all measurements take a finite time, but sometimes the start time = the stop time). When a user wants data from a file, they simply specify the parameters they want (and optionally, any filters to apply to the data). For the moment I'll ignore filtering data (for details, see isprint services).

To access the parameters for a given file, call url:

```
<the correct Madrigal server>/isprintService.py?file=<full filename> &parms=<plus-separated par
```

where the filename is the same as in getParametersService.py and parameter mnemonics were returned by getParametersService.py.

Example: To get parameters year, month, day, hour, min, sec, gdlat, glon, gdalt, and ti (temperature ions) from the file /usr/local/madroot/experiments/1998/mlh/20jan98/mil980120g.001, we call url:

```
http://www.eiscat.se/madrigal/cgi-bin/isprintService.py?file=/usr/local/madroot/experiments/199
&parms=year+month+day+hour+min+sec+gdlat+glon+gdalt+ti
```

This will return 11 columns for the 11 requested parameters in the same order as they were requested, with one line for each measurement.

madCalculatorService.py

Madrigal can derive data, as well as display data from existing experiment files. For example, Madrigal can derive MSIS neutral atmosphere parameters for any time and location. To access this derivation engine, call the madCalculatorService.py script. This script allows you to derive one or more parameters for a range of locations at a single time. If you need data at more than one time, simply call this script repeatedly. If you need data that is independent of position (such as Kp) for more than one time, use the madTimeCalculator.py service. Note that this service does **not** provide measured data from Madrigal data files; use isprintService.py for that.

It has the following input arguments:

- year - int (required)
- month - int (required)
- day - int (required)
- hour - int (required)
- min - int (required)
- sec - int (required)
- startLat - Starting geodetic latitude, -90 to 90 (required)
- endLat - Ending geodetic latitude, -90 to 90 (required)
- stepLat - Latitude step (0.1 to 90) (required)
- startLong - Starting geodetic longitude, -180 to 180 (required)
- endLong - Ending geodetic longitude, -180 to 180 (required)
- stepLong - Longitude step (0.1 to 180) (required)
- startAlt - Starting geodetic altitude, >= 0 (required)
- endAlt - Ending geodetic altitude, > 0 (required)
- stepAlt - Altitude step (>= 0.1) (required)

- parms - comma delimited string of Madrigal parameters desired (required)

The script returns comma-delimited data, one line for each combination of lat, long, and alt, with the following fields:

1. latitude 2. longitude 3. altitude 4. Values for each Madrigal parameter listed in argument parms, separated by whitespace

Example: To get the three components on the magnetic field (BN, BE, BD) at midnight 1/1/2000 at latitude = 20, longitude = 40, and altitudes of 100 to 1000 km in steps of 100 km, use:

```
http://madrigal.haystack.mit.edu/cgi-bin/madrigal/madCalculatorService.py?year=2000&month=1&day
hour=0&min=0&sec=0&startLat=20&endLat=20&stepLat=1&startLong=40&endLong=40&stepLong=1&
startAlt=100&endAlt=1000&stepAlt=100&parms=bn,be,bd
```

This will return the following six columns: latitude, longitude, altitude, BN, BE, BD:

```
20.00 40.00 100.00 3.33234e-05 1.23289e-06 1.69326e-05
20.00 40.00 200.00 3.16261e-05 1.07499e-06 1.60124e-05
20.00 40.00 300.00 3.00451e-05 9.33461e-07 1.51637e-05
20.00 40.00 400.00 2.85703e-05 8.06412e-07 1.43793e-05
20.00 40.00 500.00 2.71927e-05 6.92227e-07 1.36528e-05
20.00 40.00 600.00 2.59044e-05 5.89497e-07 1.29786e-05
20.00 40.00 700.00 2.46981e-05 4.96988e-07 1.23518e-05
20.00 40.00 800.00 2.35672e-05 4.13624e-07 1.17679e-05
20.00 40.00 900.00 2.25059e-05 3.38453e-07 1.12232e-05
20.00 40.00 1000.00 2.15087e-05 2.70640e-07 1.07142e-05
```

madTimeCalculatorService.py

The madTimeCalculatorService.py service is similar to the madCalculatorService.py service, but it calculates values for a series of times. However, it is limited to parameters that are independent of position, such as Kp.

Input cgi arguments:

1. startyear - int (required)
2. startmonth - int (required)
3. startday - int (required)
4. starthour - int (required)
5. startmin - int (required)
6. startsec - int (required)
7. endyear - int (required)
8. endmonth - int (required)
9. endday - int (required)
10. endhour - int (required)
11. endmin - int (required)
12. endsec - int (required)
13. stephours - double - number of hours between each measurement (required)
14. parms - comma delimited string of Madrigal parameters desired (required)

Returns comma-delimited data, one line for time, with the following fields:

1. year
2. month
3. day
4. hour
5. min
6. sec
7. Values for each Madrigal parameter listed in argument parms, separated by whitespace

Example: To get Kp and F10.7 for each 3 hour interval on Jan. 1, 2000, use:

```
http://grail.haystack.mit.edu/cgi-bin/madrigal/madTimeCalculatorService.py?
startyear=2000&startmonth=1&startday=1&starthour=0
&startmin=0&startsec=0&endyear=2000&endmonth=1&
endday=1&endhour=23&endmin=59&endsec=59&stephours=1&parms=kp,f10.7
```

This will return the following eight columns: year, month, day, hour, min, sec, kp, f10.7:

```
 2000 1 1 0 0 0 5.30 1.29900e-20
 2000 1 1 3 0 0 4.70 1.29900e-20
 2000 1 1 6 0 0 4.00 1.29900e-20
 2000 1 1 9 0 0 3.30 1.29900e-20
 2000 1 1 12 0 0 4.30 1.29900e-20
 2000 1 1 15 0 0 3.00 1.29900e-20
 2000 1 1 18 0 0 4.30 1.29900e-20
 2000 1 1 21 0 0 3.70 1.29900e-20
```

RadarToGeodeticService.py

The radarToGeodeticService.py script converts a radar position (azimuth, elevation, and range) to a geodetic location. Use geodeticToRadarService.py to go in the other direction.

Input cgi arguments:

1. slatgd - radar geodetic latitude
2. slon - radar longitude
3. saltgd - radar geodetic altitude
4. az - a comma-separated list of radar azimuth in degrees (0 = north)
5. el - a comma-separated list of radar elevation in degrees
6. range - a comma-separated list of radar range in km

Returns comma-delimited data, one line for point in lists:

1. geodetic latitude of point
2. longitude of point
3. geodetic altitude of point

Example: To get the geodetic point at azimuth=100, elevation=45,55, and 65, and range=1000 km from the Millstone Hill IS Radar at latitude=42.619, longitude=288.51, and altitude=0.146, use:

```
http://madrigal.haystack.mit.edu/cgi-bin/madrigal/radarToGeodeticService.py?
```

```
slatgd=42.619&slon=288.51&saltgd=0.146&az=100,100,100&el=45,55,65&range=1000,1000,1000
```

which will return the geodetic latitude, longitude, and altitude of those three radar positions:

```
  41.361338,-64.012651,742.038442
  41.645771,-65.488196,841.794521
  41.932738,-67.098375,918.473216
```

geodeticToRadar

The geodeticToRadarService.py script converts a a geodetic location to radar position (azimuth, elevation, and range) . Use radarToGeodeticService.py to go the other way.

Input cgi arguments:

1. slatgd - radar geodetic latitude
2. slon - radar longitude
3. saltgd - radar geodetic altitude
4. gdlat - a comma-separated list of geodetic latitude of point
5. glon - a comma-separated list of longitude of point
6. gdalt - a comma-separated list of geodetic altitude of point

Returns comma-delimited data, one line for point in lists:

1. radar azimuth in degrees (0 = north)
2. radar elevation in degrees
3. radar range in km

Example: To get the radar location at gdlat=42, glon=290, and gdalt=1000, 2000, and 3000 km from the Millstone Hill IS Radar at latitude=42.619, longitude=288.51, and altitude=0.146, use:

```
http://madrigal.haystack.mit.edu/cgi-bin/madrigal/geodeticToRadarService.py?
slatgd=42.619&slon=288.51&saltgd=0.146&gdlat=42,42,42&glon=290,290,290&gdalt=1000,2000,3000
```

which will return the azimuth, elevation, and range of those three geodetic positions:

```
 117.704418,80.825093,1011.252383
 116.906481,84.802395,2006.351074
 116.270492,86.139493,3004.705956
```

traceMagneticFieldService.py

The traceMagneticFieldService.py service allows you to trace magnetic field lines using either the IGRF model or the Tysganenko models.

Input cgi arguments:

- year
- month
- day
- hour
- min

- sec
- inputType (0 for geodetic, 1 for GSM)
- outputType (0 for geodetic, 1 for GSM)

The following parameters depend on inputType:

- in1 - a comma-separated list of geodetic altitudes or ZGSMs of starting point
- in2 - a comma-separated list of geodetic latitudes or XGSMs of starting point
- in3 - a comma-separated list of longitude or YGSM of starting point

Length of all three lists must be the same

- model - 0 for Tsyganenko, 1 for IGRF
- qualifier - 0 for conjugate, 1 for north_alt, 2 for south_alt, 3 for apex, 4 for GSM XY plane
- topAlt - altitude in km to stop trace at, if qualifier is north_alt or south_alt. If other qualifier, this parameter is not required.

Returns comma-delimited data, one line for point in in lists:

1. geodetic altitude or ZGSM of ending point
2. geodetic latitude or XGSM of ending point
3. longitude or YGSM of ending point

Example: To get the position of the magnetic field line when it crosses 1000 km in the northern hemisphere at midnight 1/1/2004, using the Tysganenko model, with starting points given in GSM and output in geodetic, use:

```
http://www.haystack.mit.edu/cgi-bin/madrigal/traceMagneticFieldService.py?
model=0&stopAlt=1000.000000&year=2004&month=1&day=1&hour=0&min=0&sec=0&
in1=0.269000,0.300000,0.300000,0.300000,0.300000&
in2=0.583000,0.600000,0.600000,0.600000,0.600000&
in3=-0.959870,-10.000000,-1.000000,-10.000000,-1.000000&inputType=1&outputType=0&qualifier=1
```

which in this case returns a comma-separated list of geodetic altitude, latitude, and longitude:

```
missing,missing,missing
1000.000000,74.695086,102.027236
1000.000000,19.026779,129.445239
1000.000000,74.695086,102.027236
1000.000000,19.026779,129.445239
```

getMetadata

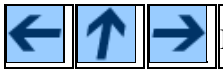
Finally, one other tcl cgi script should be included as part of the web services API: getMetadata. This cgi script allows direct downloading of any of the 10 metadata files . In general, all information needed from Madrigal should be available in an easier-to-use form from the python cgi scripts above. However, there is some information in the metadata that is not available through those scripts, and if that information is ever needed, a user can call getMetadata to download the metadata file itself as follows:

cgiUrl /getMetadata?fileType= *value*

where value is

- 0: expTab.txt

- 1: fileTab.txt
- 2: dataTab.txt
- 3: instTab.txt
- 4: parcods.tab
- 5: siteTab.txt
- 6: typeTab.txt
- 7: instKindatTab.txt
- 8: instParmTab.txt
- 9: madCatTab.txt


Madrigal web services | Doc home | Madrigal home

Previous: Remote access - introduction **Up:** Remote access programming tutorial toc **Next:** Matlab remote access


Remote access using Matlab | Doc home | Madrigal home

Previous: Madrigal web services **Up:** Remote access programming tutorial toc **Next:** Remote access using python

Remote access using Matlab

This page describes the remote Matlab API, and gives two examples of using this API. The first example uses all the basic methods, and outputs text. The second example creates a simple pcolor plot.

The remote Matlab API is organized in the same way as the Madrigal data model, from Instrument at the highest level, down to the level of data values. Readers who are not familiar with the Madrigal data model should read the material in that section before proceeding with this tutorial.

This API and example have been tested on both Windows and Linux, and require only access to the internet and Matlab 5 or greater to run. It is available for download here.

Remote Matlab methods

- getMadrigalCgiUrl - converts the Madrigal url to the cgi form required by the other methods
- getInstrumentsWeb - returns a list of all instruments in Madrigal database
- getExperimentsWeb - returns a list of experiments for a given instument(s) and date range
- getCgiurlForExperiment - returns cgiurl of experiment.url as returned by getExperimentsWeb
- getExperimentFilesWeb - returns a list of files in a given experiment
- getParametersWeb - returns a list of measured parameters in a file, and derived parameters available
- isprintWeb - returns data from a file as an array of doubles using user specified parameters and filters
- isprintUnguarded - similar to isprintWeb, except does not protect the user against requesting too much data for urlread
- madDownloadFile - downloads a Madrigal file to local computer in various formats
- madCalculatorWeb - returns derived parameters for a given time and set of spatial locations
- globalIsprint - returns user-specified data from multiple experiments

A good way to learn how to use this Matlab API is to run this example.

```
   ************************************************************
function cgiUrl = getMadrigalCgiUrl(url)
  getMadrigalCgiUrl     parse the main madrigal page to get the cgi url

  input: url to Madrigal

  output: cgi url for that Madrigal Site

  Note: parses the homepage for the accessData link


   ************************************************************
function instArray = getInstrumentsWeb(cgiurl)
  getInstrumentsWeb     returns an array of instrument structs of instruments found on remote M

  inputs:  cgiurl (string) to Madrigal site cgi directory
      (Example: 'http://www.haystack.mit.edu/cgi-bin/madrigal/')
     Note that method getMadrigalCgiUrl converts homepage url into cgiurl.

  output:
     instArray - array of instrument structs found

       instrument struct has the fields:

     instrument.name (string) Example: 'Millstone Hill Incoherent Scatter Radar'
```

```
    instrument.code (int) Example: 30
    instrument.mnemonic (3 char string) Example: 'mlh'
    instrument.latitude (double) Example: 45.0
    instrument.longitude (double) Example: 110.0
    instrument.altitude (double)  Example: 0.015 (km)

  Raises error if unable to return instrument array

  Example:
  getInstrumentsWeb('http://www.haystack.mit.edu/cgi-bin/madrigal/')


   ***********************************************************
function expArray = getExperimentsWeb(cgiurl, instCodeArray, starttime, endtime, localFlag)
  getExperimentsWeb     returns an array of experiment structs given input filter arguments fro

  Inputs:

      1. cgiurl (string) to Madrigal site cgi directory
        (Example: 'http://www.haystack.mit.edu/cgi-bin/madrigal/')
         Note that method getMadrigalCgiUrl converts homepage url into cgiurl.

      2. instCodeArray - a 1 X N array of ints containing selected instrument codes.  Special v

      3. starttime - Matlab datenum double (must be UTC)

      4. endtime - Matlab datenum double (must be UTC)

      5. localFlag - 1 if local experiments only, 0 if all experiments

   Return array of Experiment struct (May be empty):

   experiment.id (int) Example: 10000111
   experiment.url (string) Example: 'http://www.haystack.mit.edu/cgi-bin/madtoc/1997/mlh/03dec9
   experiment.name (string) Example: 'Wide Latitude Substorm Study'
   experiment.siteid (int) Example: 1
   experiment.sitename (string) Example: 'Millstone Hill Observatory'
   experiment.instcode (int) Code of instrument. Example: 30
   experiment.instname (string) Instrument name. Example: 'Millstone Hill Incoherent Scatter Ra
   experiment.starttime (double) Matlab datenum of experiment start
   experiment.endtime (double) Matlab datenum of experiment end
   experiment.isLocal (int) 1 if local, 0 if not
   experiment.madrigalUrl (string) - home url of Madrigal site with this
       experiment. Example 'http://madrigal.haystack.mit.edu/madrigal'

  Raises error if unable to return experiment array

  Example: expArray = getExperimentsWeb('http://www.haystack.mit.edu/cgi-bin/madrigal/', ...
                                        30, datenum('01/01/1998'), datenum('12/31/1998'), 1);

   Note that if the returned
   experiment is not local, the experiment.id will be -1.  This means that you
   will need to call getExperimentsWeb a second time with the cgiurl of the
   non-local experiment (getCgiurlForExperiment(experiment.madrigalUrl)).  This is because
   while Madrigal sites share metadata about experiments, the real experiment ids are only
   known by the individual Madrigal sites.  See testMadmatlab.m
   for an example of this.


   ***********************************************************
```

```
function cgiurl = getCgiurlForExperiment(experiment)
  getCgiurlForExperiment        returns cgiurl of experiment.url as returned by getExperimentsW

  inputs:  experiment struct as returned by getExperimentsWeb or getExperiments.


  output:
     cgiurl of experiment

  Simply truncates experiment.url to remove /madtoc///

  Example: If expArray is the value returned in the getExperimentsWeb example, and
           expArray(1).url = 'http://madrigal.haystack.mit.edu/cgi-bin/madrigal/madtoc/1998/mlh

     getCgiurlForExperiment(expArray(1))

     returns:

          'http://madrigal.haystack.mit.edu/cgi-bin/madrigal/'


   ********************************************************
function expFileArray = getExperimentFilesWeb(cgiurl, experimentId)
  getExperimentFilesWeb         returns an array of experiment file structs given experiment id

  Note that it is assumed that experiment is local to cgiurl.  If not,
  empty list will be returned. Use getCgiurlForExperiment to get the correct
  cgiurl for any given experiment struct.

  Inputs:

      1. cgiurl (string) to Madrigal site cgi directory that has that
      experiment.
        (Example: 'http://www.haystack.mit.edu/cgi-bin/madrigal/')
         Note that method getMadrigalCgiUrl converts homepage url into cgiurl.

      2. experiment id (int) as returned by getExperiments or
         getExperimentsWeb

   Return array of Experiment File struct (May be empty):

   file.name (string) Example '/opt/mdarigal/blah/mlh980120g.001'
   file.kindat (int) Kindat code.  Example: 3001
   file.kindatdesc (string) Kindat description: Example 'Basic Derived Parameters'
   file.category (int) (1=default, 2=variant, 3=history, 4=real-time)
   file.status (string)('preliminary', 'final', or any other description)
   file.permission (int)  0 for public, 1 for private

  Raises error if unable to return experiment file array

  Example: expFileArray =
  getExperimentFilesWeb('http://www.haystack.mit.edu/cgi-bin/madrigal/', 10001686);


   ********************************************************
function parmArray = getParametersWeb(cgiurl, filename)
  getParametesWeb       returns an array of parameter structs given filename from a remote Madr

  Note that it is assumed that filename is local to cgiurl.  If not,
  empty list will be returned.
```

```
  Inputs:

      1. cgiurl (string) to Madrigal site cgi directory that has that
      filename.
        (Example: 'http://www.haystack.mit.edu/cgi-bin/madrigal/')
         Note that method getMadrigalCgiUrl converts homepage url into cgiurl.

      2. filename (string) as returned by getExperimentFiles or
         getExperimentFilesWeb

   Return array of Parameter struct:

       parameter.mnemonic (string) Example 'dti'
       parameter.description (string) Example:
          "F10.7 Multiday average observed (Ott) - Units: W/m2/Hz"
       parameter.isError (int) 1 if error parameter, 0 if not
       parameter.units (string) Example "W/m2/Hz"
       parameter.isMeasured (int) 1 if measured, 0 if derivable
       parameter.category (string) Example: "Time Related Parameter"
       parameter.isSure (int) 1 if can be found for all records, 0 if only
           for some records (implies not all records have same measured
           parameters)

  Raises error if unable to return parameter array

  Example: parmArray = getParametersWeb('http://www.haystack.mit.edu/cgi-bin/madrigal/', ...
                                        '/opt/madrigal/experiments/1998/mlh/07jan98/mil980107g.


   ************************************************************
function records = isprintWeb(cgiUrl, file, parms, user_fullname, user_email, user_affiliation,
  isprintWeb    Create an isprint-like 3D array of doubles via a command similar to the isprint

  The calling syntax is:

               [records] = isprintWeb(cgiurl, file, parms, user_fullname, user_email, user_aff

   where

     cgiurl (string) to Madrigal site cgi directory that has that
      filename.
        (Example: 'http://www.haystack.mit.edu/cgi-bin/madrigal/')
         Note that method getMadrigalCgiUrl converts homepage url into cgiurl.

     file is path to file
         (example = '/home/brideout/data/mlh980120g.001')

     parms is the desired parameters in the form of a comma-delimited
         string of Madrigal mnemonics (example = 'gdlat,ti,dti')

     user_fullname - is user name (string)

     user_email - is user email address (string)

     user_affiliation - is user affiliation (string)

     filters is the optional filters requested in exactly the form given in isprint
         command line (example = 'time1=15:00:00 date1=01/20/1998
                       time2=15:30:00 date2=01/20/1998 filter=ti,500,1000')
         See:  http://www.haystack.mit.edu/madrigal/ug_commandLine.html for details
```

```
    missing is an optional double to represent missing values.  Defaults to NaN

    assumed is an optional double to represent assumed values.  Defaults to NaN

    knownbad is an optional double to represent knownbad values.  Defaults to NaN

    The returned records is a three dimensional array of double with the dimensions:

        [Number of rows, number of parameters requested, number of records]

    If error or no data returned, will return error explanation string instead.

  Example: data = isprintWeb('http://www.haystack.mit.edu/cgi-bin/madrigal/', ...
                             '/opt/madrigal/experiments/1998/mlh/07jan98/mil980107g.001', ..
                             'gdlat,ti,dti', ...
                             'Bill Rideout', 'wrideout@haystack.mit.edu', 'MIT');

  For now avoids limits with urlread by only asking for maxRecs records at once.  Repeatedly
  calls isprintUnguarded, which is a method without any additional filtering.

   $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $


  ************************************************************
function records = isprintUnguarded(cgiUrl, file, parms, user_fullname, user_email, user_affili
  isprintUnguarded - Create an isprint-like 3D array of doubles via a command similar to the is

  This method differs from isprintWeb in that no protection against
  downloading too much data is provided.  If too much data is requested,
  urlread will fail.

  The calling syntax for this method is:

              [records] = isprintUnguarded(cgiurl, file, parms, user_fullname, user_email, us

   where

    cgiurl (string) to Madrigal site cgi directory that has that
     filename.
       (Example: 'http://www.haystack.mit.edu/cgi-bin/madrigal/')
        Note that method getMadrigalCgiUrl converts homepage url into cgiurl.

    file is path to file
        (example = '/home/brideout/data/mlh980120g.001')

    parms is the desired parameters in the form of a comma-delimited
        string of Madrigal mnemonics (example = 'gdlat,ti,dti')

    user_fullname - is user name (string)

    user_email - is user email address (string)

    user_affiliation - is user affiliation (string)

    filters is the optional filters requested in exactly the form given in isprint
        command line (example = 'time1=15:00:00 date1=01/20/1998
                      time2=15:30:00 date2=01/20/1998 filter=ti,500,1000')
        See:  http://www.haystack.mit.edu/madrigal/ug_commandLine.html for details

    missing is an optional double to represent missing values.  Defaults to NaN
```

```
     assumed is an optional double to represent assumed values.  Defaults to NaN

     knownbad is an optional double to represent knownbad values.  Defaults to NaN

     The returned records is a three dimensional array of double with the dimensions:

         [Number of rows, number of parameters requested, number of records]

     If error or no data returned, will return error explanation string instead.

   Example: data = isprintUnguarded('http://www.haystack.mit.edu/cgi-bin/madrigal/', ...
                                '/opt/madrigal/experiments/1998/mlh/07jan98/mil980107g.001', ..
                                'gdlat,ti,dti', ...
                                'Bill Rideout', 'wrideout@haystack.mit.edu', 'MIT');

   Unlike isprintWeb, no effort is made to protect the user from
   requesting too more data than the Matlab method urlread can handle

    $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $


   ************************************************************
function  result  = madDownloadFile(cgiurl,  fullFilename, outputFile, format )
madDownloadFile downloads a Madrigal file to local computer in various
formats

    The calling syntax is:
       result  = madDownloadFile(cgiurl,  fullFilename, outputFile, [ format ] )

 Inputs:

      1. cgiurl (string) to Madrigal site cgi directory
        (Example: 'http://www.haystack.mit.edu/cgi-bin/madrigal/')
         Note that method getMadrigalCgiUrl converts homepage url into
         cgiurl.

      2. fullFilename - file to download as returned by getExperimentFilesWeb.m

      3. outputFile - name to save new file to

      4. format - one of the following strings:
          'simple', 'cedar_ascii'
              'simple' is the default if not specified, and
              is recommended for all but the most advanced users
              because it is a simple ascii space delimited column
              format, and is the easiest to parse.  See Cedar
              standard for description of cedar_ascii

    $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $



   ************************************************************
function record = madCalculatorWeb(cgiUrl, ...
                                   time, ...
                                   startLat, ...
                                   endLat, ...
                                   stepLat, ...
                                   startLong, ...
```

```
                                    endLong, ...
                                    stepLong, ...
                                    startAlt, ...
                                    endAlt, ...
                                    stepAlt, ...
                                    parms)
  madCalculatorWeb      Create a matrix of doubles via a the Madrigal derivation engine for a t

  The calling syntax is:

                [record] = madCalculatorWeb(cgiurl, time, startLat, endLat, stepLat, startLong,
                                    startAlt, endAlt, stepAlt, parms)

   where

     cgiurl (string) to Madrigal site cgi directory that has that
      filename.
        (Example: 'http://www.haystack.mit.edu/cgi-bin/madrigal/')
         Note that method getMadrigalCgiUrl converts homepage url into cgiurl.

     time - Matlab datenum double (must be UTC) 7. startLat - Starting geodetic latitude, -90 t

     endLat - Ending geodetic latitude, -90 to 90 (required)

     stepLat - Latitude step (0.1 to 90) (required)

     startLong - Starting geodetic longitude, -180 to 180 (required)

     endLong - Ending geodetic longitude, -180 to 180 (required)

     stepLong - Longitude step (0.1 to 180) (required)

     startAlt - Starting geodetic altitude, >= 0 (required)

     endAlt - Ending geodetic altitude, > 0 (required)

     stepAlt - Altitude step (>= 0.1) (required)

     parms is the desired parameters in the form of a comma-delimited
         string of Madrigal mnemonics (example = 'gdlat,ti,dti')


     The returned record is a matrix of doubles with the dimensions:

         [(num lat steps * num long steps * num alt steps), 3 + num of parms]

     The first three columns will always be lat, long, and alt, so there are three
     additional columns to the number of parameters requested via the parms argument.

     If error or no data returned, will return error explanation string instead.

     Example: result = madCalculatorWeb('http:/grail.haystack.mit.edu/cgi-bin/madrigal', ...
                                        now,45,55,5,45,55,5,200,300,50,'bmag,bn');


   ********************************************************
function [] = globalIsprint(url, ...
    parms, ...
    output, ...
    user_fullname, ...
    user_email, ...
```

```
   user_affiliation, ...
   startTime, ...
   endTime, ...
   inst, ...
   filters, ...
   kindats, ...
   expName, ...
   fileDesc)
globalIsprint is a script to search through the entire Madrigal database
for appropriate data to print in ascii to a file

   Inputs:

       url - url to homepage of site to be searched (Example:
             'http://www.haystack.mit.edu/madrigal/'

       parms - a comma delimited string listing the desired Madrigal
           parameters in mnemonic form.
           (Example: 'year,month,day,hour,min,sec,gdalt,dte,te').
           Ascii space-separated data will be returned in the same
           order as given in this string. See
           http://madrigal.haystack.mit.edu/cgi-bin/madrigal/getMetadata
           "Parameter code table" for all possible parameters.

       output - the local file name to store the resulting ascii data.
                (Example: '/tmp/isprint.txt')

       user_fullname - the full user name (Example: 'Bill Rideout')

       user email -  Example: 'brideout@haystack.mit.edu'

       user_affiliation - Example: 'MIT'

       startTime - a Matlab time to begin search at. Example:
                   datenum('20-Jan-1998 00:00:00') Time in UT

       endTime - a Matlab time to end search at. Example:
                 datenum('21-Jan-1998 23:59:59') Time in UT

       inst - instrument code (integer).  See
           http://madrigal.haystack.mit.edu/cgi-bin/madrigal/getMetadata
           "Instrument Table" for this list. Examples: 30 for Millstone
           Hill Incoherent Scatter Radar, 80 for Sondrestrom Incoherent
           Scatter Radar

   Optional inputs

       filters - is the optional filters requested in exactly the form given in isprint
        command line (example = 'filter=gdalt,,500 filter=ti,500,1000')
        See:  http://www.haystack.mit.edu/madrigal/ug_commandLine.html for details

       kindats - is an optional array of kindat (kinds of data) codes to accept.
          The default is an empty array, which will accept all kindats.

       expName - a case insensitive regular expression that matches the experiment
          name.  Default is zero-length string, which matches all experiment names.

       fileDesc - a case insensitive regular expression that matches the file description.
          Default is zero-length string, which matches all file descriptions.

   Returns: Nothing.
```

```
    Affects: Writes results to output file


  Example: globalIsprint('http://www.haystack.mit.edu/madrigal/', ...
                         'year,month,day,hour,min,sec,gdalt,dte,te', ...
                         '/tmp/isprint.txt', ...
                         'Bill Rideout', ...
                         'brideout@haystack.mit.edu', ...
                         'MIT', ...
                         datenum('20-Jan-1998 00:00:00'), ...
                         datenum('21-Jan-1998 23:59:59'), ...
                         30);

  $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $
```

Example Matlab code use this API

```
  % demo program of madmatlab running on a pc or linux

madurl = 'http://madrigal.haystack.mit.edu/madrigal';

cgiurl = getMadrigalCgiUrl(madurl)

'List all instruments, and their latitudes and longitudes:'
instArray = getInstrumentsWeb(cgiurl);
for i = 1:length(instArray)
    [s,errmsg] = sprintf('Instrument: %s, at lat %f and long %f', ...
            instArray(i).name, ...
            instArray(i).latitude, ...
            instArray(i).longitude);
     s
end
% now list all experiments from local Madrigal site with mlh (code 30) in
% 1998 - should be data if default files installed...
startdate = datenum('01/01/1998');
enddate = datenum('12/31/1998');
expArray = getExperimentsWeb(cgiurl, 30, startdate, enddate, 1);
for i = 1:length(expArray)
    [s,errmsg] = sprintf('Experiment name: %s, at url %s and id %i', ...
            expArray(i).name, ...
            expArray(i).url, ...
            expArray(i).id);
     s
end

% now list all files in the first experiment
expFileArray = getExperimentFilesWeb(cgiurl, expArray(1).id);
for i = 1:length(expFileArray)
    [s,errmsg] = sprintf('File name: %s, with kindat %i', ...
            expFileArray(i).name, ...
            expFileArray(i).kindat);
     s
end
% now first 2 parameters in the last file
parmArray = getParametersWeb(cgiurl, expFileArray(end).name)
for i = 1:10
```

```
    [s,errmsg] = sprintf('Parameter mnemonic: %s, description "%s" -- isMeasured = %i', ...
            parmArray(i).mnemonic, ...
            parmArray(i).description, ...
            parmArray(i).isMeasured);
     s
end
%  run isprintWeb for that file for first two parameters
parmStr = sprintf('%s,%s', parmArray(1).mnemonic, parmArray(2).mnemonic);
data = isprintWeb(cgiurl, expFileArray(1).name, parmStr, 'Bill Rideout', 'wrideout@haystack.mit
% print first 10 records
data(:,:,1:10)

% download that data file in simple format
result = madDownloadFile(cgiurl, expFileArray(1).name, '/tmp/junk.txt');
'downloaded file to /tmp/junk.txt'

% run globalIsprint, which can gather data from multiple files at once
globalIsprint('http://www.haystack.mit.edu/madrigal/', ...
              'year,month,day,hour,min,sec,gdalt,dte,te', ...
              '/tmp/isprint.txt', ...
              'Bill Rideout', 'brideout@haystack.mit.edu', 'MIT', ...
              datenum('20-Jan-1998 00:00:00'), datenum('21-Jan-1998 23:59:59'), 30);
'globalIsprint output saved to /tmp/isprint.txt'

% madCalculatorWeb runs the Madrigal derivation engine for any point
data = madCalculatorWeb(cgiurl, datenum(1999,2,15,12,30,0), 45,55,5,-170,-150,10,200,200,0,'sdw
'madCalculator output'
% print data
data

'The following is an example of searching for non-local experiments'
% 61 is the instrument id of Poker Flat ISR - so this will return an
% experiment not from the Millstone Madrigal site
startdate = datenum('04/01/2008');
enddate = datenum('04/30/2008');
expArray = getExperimentsWeb(cgiurl, 61, startdate, enddate, 0);

% calling this now would fail: expFileArray = getExperimentFilesWeb(cgiurl, expArray(1).id);
% Instead, get the cgiurl of the non-local experiment
if (expArray(1).isLocal == 0)
    cgiurl = getMadrigalCgiUrl(expArray(1).madrigalUrl)
    expArray = getExperimentsWeb(cgiurl, 61, startdate, enddate, 0);
end

% now you can get the files
expFileArray = getExperimentFilesWeb(cgiurl, expArray(1).id);
for i = 1:length(expFileArray)
    [s,errmsg] = sprintf('File name: %s, with kindat %i', ...
            expFileArray(i).name, ...
            expFileArray(i).kindat);
     s
end
```

	Remote access using IDL	Doc home	Madrigal home

Previous: Remote access using Python **Up:** Remote access programming tutorial toc **Next:** Remote access reference toc

Remote access using IDL

The following are the methods for accessing Madrigal data remotely:

- madgetallinstruments - returns an array of all instruments in Madrigal database
- madgetexperiments - returns an array of experiments for a given instument(s) and date range
- madgetexperimentfiles - returns an array of files in a given experiment
- madgetexperimentfileparameters - returns an array of measured parameters in a file, and derived parameters available
- madsimpleprint - returns all data in a file as a 2-D array
- madprint - returns data from a file as a 2-D array using user specified parameters and filters
- maddownloadfile - downloads a Madrigal file to local computer in a simple ascii format
- madglobalprint - returns user-specified data from multiple experiments
- madcalculator - returns derived parameters for a given time and set of spatial locations

A good way to learn how to use this IDL API is to run this example.

```
   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;+
; NAME:
;    madgetallinstruments(madurl)
;
; PURPOSE:
;    Get information on all instruments with data at the Madrigal site specified
;    by madUrl
;
; INPUTS:
;      madurl - scalar string giving a fully qualified url to the Madrigal site
;
; OUTPUT: an array of structures with the following fields:
;               1. name - (string) Example: 'Millstone Hill Incoherent Scatter Radar'
;               2. code - (int) Example: 30
;               3. mnemonic - (3 char string) Example: 'mlh'
;               4. latitude - (double) Example: 45.0
;               5. longitude (double) Example: 110.0
;               6. altitude (double)  Example: 0.015 (km)
; EXAMPLE:
;     result = madGetAllInstruments('http://madrigal.haystack.mit.edu/madrigal')
;
; $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $
;
FUNCTION madgetallinstruments, madurl


   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;+
; NAME:
;    madgetexperiments(madurl, instcode, startyear, startmonth, startday, starthour, startmin,
;                                       endyear, endmonth, endday, endhour, endmin, endsec, loca
;
; PURPOSE:
;    Get information on experiments for a given instrument and date range
;
; INPUTS:
```

```
;      madurl - scalar string giving a fully qualified url to the Madrigal site
;      instcode = int or array of ints representing instrument code(s). Special value of 0 sele
;      startyear, startmonth, startday, starthour, startmin, startsec - 6 ints representing sta
;      endyear, endmonth, endday, endhour, endmin, endsec - 6 ints representing end time
;      local - 0 if all sites desired, 1 (default) if only local experiments desired
;
; OUTPUT: an array of structures representing experiments between the dates with the following
;   1. strid (string) Example: "10000111".  Uniquely identifies experiment. Can be converted to
;         Not stored as a long long because that just doesn't seem to work, so user needs to co
;   2. url (string) Example: 'http://www.haystack.mit.edu/cgi-bin/madtoc/1997/mlh/03dec97'
;   3. name (string) Example: 'Wide Latitude Substorm Study'
;   4. siteid (int) Example: 1
;   5. sitename (string) Example: 'Millstone Hill Observatory'
;   6. instcode (int) Code of instrument. Example: 30
;   7. instname (string) Instrument name. Example: 'Millstone Hill Incoherent Scatter Radar'
;   8. startyear, startmonth, startday, starthour, startmin, startsec - 6 ints representing sta
;   9. endyear, endmonth, endday, endhour, endmin, endsec - 6 ints representing end time of exp
;   10. isLocal - 1 if a local experiment, 0 if not
;   11. madrigalUrl - url of Madrigal site.  Used if not a local experiment.
;   If no experiements found, returns a Null Pointer (IDL does not allow empty arrays)
;
;    Note that if the returned experiment is not local, the experiment id will be -1.  This mea
;    will need to rerun madgetexperiments with the url of the
;    non-local experiment (experiment.madrigalUrl).
;    This is because
;    while Madrigal sites share metadata about experiments, the real experiment ids are only
;    known by the individual Madrigal sites.  See example_madidl.pro for an example of this
;           for an example of this.
; EXAMPLE:
;     result = madGetExperiments('http://madrigal.haystack.mit.edu/madrigal', 30, 1998,1,1,0,0,
;
; $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $
;
FUNCTION madGetExperiments, madurl,  instcode, startyear, startmonth, startday, starthour, star
                                     endyear, endmonth, endday, endhour, endmin, endsec, local


   ************************************************************
;+
; NAME:
;    madgetexperimentfiles(madurl, expid, getnondefault)
;
; PURPOSE:
;    returns a list of all  Madrigal Experiment Files for a given experiment id
;
; INPUTS:
;      madurl - scalar string giving a fully qualified url to the Madrigal site
;      expid - experiment id as returned by madgetexperiments
;      getnondefault - an optional argument.  If 1, get all files, including non-default.  If 0
;             get only default files
;
; OUTPUT: an array of structures representing experiment files for that experiment with the fol
;     1. name (string) Example '/opt/madrigal/blah/mlh980120g.001'
;     2. kindat (int) Kindat code.  Example: 3001
;     3. kindatdesc (string) Kindat description: Example 'Basic Derived Parameters'
;     4. category (int) (1=default, 2=variant, 3=history, 4=real-time)
;     5. status (string)('preliminary', 'final', or any other description)
;     6. permission (int)  0 for public, 1 for private
;     7. expId - experiment id (int) of the experiment this MadrigalExperimentFile belongs in
;   If no experiement files found, returns a Null Pointer (IDL does not allow empty arrays)
; EXAMPLE:
```

```
;     result = madGetExperimentFiles('http://madrigal.haystack.mit.edu/madrigal', 300041, 0)
;
; $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $
;
FUNCTION madGetExperimentFiles, madurl,  expId, getNonDefault


   ************************************************************
;+
; NAME:
;    madgetexperimentfileparameters(madurl, fullFilename)
;
; PURPOSE:
;    returns a list of all measured and derivable parameters in file
;
; INPUTS:
;      madurl - scalar string giving a fully qualified url to the Madrigal site
;      fullFilename - full path to experiment file as returned by madGetExperimentFiles.
;
; OUTPUT: an array of structures representing parameters for that experiment file with the foll
;     1. mnemonic (string) Example 'dti'
;     2. description (string) Example: "F10.7 Multiday average observed (Ott)"
;     3. isError (int) 1 if error parameter, 0 if not
;     4. units (string) Example "W/m2/Hz"
;     5. isMeasured (int) 1 if measured, 0 if derivable
;     6. category (string) Example: "Time Related Parameter"
;     7. isSure (int)  1 if parameter can be found for every record, 0 if can only be found for
;     8. isAddIncrement - 1 if additional increment, 0 if normal, -1 if unknown (this capabilit
;                         only added with Madrigal 2.5)
; EXAMPLE:
;     result = madGetExperimentFileParameters('http://madrigal.haystack.mit.edu/madrigal', '/op
;
; $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $
;
FUNCTION madGetExperimentFileParameters, madurl,  fullFilename


   ************************************************************
;+
; NAME:
;    madsimpleprint(madurl, fullFilename, user_fullname, user_email, user_affiliation)
;
; PURPOSE:
;    madSimplePrint prints the data in the given file is a simple ascii format.
;
;        madSimplePrint prints only the parameters in the file, without filters or derived
;        parameters.  To choose which parameters to print, to print derived parameters, or
;        to filter the data, use isprint instead.
;
; INPUTS:
;      madurl - scalar string giving a fully qualified url to the Madrigal site
;      fullFilename - full path to experiment file as returned by madGetExperimentFiles.
;      user_fullname - full name of user making request
;      user_email - email address of user making request
;      user_affiliation - affiliation of user making request
;
; OUTPUT: a structure with following fields:
;     1. parameters - an array of strings giving the mnemonics of the parameters.  Equal to the
;     2. data - a two dimensional array of double.  Number of columns = number of parameters ab
;         is the number of measurments.  Note that Madrigal often contains a number a measurmen
```

```
;         (such as when a radar makes different range measurements at the same time), so two or
; EXAMPLE:
;     result = madSimplePrint('http://madrigal.haystack.mit.edu/madrigal', '/opt/madrigal/blah/
;                                                 'Bill Rideout', 'brideout@haystack.mit.edu', 'MI
;
; $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $
;
FUNCTION madSimplePrint, madurl,  fullFilename, user_fullname, user_email, user_affiliation


   ************************************************************
;+
; NAME:
;    madprint(madurl, fullFilename, parms, filters, user_fullname, user_email, user_affiliation
;
; PURPOSE:
;    madPrint returns a two-dimensional array of doubles based on user-specified parmeters and
;
;       See madSimplePrint to print a file with all data and just those parameters in the file
;
;        madPrint allows you to choose which parameters to print, to print derived parameters,
;        to filter the data.
;
; INPUTS:
;      madurl - scalar string giving a fully qualified url to the Madrigal site
;      fullFilename - full path to experiment file as returned by madGetExperimentFiles.
;      parms - Comma delimited string listing requested parameters (no spaces allowed).
;      filters - Space delimited string listing filters desired, as in isprint command (see
;                  http://madrigal.haystack.mit.edu/madrigal/ug_commandLine.html#isprint for d
;      user_fullname - full name of user making request
;      user_email - email address of user making request
;      user_affiliation - affiliation of user making request
;      ignore_no_data - if 0 (the default), raises error if no data, If 1, ignores no data
;
; OUTPUT:  a two dimensional array of doubles.  Number of columns = number of parameters reques
;         is the number of measurments.  Note that Madrigal often contains a number a measurmen
;         (such as when a radar makes different range measurements at the same time), so two or
; EXAMPLE:
;     result = madPrint('http://madrigal.haystack.mit.edu/madrigal', '/opt/madrigal/blah/mlh980
;                                    'year,month,day,hour,min,sec,gdalt,ti', 'filter=gdalt,500,6
;                                    'Bill Rideout', 'brideout@haystack.mit.edu', 'MIT')
;
; $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $
;
FUNCTION madPrint, madurl,  fullFilename, parms, filters, user_fullname, user_email, user_affil


   ************************************************************
;+
; NAME:
;    maddownloadfile, madurl, fullFilename, outputFile, user_fullname, user_email, user_affilia
;
; PURPOSE:
;    downloads a Madrigal file in simple ascii format to local computer
;
; INPUTS:
;      madurl - scalar string giving a fully qualified url to the Madrigal site
;      fullFilename - full path to experiment file as returned by madGetExperimentFiles.
;      outputFile - path to save file locally
;      user_fullname - full name of user making request
```

```
;      user_email - email address of user making request
;      user_affiliation - affiliation of user making request
;
; EXAMPLE:
;     maddownloadfile, 'http://madrigal.haystack.mit.edu/madrigal', '/opt/madrigal/blah/mlh9801
;                                     'Bill Rideout', 'brideout@haystack.mit.edu', 'MIT'
;
; $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $
;
PRO maddownloadfile, madurl,  fullFilename, outputFile, user_fullname, user_email, user_affilia


   ************************************************************
; globalIsprint is a procedure to search through the entire Madrigal database
; for appropriate data to print in ascii to a file
;
;    Inputs:
;
;        madurl - url to homepage of site to be searched (Example:
;              'http://www.haystack.mit.edu/madrigal/'
;
;        parms - a comma delimited string listing the desired Madrigal
;            parameters in mnemonic form.
;            (Example: 'year,month,day,hour,min,sec,gdalt,dte,te').
;            Ascii space-separated data will be returned in the same
;            order as given in this string. See
;            http://madrigal.haystack.mit.edu/cgi-bin/madrigal/getMetadata
;            "Parameter code table" for all possible parameters.
;
;        output - the local file name to store the resulting ascii data.
;                 (Example: '/tmp/isprint.txt')
;
;        user_fullname - the full user name (Example: 'Bill Rideout')
;
;        user email -  Example: 'brideout@haystack.mit.edu'
;
;        user_affiliation - Example: 'MIT'
;
;        startDate - a time in IDL Julian Date format at which to begin the search
;
;        endDate - a time in IDL Julian Date format at which to end the search
;
;        inst - instrument code (integer).  See
;            http://madrigal.haystack.mit.edu/cgi-bin/madrigal/getMetadata
;            "Instrument Table" for this list. Examples: 30 for Millstone
;            Hill Incoherent Scatter Radar, 80 for Sondrestrom Incoherent
;            Scatter Radar
;
;    Optional inputs
;
;        filters - is the optional filters requested in exactly the form given in isprint
;         command line (example = 'filter=gdalt,,500 filter=ti,500,1000')
;         See:  http://www.haystack.mit.edu/madrigal/ug_commandLine.html for details
;
;        kindats - is an optional array of kindat (kinds of data) codes to accept.
;           The default is a null pointer, which will accept all kindats.
;
;        expName - a case insensitive string as used by strmatch that matches the experiment
;           name.  Default is zero-length string, which matches all experiment names.
;
;        fileDesc - a case insensitive string as used by strmatch that matches the file descrip
```

```
;           Default is zero-length string, which matches all file descriptions.
;
;    Returns: Nothing.
;
;    Affects: Writes results to output file
;
;
;  Example:
;   madglobalprint, 'http://www.haystack.mit.edu/madrigal/',  $
;                           'year,month,day,hour,min,sec,gdalt,dte,te', $
;                           '/tmp/isprint.txt', $
;                           'Bill Rideout',  'brideout@haystack.mit.edu', 'MIT', $
;                           julday(1,19,1998,0,0,0),  julday(1,21,1998,23,59,59), 30
;
;  $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $
;
pro madglobalprint, madurl, parms, output, user_fullname, user_email, user_affiliation, startDa
                            inst, filters, kindats, expName,  fileDesc


   ********************************************************
;+
; NAME:
;    madcalculator(madurl, year,  month, day, hour, min, sec, startLat, endLat, stepLat,
;                               startLong, endLong, stepLong, startAlt, endAlt, stepAlt,  parms)
;
; PURPOSE:
;    madcalculator calculates requested derived parameter for a given time and grid of spatial
;
; INPUTS:
;      madurl - scalar string giving a fully qualified url to the Madrigal site
;      year,  month, day, hour, min, sec - time at which to run calculation
;      startLat - Starting geodetic latitude, -90 to 90 (float)
;      endLat - Ending geodetic latitude, -90 to 90 (float)
;      stepLat - Latitude step (0.1 to 90) (float)
;      startLong - Starting geodetic longitude, -180 to 180  (float)
;      endLong - Ending geodetic longitude, -180 to 180 (float)
;      stepLong - Longitude step (0.1 to 180) (float)
;      startAlt - Starting geodetic altitude, >= 0 (float)
;      endAlt - Ending geodetic altitude, > 0 (float)
;      stepAlt - Altitude step (>= 0.1) (float)
;      parms - comma delimited string of Madrigal parameters desired
;
; OUTPUT: a two dimensional array of doubles.  Number of columns = 3 + number of parameters req
;                first three parameters are always geodetic latitude, longitude, and altitude.
;                is the number of latitudes * number of longitudes * number of altitudes.
; EXAMPLE:
;     result = madcalculator(1999,2,15,12,30,0,45,55,5,-170,-150,10,200,200,0,'bmag,bn')
;
; $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $
;
FUNCTION madcalculator, madurl, year,  month, day, hour, min, sec, startLat, endLat, stepLat,
                        startLong, endLong, stepLong, startAlt, endAlt, stepAlt,  parms
```

Example IDL code

```
  ; Simple example of all methods in the remote IDL interface to Madrigal (madidl)
;
; $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $
```

```
PRO example_madidl

     ; these are the instrument codes for the Millstone ISR.  Run the function madGetAllInstrum
     ; to see a list of all instruments available in Madrigal
     instList = [30,31,32]

     ; run this test for two different sites (Haystack and SRI), using the test files in each s
     siteList = ['http://madrigal.haystack.mit.edu/madrigal', $
                     'http://isr.sri.com/madrigal/' ]
     for i=0,  n_elements(siteList)-1  do begin
         site = siteList[i]
         print, 'Testing Madrigal url ', site

         ; the highest level of Madrigal data is the instrument - here we get a list of all ava
         ; See madgetallinstruments.pro for a definition of the instrument structure returned i
         result = madGetAllInstruments(site)
         print, 'madGetAllInstruments returned', result
         print, ""

         ; the next level of Madrigal is the experiment.  An experiment has only one instrument
         ; it.  In this case, we ask for all experiments in Jan 1998 for any Millstone ISR rada
         ; is a standard test file, its found at both Millstone and SRI
         ; See madgetexperiments.pro for a definition of the experiment structure returned in a
         result = madGetExperiments(site, instList, 1998,1,19,0,0,0, 1998,1,20,23,59,59,1)
         print, 'madGetExperiments returned ', result
         print, ""
         ; verify at least one returned - note a null pointer is returned since IDL does not su
         resultSize = size(result)
         if (resultSize[1] eq 10) then continue ; no experiments found

         ; next we get an array of files in that experiment.  See madgetexperimentfiles.pro for
         ; of the experimentfile structure returned in an array
         expId = long64(result[0].strid)
         result = madGetExperimentFiles(site, expid, 1)
         print, 'madGetExperimentFiles returned ', result
         print, ""
         ; verify at least one returned - note a null pointer is returned since IDL does not su
         resultSize = size(result)
         if (resultSize[1] eq 10) then continue ; no experiment files found

         ; next we want to get a list of all parameters.  Note Madrigal allows access to both p
         ; (called measured parameters) and parameters derivable form the measured ones (called
         ; See madgetexperimentfileparameters.pro for a definition of the parmeter structure re
         fullFilename = result[0].name
         result = madGetExperimentFileParameters(site, fullFilename)
         print, 'madGetExperimentFileParameters returned ', result
         print, ""

         ; the next method effectively downloads the file into an IDL 2-D array.  It only downl
         ; in the file.  See more advanced method madPrint to choose measured or derived parame
         ; See madsimpleprint.pro for details of how the data is returned
         result = madSimplePrint(site, fullFilename,  'Bill Rideout', 'brideout@haystack.mit.ed
         print, 'size of data returned by madSimplePrint is ', size(result.data)
         print, 'madSimplePrint parameters are ', result.parameters
         print, ""

         ; the next method is similar to madsimpleprint, except the user chooses which measured
         ; to download, and can filter data using the filter string as discussed in
         ;    http://madrigal.haystack.mit.edu/madrigal/ug_commandLine.html#isprint
         ; See madprint.pro for details of how the data is returned
```

```
    result = madPrint(site, fullFilename,  'year,month,day,hour,min,sec,gdalt,ti', 'filter
                                    'Bill Rideout', 'brideout@haystack.mit.edu', 'MIT')
    print, 'size of data returned by madPrint is ', size(result)
    print, ""

    ; the next method is similar to madSimplePrint, except that it downloads the data from
    ; a file into a simple column delimited formated file on your local computer
    maddownloadfile, site, fullFilename, '/tmp/junk5.txt', 'Bill Rideout',  'brideout@hays
    print, 'file has been downloaded to /tmp/junk5.txt'

    ; The next method is used to run the Madrigal derivation engine for any random time an
    ; In other words, things such as Magnetic field and geophysicals indices can be calcul
    ; See madCalculator.pro for details of how the data is returned
    result = madCalculator(site, 1999,2,15,12,30,0,45,55,5,-170,-150,10,200,200,0,'bmag,bn
    print, 'madCalculator returned ', result
    print, ""

    ; The final procedure globalPrint allows you to get data from multiple files in a part
    ; at once.  Since it can return very large data sets, it writes to a file rather than
    madglobalPrint, site,  'year,month,day,hour,min,sec,gdalt,dte,te', $
                           '/tmp/isprint.txt', 'Bill Rideout',  'brideout@haystack.mit.edu'
                           julday(1,19,1998,0,0,0),  julday(1,21,1998,23,59,59), 30, $
                           '', ptr_new(), '*world*'


endfor ; next Madrigal site


; an example of getting data from a site from a different Madrigal site - search for Poke
site = 'http://madrigal.haystack.mit.edu/madrigal'
instList = [61] ; Poker Flat ISR
result = madGetExperiments(site, instList, 2008,4,1,0,0,0, 2008,4,30,23,59,59,0)
print, 'madGetExperiments returned ', result
; handle a non-local experiment
if (result[0].isLocal eq 0) then begin
    site = result[0].madrigalUrl
    ; call madGetExperiments again with local url
    result = madGetExperiments(site, instList, 2008,4,1,0,0,0, 2008,4,30,23,59,59,0)
endif
expId = long64(result[0].strid)
result = madGetExperimentFiles(site, expid, 1)
print, 'madGetExperimentFiles returned ', result
print, ""

END
```


Remote access using python	Doc home	Madrigal home	

Previous: Remote access using python **Up:** Remote access programming tutorial toc **Next:** Remote access reference toc

Remote access using python	Doc home	Madrigal home	

Remote access using python

This page describes the remote Python API, and gives some examples of using this API. These examples have been tested on both Windows and Linux, and require only access to the internet and python 2.3 to run. It is available for download here.

The remote Python API is organized in the same way as the Madrigal data model, from Instrument at the highest level, down to the level of data values. Readers who are not familiar with the Madrigal data model should read the material in that section before proceeding with this tutorial.

The basic object in the remote Python API is the *MadrigalData*, found in the madrigalWeb module. To initialize *MadrigalData* requires only the url of the home page on any Madrigal 2.3 (or above) site as an argument. Calling the methods of this object will return all possible information from one Madrigal site. The other objects in madrigalWeb are simply there to hold returned information - for example, the *MadrigalExperiment* object holds information about one experiment.

MadrigalData has the following methods:

- getAllInstruments - returns a list of all *MadrigalInstrument* objects at all Madrigal sites
- getExperiments - returns a list of all *MadrigalExperiment* objects that meet the criteria you set at any Madrigal site
- getExperimentFiles - returns a list of all default *MadrigalExperimentFile* objects for a given experiment id
- getExperimentFileParameters - returns a list of all measured and derivable parameters in file
- ***downloadFile*** - downloadFile will download a Cedar file in the specified format.
- ***simplePrint*** - prints the data in the given file is a simple ascii format.
- isprint - returns as a string the isprint output given file, parms, filters without headers or summary. This is the main method that accesses the raw data in a Madrigal data file.
- madCalculator - returns derived parameters for a range of latitudes, longitudes, and altitudes at a given time. Note that this method does not returned measured values from a file - use isprint for that. Instead, it is useful for accessing parameters available via the Madrigal derivation engine, such as magnetic field or MSIS parameters.
- ***madCalculator2*** - madCalculator2 is similar to madCalculator, except that a random collection of points in space can be specified, rather than a grid of points.
- ***madCalculator3*** - madCalculator3 is similar to madCalculator, except that multiple times can be specified, where each time can have its own unique spatial positions and 1D and 2D parms.
- madTimeCalculator - is similar to madCalculator, except that it returns data from a range of times, but only returns parameters such as Kp that are independent of position.
- radarToGeodetic - converts arrays of az, el, and ranges to geodetic locations.
- geodeticToRadar - converts arrays of points in space to az, el, and range.

See the Madrigal Python API reference guide for complete documentation.

Two applications written with the remote Python API follow. The first is a simple regression test that is run to test web services when Madrigal is installed. The second is a script that downloads realtime data from any desired Madrigal site.

Simple regression test

This simple script calls the following MadrigalData methods:

- getAllInstruments
- getExperiments
- getExperimentFiles
- downloadFile
- simplePrint
- isprint
- madCalculator

This example also shows how to get data from a different Madrigal site than the one you start with.

To use this regression test, cd to the examples directory in the installation directory, and type:

python exampleMadrigalWebServices.py

```
"""exampleMadrigalWebServices.py runs an example of the Madrigal Web Services interface
   for a given Madrigal server.

   usage:

   python exampleMadrigalWebServices.py



"""

# $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $

import madrigalWeb.madrigalWeb

# constants
user_fullname = 'Bill Rideout - automated test'
user_email = 'brideout@haystack.mit.edu'
user_affiliation = 'MIT Haystack'

madrigalUrl = 'http://madrigal.haystack.mit.edu/madrigal'


testData = madrigalWeb.madrigalWeb.MadrigalData(madrigalUrl)



print 'Example of call to getAllInstruments'
instList = testData.getAllInstruments()
# print out Millstone
for inst in instList:
    if inst.code == 30:
        print (str(inst) + '\n')


print 'Example of call to getExperiments'
expList = testData.getExperiments(30, 1998,1,19,0,0,0,1998,1,22,0,0,0)
for exp in expList:
    # should be only one
    print (str(exp) + '\n')
```

```
print 'Example of call to getExperimentFiles'
fileList = testData.getExperimentFiles(expList[0].id)
for thisFile in fileList:
    if thisFile.category == 1:
        print (str(thisFile.name) + '\n')
        thisFilename = thisFile.name
        break

print 'Example of downloadFile - simple and hdf5 formats:'
result = testData.downloadFile(thisFilename, "/tmp/test.txt", "simple")
result = testData.downloadFile(thisFilename, "/tmp/test.hdf5", "simple")

print 'Example of simplePrint - only first 1000 characters printed'
result = testData.simplePrint(thisFilename, user_fullname, user_email, user_affiliation)
print result[:1000]
print

print 'Example of call to getExperimentFileParameters - only first 10 printed'
fileParms = testData.getExperimentFileParameters(thisFilename)
for i in range(10):
    print fileParms[i]
print


print 'Example of call to isprint (prints data)'
print(testData.isprint(thisFilename,
                       'gdalt,ti',
                       'filter=gdalt,500,600 filter=ti,1900,2000',
                       user_fullname, user_email, user_affiliation))


print 'Example of call to madCalculator (gets derived data at any time)'
result = testData.madCalculator(1999,2,15,12,30,0,45,55,5,-170,-150,10,200,200,0,'sdwht,kp')
for line in result:
    for value in line:
        print ('%8.2e ' % (value))
    print('\n')

print 'Example of searching all Madrigal sites for an experiment - here we search for PFISR dat
expList = testData.getExperiments(61,2008,4,1,0,0,0,2008,4,30,0,0,0,local=0)
print expList[0]

print 'Since this experiment is not local (note the experiment id = -1), we need to create a ne
testData2 = madrigalWeb.madrigalWeb.MadrigalData(expList[0].madrigalUrl)

print 'Now repeat the same calls as above to get PFISR data from the SRI site'
expList2 = testData2.getExperiments(61,2008,4,1,0,0,0,2008,4,30,0,0,0,local=1)
print 'This is a PFISR experiment'
print expList2[0]
```

Script to download realtime data from Madrigal

The following is a demonstration script that shows how real-time data can be imported from any Madrigal site that is updated on a real-time basis.

In this example, data is imported from http://www.haystack.mit.edu/madrigal from "Millstone Hill IS Radar". The following Madrigal parameters are retrieved:

year,month,day,hour,min,sec,gdlat,glon,gdalt,az,el,vo,dvo

for all records from the past 15 minutes.

Although the particular Madrigal site (http://www.haystack.mit.edu/madrigal), the instrument ("Millstone Hill IS Radar"), the parameters, and the times are hard-coded in this example, they could be easily be modified to be arguments.

To avoid missing data, we choose one parameter to be the filter parameter: vo. By filtering on this parameter, any "missing" values are filtered out.

To run this script requires the python Madrigal API be installed, which can be downloaded from http://www.haystack.edu/madrigal/madDownload.html.

```
import os,sys,os.path
import string
import time


import madrigalWeb.madrigalWeb


#constants
madrigalUrl = 'http://www.haystack.mit.edu/madrigal'
instrument = 'Millstone Hill IS Radar'

user_fullname = 'Put your name here!!!'
user_email = 'your@email.here'
user_affiliation = 'Put your affiliation here!!!'


# each line of data contains the following parameters
params = 'year,month,day,hour,min,sec,gdlat,glon,gdalt,azm,elm,vo,dvo'
filterParm = 'vo'
timeDelay = 15

# create the main object to get all needed info from Madrigal
madrigalObj = madrigalWeb.madrigalWeb.MadrigalData(madrigalUrl)


# these next few lines convert instrument name to code
code = None
instList = madrigalObj.getAllInstruments()
for inst in instList:
    if inst.name.lower() == instrument.lower():
        code = inst.code
        break

if code == None:
    raise ValueError, 'Unknown instrument %s' % (instrument)


# next, get a list of real time experiments in the last timeDelay minutes
```

```
startTime = time.gmtime(time.time() - timeDelay*60.0)
endTime = time.gmtime(time.time())


try:
    expList = madrigalObj.getExperiments(code, startTime[0],
                                         startTime[1],
                                         startTime[2],
                                         startTime[3],
                                         startTime[4],
                                         startTime[5],
                                         endTime[0],
                                         endTime[1],
                                         endTime[2],
                                         endTime[3],
                                         endTime[4],
                                         endTime[5])

except:
    raise ValueError, 'No realtime experiments found'


# assume there's only one realtime experiment, and get the file names
fileList = madrigalObj.getExperimentFiles(expList[0].id)

if len(fileList) == 0:
    raise ValueError, 'No realtime experiment files found'


# get data from each of the files
startDateStr = time.strftime('%m/%d/%Y', startTime)
startDateStr = ' date1=' + startDateStr
startTimeStr = time.strftime('%H:%M:%S', startTime)
startTimeStr = ' time1=' + startTimeStr
endDateStr = time.strftime('%m/%d/%Y', endTime)
endDateStr = ' date2=' + endDateStr
endTimeStr = time.strftime('%H:%M:%S', endTime)
endTimeStr = ' time2=' + endTimeStr

filterString = 'filter=%s,-1E30,1E30' % (filterParm) + startDateStr + startTimeStr + endDateStr
for dataFile in fileList:
    result = madrigalObj.isprint(dataFile.name, params, filterString,
                                 user_fullname, user_email, user_affiliation)
    # make sure it succeeded
    if result.find('No records were selected') != -1:
        continue
    if result.find('****') != -1:
        continue
    print result
```


Remote access using python	Doc home	Madrigal home	

Previous: Remote access using Matlab **Up:** Remote access programming tutorial toc **Next:** Remote access using IDL

Remote data access programming reference - toc	Doc home	Madrigal home	

Madrigal remote data access programming reference guide - Table of Contents

---

The following is a reference guide to the python cgi scripts that provide the remote Madrigal web interface:

Modules and Packages

traceMagneticFieldService

Table of Contents

Module: geodeticToRadarService **geodeticToRadarService.py**

Classes

geodeticToRadarService	geodeticToRadarService is web service that allows access to the madrigal._Madrec.geodeticToRadar method.

Table of Contents

Table of Contents

Class: geodeticToRadarService **geodeticToRadarService.py**

geodeticToRadarService is web service that allows access to the madrigal._Madrec.geodeticToRadar method.

Like all my python cgi scripts, geodeticToRadarService has the following structure: the entire cgi is contained in one class, with a main function at the end which serves simply to call the __init__ function of the class. This __init__ function is responsible for calling all other class methods. It is made up of a single try block, with the purpose of reporting all exceptions in well-formatted text to both the user and the administrator. The __init__ function first makes sure the pythonlib can be found. It then calls setScriptState to validate the the cgi arguments, which are simply the arguments for the isprint command.

If any uncaught exception is thrown, its caught by the __init__ try block. If its an MadrigalError, additional information is available. The catch blocks attempt to display the error message on the screen by backing out of of large number of possible tags, which might prevent its display (in any case, the error message will always be available in the page source. The formatted error message is also sent to the email address given in the siteTab.txt metadata file.

This script is not meant to be used directly by a user, and thus is named Service. It is meant to be used by scripting languages such as Matlab that want to call getInstruments via the web

Input cgi arguments:

```
slatgd  - radar geodetic latitude

slon - radar longitude

saltgd - radar geodetic altitude

gdlat - a comma-separated list of geodetic latitude of point

glon - a comma-separated list of longitude of point

gdalt - a comma-separated list of geodetic altitude of point
```

Returns comma-delimited data, one line for point in lists:

1. radar azimuth in degrees (0 = north)
2. radar elevation in degrees
3. radar range in km

If error, returns error description

Change history:

Written by Bill Rideout Oct. 10, 2005

$Id: geodeticToRadarService_geodeticToRadarService.py.html 3304 2011-01-17 15:25:59Z brideout $

Methods

__init__

```
__init__ ( self )
```

__init__ run the entire geodeticToRadarService script. All other functions are private and called by __init__.

Inputs: None

Returns: void

Affects: Ouputs geodeticToRadar data as a service.

Exceptions: None.

geodeticToRadar

```
geodeticToRadar ( self )
```

setScriptState

```
setScriptState ( self )
```

Exceptions

'length of three comman-delimited input lists must be equal'

Table of Contents

This document was automatically generated on Fri Dec 30 09:31:03 2005 by HappyDoc version r1_5

Table of Contents

Module: getExperimentFilesService	**getExperimentFilesService.py**

Classes

getExperimentFilesService	getExperimentFilesService is the class that allows remote access to the getExperimentFiles.py script.

Table of Contents

This document was automatically generated on Thu Jan 19 13:26:16 2006 by HappyDoc version r1_5

Table of Contents

Class: getExperimentFilesService	**getExperimentFilesService.py**

getExperimentFilesService is the class that allows remote access to the getExperimentFiles.py script.

Like all my python cgi scripts, getExperimentFilesService has the following structure: the entire cgi is contained in one class, with a main function at the end which serves simply to call the __init__ function of the class. This __init__ function is responsible for calling all other class methods. It is made up of a single try block, with the purpose of reporting all exceptions in well-formatted text to both the user and the administrator. The __init__

function first makes sure the pythonlib can be found. It then calls setScriptState to validate the the cgi arguments, which are simply the arguments for the isprint command.

If any uncaught exception is thrown, its caught by the __init__ try block. If its an MadrigalError, additional information is available. The catch blocks attempt to display the error message on the screen by backing out of of large number of possible tags, which might prevent its display (in any case, the error message will always be available in the page source. The formatted error message is also sent to the email address given in the siteTab.txt metadata file.

This script calls madrigal.ui.web.ui.isTrusted() to determine if user is trusted and should see private files.

This script is not meant to be used directly by a user, and thus is named Service. It is meant to be used by scripting languages such as Matlab that want to call getExperimentFiles via the web

Input cgi arguments:

1. id (int) - experiment id

Returns comma-delimited data, one line for each experiment file, with the following fields:

1. file.name (string) Example `/opt/mdarigal/blah/mlh980120g.001`
2. file.kindat (int) Kindat code. Example: 3001
3. file.kindat desc (string) Kindat description: Example `Basic Derived Parameters`
4. file.category (int) (1=default, 2=variant, 3=history, 4=real-time)
5. file.status (string)(`preliminary`, `final`, or any other description)
6. file.permission (int) 0 for public, 1 for private. For now will not return private files.

Calls script getExperimentFiles.py.

If error, returns error description.

Change history:

Written by Bill Rideout Dec. 16, 2003

$Id: getExperimentFilesService_getExperimentFilesService.py.html 3304 2011-01-17 15:25:59Z brideout $

Methods

getExperimentFilesService is the class that allows remote access to the getExperimentFiles.py script.

__init__

```
__init__ ( self )
```

__init__ run the entire getExperimentFilesService script. All other functions are private and called by __init__.

Inputs: None

Returns: void

Affects: Ouputs getInstrument data as a service.

Exceptions: None.

createObjects

```
createObjects ( self )
```

getExperimentFiles

```
getExperimentFiles ( self )
```

setScriptState

```
setScriptState ( self )
```

Table of Contents

This document was automatically generated on Thu Jan 19 13:26:16 2006 by HappyDoc version r1_5

Table of Contents

Module: getExperimentsService — **getExperimentsService.py**

Classes

getExperimentsService	getExperimentsService is the class that allows remote access to the getExperiments.py script.

Table of Contents

This document was automatically generated on Thu Jan 19 13:26:16 2006 by HappyDoc version r1_5

Table of Contents

Class: **getExperimentsService.py**

getExperimentsService

getExperimentsService is the class that allows remote access to the getExperiments.py script.

Like all my python cgi scripts, getExperimentsService has the following structure: the entire cgi is contained in one class, with a main function at the end which serves simply to call the __init__ function of the class. This __init__ function is responsible for calling all other class methods. It is made up of a single try block, with the purpose of reporting all exceptions in well-formatted text to both the user and the administrator. The __init__ function first makes sure the pythonlib can be found. It then calls setScriptState to validate the the cgi arguments, which are simply the arguments for the isprint command.

If any uncaught exception is thrown, its caught by the __init__ try block. If its an MadrigalError, additional information is available. The catch blocks attempt to display the error message on the screen by backing out of of large number of possible tags, which might prevent its display (in any case, the error message will always be available in the page source. The formatted error message is also sent to the email address given in the siteTab.txt metadata file.

This script is not meant to be used directly by a user, and thus is named Service. It is meant to be used by scripting languages such as Matlab that want to call getExperiments via the web

Input cgi arguments:

1. code - int representing instrument code. Special value of 0 selects all instruments. More than one allowed.
2. startyear - int
3. startmonth - int
4. startday - int
5. starthour - int
6. startmin - int
7. startsec - int
8. endyear - int
9. endmonth - int
10. endday - int
11. endhour - int
12. endmin - int
13. endsec - int
14. local - 1 if local experiments only, 0 if all experiments

Returns comma-delimited data, one line for each experiment, with the following fields:

1. experiment.id (int) Example: 10000111
2. experiment.url (string) Example: `http://www.haystack.mit.edu/cgi-bin/madtoc/1997/mlh/03d`
3. experiment.name (string) Example: `Wide Latitude Substorm Study`
4. experiment.siteid (int) Example: 1
5. experiment.sitename (string) Example: `Millstone Hill Observatory`
6. experiment.instcode (int) Code of instrument. Example: 30
7. experiment.instname (string) Instrument name. Example: `Millstone Hill Incoherent Scatter Radar`
8. experiment.start year (int) year of experiment start
9. experiment.start month (int) month of experiment start
10. experiment.start day (int) day of experiment start
11. experiment.start hour (int) hour of experiment start
12. experiment.start minute (int) min of experiment start
13. experiment.start second (int) sec of experiment start
14. experiment.end year (int) year of experiment end
15. experiment.end month (int) month of experiment end
16. experiment.end day (int) day of experiment end
17. experiment.end hour (int) hour of experiment end
18. experiment.end minute (int) min of experiment end
19. experiment.end second (int) sec of experiment end
20. experiment.isLocal (int) 1 if local, 0 if not

Call script getExperiments.py.

If error, returns error description

Change history:

Written by Bill Rideout Dec. 11, 2003

$Id: getExperimentsService_getExperimentsService.py.html 3304 2011-01-17 15:25:59Z brideout $

Methods

__init__
createObjects
getExperiments
setScriptState

__init__

```
__init__ ( self )
```

__init__ run the entire getExperimentsService script. All other functions are private and called by __init__.

Inputs: None

Returns: void

Affects: Ouputs getInstrument data as a service.

Exceptions: None.

createObjects

```
createObjects ( self )
```

getExperiments

```
getExperiments ( self )
```

setScriptState

```
setScriptState ( self )
```

Table of Contents

This document was automatically generated on Thu Jan 19 13:26:16 2006 by HappyDoc version r1_5

Table of Contents

Module: getInstrumentsService — **getInstrumentsService.py**

Classes

getInstrumentsService	getInstrumentsService is the class that allows remote access to the getInstruments.py script.

Table of Contents

This document was automatically generated on Thu Jan 19 13:26:16 2006 by HappyDoc version r1_5

Table of Contents

Class: getInstrumentsService — **getInstrumentsService.py**

getInstrumentsService is the class that allows remote access to the getInstruments.py script.

Like all my python cgi scripts, getInstrumentsService has the following structure: the entire cgi is contained in one class, with a main function at the end which serves simply to call the __init__ function of the class. This __init__ function is responsible for calling all other class methods. It is made up of a single try block,

with the purpose of reporting all exceptions in well-formatted text to both the user and the administrator. The __init__ function first makes sure the pythonlib can be found. It then calls setScriptState to validate the the cgi arguments, which are simply the arguments for the isprint command.

If any uncaught exception is thrown, its caught by the __init__ try block. If its an MadrigalError, additional information is available. The catch blocks attempt to display the error message on the screen by backing out of of large number of possible tags, which might prevent its display (in any case, the error message will always be available in the page source. The formatted error message is also sent to the email address given in the siteTab.txt metadata file.

This script is not meant to be used directly by a user, and thus is named Service. It is meant to be used by scripting languages such as Matlab that want to call getInstruments via the web

Input cgi arguments (None)

Returns comma-delimited data, one line for each experiment, with the following fields:

1. instrument.name Example: `Millstone Hill Incoherent Scatter Radar`
2. instrument.code Example: 30
3. instrument.mnemonic (3 char string) Example: `mlh`
4. instrument.latitude Example: 45.0
5. instrument.longitude Example: 110.0
6. instrument.altitude Example: 0.015 (km)

Calls script getInstruments.py.

If error, returns error description

Change history:

Written by Bill Rideout Dec. 11, 2003

$Id: getInstrumentsService_getInstrumentsService.py.html 3304 2011-01-17 15:25:59Z brideout $

Methods

__init__
createObjects
getInstruments
setScriptState

__init__

```
__init__ ( self )
```

__init__ run the entire getInstrumentsService script. All other functions are private and called by __init__.

Inputs: None

Returns: void

Affects: Ouputs getInstrument data as a service.

Exceptions: None.

createObjects

```
createObjects ( self )
```

getInstruments

```
getInstruments ( self )
```

setScriptState

```
setScriptState ( self )
```

Table of Contents

Table of Contents

Module: getParametersService — **getParametersService.py**

Classes

getParametersService	getParametersService is the class that allows remote access to the getParameters.py script.

Table of Contents

Table of Contents

Class: getParametersService — **getParametersService.py**

__init__ run the entire getInstrumentsService script. All other functions are private and called by __init__.

getParametersService is the class that allows remote access to the getParameters.py script.

Like all my python cgi scripts, getParametersService has the following structure: the entire cgi is contained in one class, with a main function at the end which serves simply to call the __init__ function of the class. This __init__ function is responsible for calling all other class methods. It is made up of a single try block, with the purpose of reporting all exceptions in well-formatted text to both the user and the administrator. The __init__ function first makes sure the pythonlib can be found. It then calls setScriptState to validate the the cgi arguments, which are simply the arguments for the isprint command.

If any uncaught exception is thrown, its caught by the __init__ try block. If its an MadrigalError, additional information is available. The catch blocks attempt to display the error message on the screen by backing out of of large number of possible tags, which might prevent its display (in any case, the error message will always be available in the page source. The formatted error message is also sent to the email address given in the siteTab.txt metadata file.

This script is not meant to be used directly by a user, and thus is named Service. It is meant to be used by scripting languages such as Matlab that want to call getParameters via the web

Input cgi arguments:

1. filename (string) - full path to madrigal file

Returns backslash-delimited data, one for each parameter either measured or derivable, with the following fields:

1. parameter.mnemonic (string) Example `dti`
2. parameter.description (string) Example: "F10.7 Multiday average observed (Ott)"
3. parameter.isError (int) 1 if error parameter, 0 if not
4. parameter.units (string) Example "W/m2/Hz"
5. parameter.isMeasured (int) 1 if measured, 0 if derivable
6. parameter.category (string) Example: "Time Related Parameter"
7. parameter.isSure (int) - 1 if parameter can be found for every record, 0 if can only be found for some
8. parameter.isAddIncrement - 1 if additional increment, 0 if normal (Added in Madrigal 2.5)

Call script getParameters.py.

If error, returns error description

Change history:

Written by Bill Rideout Dec. 16, 2003

$Id: getParametersService_getParametersService.py.html 3304 2011-01-17 15:25:59Z brideout $

Methods

__init__

```
__init__ ( self )
```

__init__ run the entire getParametersService script. All other functions are private and called by __init__.

Inputs: None

Returns: void

Affects: Ouputs getInstrument data as a service.

Exceptions: None.

createObjects

```
createObjects ( self )
```

getParameters

```
getParameters ( self )
```

setScriptState

```
setScriptState ( self )
```

Table of Contents

This document was automatically generated on Mon Aug 9 12:49:41 2010 by HappyDoc version r1_5

Table of Contents

Module: getVersionService **getVersionService.py**

Classes	
getVersionService	getVersionService is the class that gets the Madrigal version.

Table of Contents

This document was automatically generated on Fri Oct 21 15:43:17 2011 by HappyDoc version r1_5

Table of Contents

Class: getVersionService **getVersionService.py**

getVersionService is the class that gets the Madrigal version.

Like all my python cgi scripts, getVersionService has the following structure: the entire cgi is contained in one class, with a main function at the end which serves simply to call the __init__ function of the class. This __init__ function is responsible for calling all other class methods. It is made up of a single try block, with the purpose of reporting all exceptions in well-formatted text to both the user and the administrator. The __init__ function first makes sure the pythonlib can be found. It then calls setScriptState to validate the the cgi arguments, which are simply the arguments for the isprint command.

If any uncaught exception is thrown, its caught by the __init__ try block. If its an MadrigalError, additional information is available. The catch blocks attempt to display the error message on the screen by backing out of of large number of possible tags, which might prevent its display (in any case, the error message will always be available in the page source. The formatted error message is also sent to the email address given in the siteTab.txt metadata file.

This script is not meant to be used directly by a user, and thus is named Service. It is meant to be used by scripting languages such as Matlab that want to call getInstruments via the web

Input cgi arguments (None)

Returns version as hyphen-separated numbers

If error, returns error description

Change history:

Written by Bill Rideout Sep. 20, 2011

$Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $

Methods

__init__

```
__init__ ( self )
```

__init__ run the entire getVersionService script. All other functions are private and called by __init__.

Inputs: None

Returns: void

Affects: Ouputs getInstrument data as a service.

Exceptions: None.

createObjects

```
createObjects ( self )
```

getVersion

```
getVersion ( self )
```

Exceptions

IOError, 'problem parsing MADROOT/source/configure.ac'

setScriptState

```
setScriptState ( self )
```

Table of Contents

This document was automatically generated on Fri Oct 21 15:43:17 2011 by HappyDoc version r1_5

Table of Contents

Module: isprintService **isprintService.py**

Classes	
IsprintService	IsprintService is the class that produces allows remote access to isprint functionality.

Table of Contents

This document was automatically generated on Thu Jan 19 13:38:49 2006 by HappyDoc version r1_5

Table of Contents

Class: IsprintService **isprintService.py**

IsprintService is the class that produces allows remote access to isprint functionality.

Like all my python cgi scripts, IsprintService has the following structure: the entire cgi is contained in one class, with a main function at the end which serves simply to call the __init__ function of the class. This __init__ function is responsible for calling all other class methods. It is made up of a single try block, with the purpose of reporting all exceptions in well-formatted text to both the user and the administrator. The __init__ function first makes sure the pythonlib can be found. It then calls setScriptState to validate the the cgi arguments, which are simply the arguments for the isprint command.

If any uncaught exception is thrown, its caught by the __init__ try block. If its an MadrigalError, additional information is available. The catch blocks attempt to display the error message on the screen by backing out of of large number of possible tags, which might prevent its display (in any case, the error message will always be available in the page source. The formatted error message is also sent to the email address given in the siteTab.txt metadata file.

This script is not meant to be used directly by a user, and thus is named Service. It is meant to be used by scripting languages such as Matlab that want to call isprint via the web

Input cgi arguments (see isprint command for details):

`file`: The file to be analyzed by isprint.

`parms`: Space delimited list of requested parameters.

`filters`: Space delimited list of filters desired, as in isprint command

`header`: y for headers, n for no header. Defaults to no header

`user_fullname` user name - if given, allows logging, but not required

`user_email` user email - if given, allows logging, but not required

`user_affiliation` user affiliation - if given, allows logging, but not required

Returns isprint output summary. If error, returns error description.

Change history:

Written by Bill Rideout Nov. 13, 2003

Methods

__init__

```
__init__ ( self )
```

__init__ run the entire IsprintService script. All other functions are private and called by __init__.

Inputs: None

Returns: void

Affects: Ouputs isprint data as a service.

Exceptions: None.

createObjects

```
createObjects ( self )
```

outputReport

```
outputReport ( self )
```

setScriptState

```
setScriptState ( self )
```

Table of Contents

This document was automatically generated on Thu Jan 19 13:38:49 2006 by HappyDoc version r1_5

Table of Contents

Module: listFileTimesService **listFileTimesService.py**

__init__ run the entire IsprintService script. All other functions are private and called by __init__.

Classes	
listFileTimesService	listFileTimesService is web service that allows access to the madrigal._Madrec.geodeticToRadar method.

Table of Contents

This document was automatically generated on Fri Sep 9 15:11:48 2011 by HappyDoc version r1_5

Table of Contents

Class: listFileTimesService

listFileTimesService is web service that allows access to the madrigal._Madrec.geodeticToRadar method.

Like all my python cgi scripts, listFileTimesService has the following structure: the entire cgi main function at the end which serves simply to call the __init__ function of the class. This __ calling all other class methods. It is made up of a single try block, with the purpose of reportin text to both the user and the administrator. The __init__ function first makes sure the pythonli setScriptState to validate the the cgi arguments, which are simply the arguments for the isprint

If any uncaught exception is thrown, its caught by the __init__ try block. If its an MadrigalErr available. The catch blocks attempt to display the error message on the screen by backing out which might prevent its display (in any case, the error message will always be available in the message is also sent to the email address given in the siteTab.txt metadata file.

This script is not meant to be used directly by a user, and thus is named Service. It is meant to such as Matlab that want to call listFileTimes via the web

Input cgi arguments:

```
None required.  Optional: expDir - experiment directory to list.  C
experiments[0-9]*
```

Returns comma-delimited data, one line for point in lists:

1. Full path of file
2. File modification time in form YYYY-MM-DD HH:MM:SS (UT time)

If error, returns error description

Change history:

Written by Bill Rideout Sep. 7, 2011

$Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $

Methods

__init__
listFileTimes
setScriptState

__init__

```
__init__ ( self )
```

**__init__ run the entire listFileTimesService scri
are private and called by __init__.**

Inputs: None

Returns: void

Affects: Ouputs file names and modification date as a service.

Exceptions: None.

listFileTimes

```
listFileTimes ( self )
```

setScriptState

```
setScriptState ( self )
```

Table of Contents

This document was automatically generated on Fri Sep 9 15:11:48 2011 by HappyDoc version r1_5

Table of Contents

Module: madCalculatorService — **madCalculatorService.py**

Classes

madCalculatorService	madCalculatorService is the class that allows remote access to the madCalculator.py script.

Table of Contents

This document was automatically generated on Thu Jan 19 13:26:16 2006 by HappyDoc version r1_5

Table of Contents

Class: madCalculatorService **madCalculatorService.py**

madCalculatorService is the class that allows remote access to the madCalculator.py script.

Like all my python cgi scripts, madCalculatorService has the following structure: the entire cgi is contained in one class, with a main function at the end which serves simply to call the __init__ function of the class. This __init__ function is responsible for calling all other class methods. It is made up of a single try block, with the purpose of reporting all exceptions in well-formatted text to both the user and the administrator. The __init__ function first makes sure the pythonlib can be found. It then calls setScriptState to validate the the cgi arguments, which are simply the arguments for the isprint command.

If any uncaught exception is thrown, its caught by the __init__ try block. If its an MadrigalError, additional information is available. The catch blocks attempt to display the error message on the screen by backing out of of large number of possible tags, which might prevent its display (in any case, the error message will always be available in the page source. The formatted error message is also sent to the email address given in the siteTab.txt metadata file.

This script is not meant to be used directly by a user, and thus is named Service. It is meant to be used by scripting languages such as Matlab that want to call madCalculator via the web

Input cgi arguments:

1. year - int (required)
2. month - int (required)
3. day - int (required)
4. hour - int (required)
5. min - int (required)
6. sec - int (required)
7. startLat - Starting geodetic latitude, -90 to 90 (required)
8. endLat - Ending geodetic latitude, -90 to 90 (required)
9. stepLat - Latitude step (0.1 to 90) (required)
10. startLong - Starting geodetic longitude, -180 to 180 (required)
11. endLong - Ending geodetic longitude, -180 to 180 (required)
12. stepLong - Longitude step (0.1 to 180) (required)
13. startAlt - Starting geodetic altitude, >= 0 (required)
14. endAlt - Ending geodetic altitude, > 0 (required)
15. stepAlt - Altitude step (>= 0.1) (required)
16. parms - comma delimited string of Madrigal parameters desired (required)
17. oneD - string in form <parm>,<value> This argument allows the user to set any number of one-D parameters to be used in the calculation. Value must be parameter name, comma, value as double. Example: &oneD=kinst,31.0&oneD=elm,45.0 (optional - 0 or more allowed)

Returns comma-delimited data, one line for each combination of lat, long, and alt, with the following fields:

1. latitude
2. longitude
3. altitude
4. Values for each Madrigal parameter listed in argument parms, separated by whitespace

Calls script madCalculator.py.

If error, returns error description

Change history:

Written by Bill Rideout Feb. 6, 2004

$Id: madCalculatorService_madCalculatorService.py.html 3304 2011-01-17 15:25:59Z brideout $

Methods

__init__

```
__init__ ( self )
```

__init__ run the entire madCalculatorService script. All other functions are private and called by __init__.

Inputs: None

Returns: void

Affects: Ouputs getInstrument data as a service.

Exceptions: None.

createObjects

```
createObjects ( self )
```

madCalculator

```
madCalculator ( self )
```

	setScriptState
	`setScriptState ( self )`

Table of Contents

This document was automatically generated on Fri Oct 3 15:54:26 2008 by HappyDoc version r1_5

Table of Contents

Module: madTimeCalculatorService	**madTimeCalculatorService.py**
Classes	
madTimeCalculatorService	madTimeCalculatorService is the class that allows remote access to the madTimeCalculator.py script.

Table of Contents

This document was automatically generated on Thu Jan 19 13:26:16 2006 by HappyDoc version r1_5

Table of Contents

Class: madTimeCalculatorService	**madTimeCalculatorService.py**

madTimeCalculatorService is the class that allows remote access to the madTimeCalculator.py script.

Like all my python cgi scripts, madTimeCalculatorService has the following structure: the entire cgi is contained in one class, with a main function at the end which serves simply to call the __init__ function of the class. This __init__ function is responsible for calling all other class methods. It is made up of a single try block, with the purpose of reporting all exceptions in well-formatted text to both the user and the administrator. The __init__ function first makes sure the pythonlib can be found. It then calls setScriptState to validate the the cgi arguments, which are simply the arguments for the isprint command.

If any uncaught exception is thrown, its caught by the __init__ try block. If its an MadrigalError, additional information is available. The catch blocks attempt to display the error message on the screen by backing out of of large number of possible tags, which might prevent its display (in any case, the error message will always be available in the page source. The formatted error message is also sent to the email address given in the siteTab.txt metadata file.

This script is not meant to be used directly by a user, and thus is named Service. It is meant to be used by scripting languages such as Matlab that want to call madTimeCalculator via the web

Input cgi arguments:

1. startyear - int (required)
2. startmonth - int (required)
3. startday - int (required)
4. starthour - int (required)
5. startmin - int (required)
6. startsec - int (required)
7. endyear - int (required)
8. endmonth - int (required)
9. endday - int (required)
10. endhour - int (required)
11. endmin - int (required)
12. endsec - int (required)
13. stephours - double - number of hours between each measurement (required)
14. parms - comma delimited string of Madrigal parameters desired (required)

Returns comma-delimited data, one line for time, with the following fields:

1. year
2. month
3. day
4. hour
5. min
6. sec
7. Values for each Madrigal parameter listed in argument parms, separated by whitespace

Calls script madTimeCalculator.py.

If error, returns error description

Change history:

Written by Bill Rideout July. 12, 2004

$Id: madTimeCalculatorService_madTimeCalculatorService.py.html 3304 2011-01-17 15:25:59Z brideout $

Methods	
	__init__ createObjects madTimeCalculator

madTimeCalculatorService is the class that allows remote access to the madTimeCalculator.py script.

setScriptState

__init__

```
__init__ ( self )
```

__init__ run the entire madTimeCalculatorService script. All other functions are private and called by __init__.

Inputs: None

Returns: void

Affects: Ouputs madTimeCalculator data as a service.

Exceptions: None.

createObjects

```
createObjects ( self )
```

madTimeCalculator

```
madTimeCalculator ( self )
```

setScriptState

```
setScriptState ( self )
```

Table of Contents

This document was automatically generated on Thu Jan 19 13:26:16 2006 by HappyDoc version r1_5

Table of Contents

Module: radarToGeodeticService	**radarToGeodeticService.py**

Classes

radarToGeodeticService	radarToGeodeticService is web service that allows access to the madrigal._Madrec.radarToGeodetic method.

Table of Contents

This document was automatically generated on Fri Dec 30 09:31:03 2005 by HappyDoc version r1_5

Table of Contents

Class: radarToGeodeticService

radarToGeodeticService is web service that allows access to th madrigal._Madrec.radarToGeodetic method.

Like all my python cgi scripts, radarToGeodeticService has the following structure: the e cgi is contained in one class, with a main function at the end which serves simply to call t __init__ function of the class. This __init__ function is responsible for calling all other cl methods. It is made up of a single try block, with the purpose of reporting all exceptions i well-formatted text to both the user and the administrator. The __init__ function first mak sure the pythonlib can be found. It then calls setScriptState to validate the the cgi argume which are simply the arguments for the isprint command.

If any uncaught exception is thrown, its caught by the __init__ try block. If its an MadrigalError, additional information is available. The catch blocks attempt to display th error message on the screen by backing out of of large number of possible tags, which mi prevent its display (in any case, the error message will always be available in the page so The formatted error message is also sent to the email address given in the siteTab.txt metadata file.

This script is not meant to be used directly by a user, and thus is named Service. It is mea be used by scripting languages such as Matlab that want to call getInstruments via the we

Input cgi arguments:

```
slatgd  - radar geodetic latitude

slon - radar longitude

saltgd - radar geodetic altitude

az - a comma-separated list of radar azimuth in degrees (0 = no

el - a comma-separated list of radar elevation in degrees

range - a comma-separated list of radar range in km
```

Returns comma-delimited data, one line for point in lists:

1. geodetic latitude of point
2. longitude of point
3. geodetic altitude of point

If error, returns error description

Change history:

Written by Bill Rideout Oct. 10, 2005

$Id: radarToGeodeticService_radarToGeodeticService.py.html 3304 2011-01-17 15:25:5
brideout $

Methods

__init__
radarToGeodetic
setScriptState

__init__

```
__init__ ( self )
```

__init__ run the entire radarToGeodeticService script. All other functions are private and called b __init__.

Inputs: None

Returns: void

Affects: Ouputs radarToGeodetic data as a service.

Exceptions: None.

radarToGeodetic

```
radarToGeodetic ( self )
```

setScriptState

```
setScriptState ( self )
```

Exceptions

'length of three comman-delimited input lists must be equal'

Table of Contents

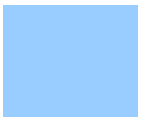
This document was automatically generated on Fri Dec 30 09:31:03 2005 by HappyDoc version r1_5

Table of Contents

Module: traceMagneticFieldService **traceMagneticFieldService.py**

Classes

traceMagneticFieldService	traceMagneticFieldService is web service that allows access to the



madrigal._Madrec.pyTraceMagneticField method.

Table of Contents

This document was automatically generated on Fri Dec 30 09:31:03 2005 by HappyDoc version r1_5

Table of Contents

Class: traceMagneticFieldService

traceMagneticFieldService is web service that allows access madrigal._Madrec.pyTraceMagneticField method.

Like all my python cgi scripts, traceMagneticFieldService has the following structure: with a main function at the end which serves simply to call the __init__ function of the responsible for calling all other class methods. It is made up of a single try block, with in well-formatted text to both the user and the administrator. The __init__ function firs found. It then calls setScriptState to validate the the cgi arguments, which are simply th

If any uncaught exception is thrown, its caught by the __init__ try block. If its an Madr available. The catch blocks attempt to display the error message on the screen by backi tags, which might prevent its display (in any case, the error message will always be ava formatted error message is also sent to the email address given in the siteTab.txt metad

This script is not meant to be used directly by a user, and thus is named Service. It is m such as Matlab that want to call getInstruments via the web

Input cgi arguments:

```
year

month

day

hour

min

sec

inputType (0 for geodetic, 1 for GSM)

outputType (0 for geodetic, 1 for GSM)

    The following parameter depend on inputType:

in1 - a comma-separated list of geodetic altitudes or ZGSMs o

in2 - a comma-separated list of geodetic latitudes or XGSMs o
```

```
in3 - a comma-separated list of longitude or YGSM of starting

    Length of all three lists must be the same

model - 0 for Tsyganenko, 1 for IGRF

qualifier - 0 for conjugate, 1 for north_alt, 2 for south_alt

stopAlt - altitude in km to stop trace at, if qualifier is no
If other qualifier, this parameter is not required.
```

Returns comma-delimited data, one line for point in in lists:

1. geodetic altitude or ZGSM of ending point
2. geodetic latitude or XGSM of ending point
3. longitude or YGSM of ending point

If error, returns error description

Change history:

Written by Bill Rideout Dec. 10, 2004

Methods

__init__
setScriptState
traceMagneticField

__init__

```
__init__ ( self )
```

__init__ run the entire traceMagneticFieldS functions are private and called by __init__

Inputs: None

Returns: void

Affects: Ouputs getInstrument data as a service.

Exceptions: None.

setScriptState

```
setScriptState ( self )
```

Exceptions

'length of three comman-delimited input

traceMagneticField

```
traceMagneticField ( self )
```

Exceptions

'Present IGRF line trace code cannot tra

Table of Contents

This document was automatically generated on Fri Dec 30 09:31:03 2005 by HappyDoc version r1_5

Scripts supporting the cgi interface | Doc home | Madrigal home

Previous: Web services - cgi reference **Up:** Remote data access reference - toc **Next:** Matlab API reference

---

The following is the reference guide to the python scripts that are used to implement some of the python web service cgi scripts:

HappyDoc Generated Documentation

Modules and Packages

getExperimentFiles	getExperimentFiles.py is a script run to return a text output of selected experiment files data.
getExperiments	getExperiments.py is a script run to return a text output of selected experiments data.
getInstruments	getInstruments.py is a script run to return a text output of all instrument data.
getParameters	getParameters.py is a script run to return a text output of parameter information for a given file.
madCalculator	madCalculator.py is a script run to return a text output of derived Madrigal parameters for a time and a range of gdlat, glon, and gdalt.
madTimeCalculator	madTimeCalculator.py is a script run to return a text output of derived Madrigal parameters for parameters that depend only on time.

Scripts supporting the cgi interface | Doc home | Madrigal home

Previous: Web services - cgi reference **Up:** Remote data access reference - toc **Next:** Matlab API reference

---

Table of Contents

__init__ run the entire traceMagneticFieldService script. All other functions are private and called by __init__

Module:	**getExperimentFiles.py**

getExperimentFiles

getExperimentFiles.py is a script run to return a text output of selected experiment files data.

It is presently used by the madmatlab methods getExperimentFiles, and via the cgi script getExperimentFilesService.py the madmatlab method getExperimentFilesWeb. It has the following input arguments:

--id=<experiment id> (integer)

--trusted=<0 if not, 1 if is> (default to not trusted)

Returns comma-delimited data, one line for each experiment file, with the following fields:

1. file.name (string) Example `/opt/mdarigal/blah/mlh980120g.001`
2. file.kindat (int) Kindat code. Example: 3001
3. file.kindat desc (string) Kindat description: Example `Basic Derived Parameters`
4. file.category (int) (1=default, 2=variant, 3=history, 4=real-time)
5. file.status (string)(`preliminary`, `final`, or any other description)
6. file.permission (int) 0 for public, 1 for private. For now will not return private files.

Returns empty string if experiment id nor found, and returns status -1.

This script, and the corresponding cgi script getExperimentFilesService.py are available to any scripting language that wants to access Madrigal metadata.

Table of Contents

This document was automatically generated on Fri Dec 30 10:17:51 2005 by HappyDoc version r1_5

Table of Contents

Module:	**getExperiments.py**

getExperiments

getExperiments.py is a script run to return a text output of selected experiments data.

It is presently used by the madmatlab methods getExperiments, and via the cgi script getExperimentsService.py the madmatlab method getExperimentsWeb. It has the following input arguments:

--code=<instrument code> (integer) - more than one allowed; 0 indicates all instruments (default)

--startyear=<year>

--startmonth=<month> (1-12) (defaults to 1)

--startday=<day> (1-31) (defaults to 1)

--starthour=<hour> (0-24)(defaults to 0)

--startmin=<min> (0-59)(defaults to 0)

--startsec=<sec> (0-61)(defaults to 0)

--endyear=<year>

--endmonth=<month> (1-12) (defaults to 12)

--endday=<day> (1-31) (defaults to 31)

--endhour=<hour> (0-24) (defaults to 23)

--endmin=<min> (0-59) (defaults to 59)

--endsec=<sec> (0-61) (defaults to 59)

--local=<1 or 0> - if 1, return local experiments only, otherwise, return all. Defaults to local.

--trusted=<0 if not, 1 if is> (default to not trusted). If not trusted, skips private experiments

If no code given, all instruments assumed. If no startyear given, starttime filter not used. If no endyear given, no endtime filter used.

Returns comma-delimited data, one line for each experiment, with the following fields:

1. experiment.id (int) Example: 10000111
2. experiment.url (string) Example: `http://www.haystack.mit.edu/cgi-bin/madtoc/1997/mlh/03dec97`
3. experiment.name (string) Example: `Wide Latitude Substorm Study`
4. experiment.siteid (int) Example: 1
5. experiment.sitename (string) Example: `Millstone Hill Observatory`
6. experiment.instcode (int) Code of instrument. Example: 30
7. experiment.instname (string) Instrument name. Example: `Millstone Hill Incoherent Scatter Radar`
8. experiment.start year (int) year of experiment start
9. experiment.start month (int) month of experiment start
10. experiment.start day (int) day of experiment start
11. experiment.start hour (int) hour of experiment start
12. experiment.start minute (int) min of experiment start
13. experiment.start second (int) sec of experiment start
14. experiment.end year (int) year of experiment end
15. experiment.end month (int) month of experiment end
16. experiment.end day (int) day of experiment end
17. experiment.end hour (int) hour of experiment end

18. experiment.end minute (int) min of experiment end
19. experiment.end second (int) sec of experiment end
20. experiment.isLocal (int) 1 if local, 0 if not

This script, and the corresponding cgi script getExperimentsService.py are available to any scripting language that wants to access Madrigal metadata.

Table of Contents

This document was automatically generated on Mon Aug 9 13:30:39 2010 by HappyDoc version r1_5

Table of Contents

Module:	**getInstruments.py**

getInstruments

getInstruments.py is a script run to return a text output of all instrument data.

It is presently used by the madmatlab methods getInstruments, and via the cgi script getInstrumentsService.py the madmatlab method getInstrumentsWeb. It has no input arguments.

Returns comma-delimited data, one line for each experiment, with the following fields:

1. instrument.name Example: `Millstone Hill Incoherent Scatter Radar`
2. instrument.code Example: 30
3. instrument.mnemonic (3 char string) Example: `mlh`
4. instrument.latitude Example: 45.0
5. instrument.longitude Example: 110.0
6. instrument.altitude Example: 0.015 (km)

This script, and the corresponding cgi script getInstrumentsService.py are available to any scripting language that wants to access Madrigal metadata.

Table of Contents

This document was automatically generated on Fri Dec 30 10:17:51 2005 by HappyDoc version r1_5

Table of Contents

Module:	**getParameters.py**

getParameters

getParameters.py is a script run to return a text output of parameter information for a given file.

It is presently used by the madmatlab methods getParameters, and via the cgi script getParametersService.py the madmatlab method getParametersWeb. It has the following input argument:

--filename=<full path to data file>

Returns backslash-delimited data, one for each parameter either measured or derivable, with the following fields:

1. parameter.mnemonic (string) Example `dti`
2. parameter.description (string) Example: "F10.7 Multiday average observed (Ott)"
3. parameter.isError (int) 1 if error parameter, 0 if not
4. parameter.units (string) Example "W/m2/Hz"
5. parameter.isMeasured (int) 1 if measured, 0 if derivable
6. parameter.category (string) Example: "Time Related Parameter"
7. parameter.isSure (int) - 1 if parameter can be found for every record, 0 if can only be found for some
8. parameter.isAddIncrement - 1 if additional increment, 0 if normal (Added in Madrigal 2.5)

Returns empty string if filename not found, and returns status -1.

Uses backslash-delimited since some strings contain commas.

This script, and the corresponding cgi script getParametersService.py are available to any scripting language that wants to access Madrigal parameter info.

Table of Contents

This document was automatically generated on Mon Aug 9 13:30:39 2010 by HappyDoc version r1_5

Table of Contents

Module:	**madCalculator.py**
madCalculator	madCalculator.py is a script run to return a text output of derived Madrigal parameters for a time and a range of gdlat, glon, and gdalt. It is presently used by the madmatlab methods madCalculator, and via the cgi script madCalculatorService.py the madmatlab method madCalculatorWeb. It has the following input arguments: --date=<MM/DD/YYYY> (required) --time=<HH:MM:SS> (optional - if no HH:MM:SS given, 00:00:00 assumed) --startLat=<latitude> Starting geodetic latitude, -90 to 90 (required) --endLat=<latitude> Ending geodetic latitude, -90 to 90 (required) --stepLat=<latitude step> Latitude step (0.1 to 90) (required) --startLong=<longitude> Starting geodetic longitude, -180 to 180 (required) --endLong=<longitude> Ending geodetic longitude, -180 to 180 (required)

__init__ run the entire traceMagneticFieldService script. All other functions are private and called by __init__

--stepLong=<longitude step> Longitude step (0.1 to 180) (required)

--startAlt=<altitude> Starting geodetic altitude, >= 0 (required)

--endAlt=<altitude> Ending geodetic altitude, > 0 (required)

--stepAlt=<altitude step> Altitude step (>= 0.1) (required)

--parms=<comma delimited string of Madrigal parameters desired> (required)

--showHeader (optional) Prints header line before data

--oneD=<parm>,<value> (optional - 0 or more allowed) This argument allows the user to set any number of one-D parameters to be used in the calculation. Value must be parameter name, comma, value as double. Example: --oneD=kinst,31.0 --oneD=elm=45.0

Returns comma-delimited data, one line for each combination of lat, long, and alt, with the following fields:

1. latitude 2. longitude 3. altitude 4. Values for each Madrigal parameter listed in argument parms, separated by whitespace

If --showHeader given, also prints header line before data

This script, and the corresponding cgi script madCalculatorService.py are available to any scripting language that wants to access Madrigal derived parameters.

$Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $

Functions	
	isnan
	isnan
	`isnan ( x )`

Table of Contents

This document was automatically generated on Mon Aug 9 13:30:39 2010 by HappyDoc version r1_5

Table of Contents

Module: madTimeCalculator **madTimeCalculator.py**

__init__ run the entire traceMagneticFieldService script. All other functions are private and called by __init__

madTimeCalculator.py is a script run to return a text output of derived Madrigal parameters for parameters that depend only on time.

It is presently used by the cgi script madTimeCalculatorService.py.

It has the following input arguments:

--date1=<MM/DD/YYYY> (required)

--time1=<HH:MM:SS> (required)

--date2=<MM/DD/YYYY> (required)

--time2=<HH:MM:SS> (required)

--stepHours=<step hours between each calculation (double)> (required)

--parms=<comma delimited string of Madrigal parameters desired> (required)

--showHeader (optional) Prints header line before data

Returns comma-delimited data, one line for each time with the following fields:

1. year
2. month
3. day
4. hour
5. min
6. sec
7. Values for each Madrigal parameter listed in argument parms, separated by whitespace

If --showHeader given, also prints header line before data

This script, and the corresponding cgi script madTimeCalculatorService.py are available to any scripting language that wants to access Madrigal derived parameters.

Functions

isnan

isnan

```
isnan ( x )
```

Table of Contents

This document was automatically generated on Fri Dec 30 10:17:51 2005 by HappyDoc version r1_5

__init__ run the entire traceMagneticFieldService script. All other functions are private and called by __init__

Matlab Madrigal remote data access API reference guide

The following are the methods for accessing Madrigal data remotely:

- getMadrigalCgiUrl - converts the Madrigal url to the cgi form required by the other methods
- getInstrumentsWeb - returns a list of all instruments in Madrigal database
- getExperimentsWeb - returns a list of experiments for a given instument(s) and date range
- getCgiurlForExperiment - returns cgiurl of experiment.url as returned by getExperimentsWeb
- getExperimentFilesWeb - returns a list of files in a given experiment
- getParametersWeb - returns a list of measured parameters in a file, and derived parameters available
- isprintWeb - returns data from a file as an array of doubles using user specified parameters and filters
- isprintUnguarded - similar to isprintWeb, except does not protect the user against requesting too much data for urlread
- madDownloadFile - downloads a Madrigal file to local computer in various formats
- madCalculatorWeb - returns derived parameters for a given time and set of spatial locations
- globalIsprint - returns user-specified data from multiple experiments

A good way to learn how to use this Matlab API is to run this example.

```
   ************************************************************
function cgiUrl = getMadrigalCgiUrl(url)
  getMadrigalCgiUrl     parse the main madrigal page to get the cgi url

  input: url to Madrigal

  output: cgi url for that Madrigal Site

  Note: parses the homepage for the accessData link


   ************************************************************
function instArray = getInstrumentsWeb(cgiurl)
  getInstrumentsWeb     returns an array of instrument structs of instruments found on remote M

  inputs:  cgiurl (string) to Madrigal site cgi directory
      (Example: 'http://www.haystack.mit.edu/cgi-bin/madrigal/')
      Note that method getMadrigalCgiUrl converts homepage url into cgiurl.

  output:
     instArray - array of instrument structs found

       instrument struct has the fields:

     instrument.name (string) Example: 'Millstone Hill Incoherent Scatter Radar'
     instrument.code (int) Example: 30
     instrument.mnemonic (3 char string) Example: 'mlh'
     instrument.latitude (double) Example: 45.0
     instrument.longitude (double) Example: 110.0
     instrument.altitude (double)  Example: 0.015 (km)

  Raises error if unable to return instrument array

  Example:
  getInstrumentsWeb('http://www.haystack.mit.edu/cgi-bin/madrigal/')
```

```
   ***********************************************************
function expArray = getExperimentsWeb(cgiurl, instCodeArray, starttime, endtime, localFlag)
  getExperimentsWeb     returns an array of experiment structs given input filter arguments fro

  Inputs:

      1. cgiurl (string) to Madrigal site cgi directory
        (Example: 'http://www.haystack.mit.edu/cgi-bin/madrigal/')
         Note that method getMadrigalCgiUrl converts homepage url into cgiurl.

      2. instCodeArray - a 1 X N array of ints containing selected instrument codes.  Special v

      3. starttime - Matlab datenum double (must be UTC)

      4. endtime - Matlab datenum double (must be UTC)

      5. localFlag - 1 if local experiments only, 0 if all experiments

   Return array of Experiment struct (May be empty):

   experiment.id (int) Example: 10000111
   experiment.url (string) Example: 'http://www.haystack.mit.edu/cgi-bin/madtoc/1997/mlh/03dec9
   experiment.name (string) Example: 'Wide Latitude Substorm Study'
   experiment.siteid (int) Example: 1
   experiment.sitename (string) Example: 'Millstone Hill Observatory'
   experiment.instcode (int) Code of instrument. Example: 30
   experiment.instname (string) Instrument name. Example: 'Millstone Hill Incoherent Scatter Ra
   experiment.starttime (double) Matlab datenum of experiment start
   experiment.endtime (double) Matlab datenum of experiment end
   experiment.isLocal (int) 1 if local, 0 if not
   experiment.madrigalUrl (string) - home url of Madrigal site with this
       experiment. Example 'http://madrigal.haystack.mit.edu/madrigal'

  Raises error if unable to return experiment array

  Example: expArray = getExperimentsWeb('http://www.haystack.mit.edu/cgi-bin/madrigal/', ...
                                        30, datenum('01/01/1998'), datenum('12/31/1998'), 1);

   Note that if the returned
   experiment is not local, the experiment.id will be -1.  This means that you
   will need to call getExperimentsWeb a second time with the cgiurl of the
   non-local experiment (getCgiurlForExperiment(experiment.madrigalUrl)).  This is because
   while Madrigal sites share metadata about experiments, the real experiment ids are only
   known by the individual Madrigal sites.  See testMadmatlab.m
   for an example of this.



   ***********************************************************
function cgiurl = getCgiurlForExperiment(experiment)
  getCgiurlForExperiment        returns cgiurl of experiment.url as returned by getExperimentsW

  inputs:  experiment struct as returned by getExperimentsWeb or getExperiments.


  output:
     cgiurl of experiment

  Simply truncates experiment.url to remove /madtoc///

  Example: If expArray is the value returned in the getExperimentsWeb example, and
           expArray(1).url = 'http://madrigal.haystack.mit.edu/cgi-bin/madrigal/madtoc/1998/mlh
```

```
     getCgiurlForExperiment(expArray(1))

     returns:

          'http://madrigal.haystack.mit.edu/cgi-bin/madrigal/'


   ***********************************************************
function expFileArray = getExperimentFilesWeb(cgiurl, experimentId)
  getExperimentFilesWeb         returns an array of experiment file structs given experiment id

  Note that it is assumed that experiment is local to cgiurl.  If not,
  empty list will be returned. Use getCgiurlForExperiment to get the correct
  cgiurl for any given experiment struct.

  Inputs:

      1. cgiurl (string) to Madrigal site cgi directory that has that
      experiment.
        (Example: 'http://www.haystack.mit.edu/cgi-bin/madrigal/')
         Note that method getMadrigalCgiUrl converts homepage url into cgiurl.

      2. experiment id (int) as returned by getExperiments or
         getExperimentsWeb

   Return array of Experiment File struct (May be empty):

   file.name (string) Example '/opt/mdarigal/blah/mlh980120g.001'
   file.kindat (int) Kindat code.  Example: 3001
   file.kindatdesc (string) Kindat description: Example 'Basic Derived Parameters'
   file.category (int) (1=default, 2=variant, 3=history, 4=real-time)
   file.status (string)('preliminary', 'final', or any other description)
   file.permission (int)  0 for public, 1 for private

  Raises error if unable to return experiment file array

  Example: expFileArray =
  getExperimentFilesWeb('http://www.haystack.mit.edu/cgi-bin/madrigal/', 10001686);


   ***********************************************************
function parmArray = getParametersWeb(cgiurl, filename)
  getParametesWeb       returns an array of parameter structs given filename from a remote Madr

  Note that it is assumed that filename is local to cgiurl.  If not,
  empty list will be returned.

  Inputs:

      1. cgiurl (string) to Madrigal site cgi directory that has that
      filename.
        (Example: 'http://www.haystack.mit.edu/cgi-bin/madrigal/')
         Note that method getMadrigalCgiUrl converts homepage url into cgiurl.

      2. filename (string) as returned by getExperimentFiles or
         getExperimentFilesWeb

   Return array of Parameter struct:
```

```
      parameter.mnemonic (string) Example 'dti'
      parameter.description (string) Example:
         "F10.7 Multiday average observed (Ott) - Units: W/m2/Hz"
      parameter.isError (int) 1 if error parameter, 0 if not
      parameter.units (string) Example "W/m2/Hz"
      parameter.isMeasured (int) 1 if measured, 0 if derivable
      parameter.category (string) Example: "Time Related Parameter"
      parameter.isSure (int) 1 if can be found for all records, 0 if only
          for some records (implies not all records have same measured
          parameters)

  Raises error if unable to return parameter array

  Example: parmArray = getParametersWeb('http://www.haystack.mit.edu/cgi-bin/madrigal/', ...
                                        '/opt/madrigal/experiments/1998/mlh/07jan98/mil980107g.


   ********************************************************
function records = isprintWeb(cgiUrl, file, parms, user_fullname, user_email, user_affiliation,
  isprintWeb    Create an isprint-like 3D array of doubles via a command similar to the isprint

  The calling syntax is:

               [records] = isprintWeb(cgiurl, file, parms, user_fullname, user_email, user_aff

   where

     cgiurl (string) to Madrigal site cgi directory that has that
      filename.
        (Example: 'http://www.haystack.mit.edu/cgi-bin/madrigal/')
         Note that method getMadrigalCgiUrl converts homepage url into cgiurl.

     file is path to file
         (example = '/home/brideout/data/mlh980120g.001')

     parms is the desired parameters in the form of a comma-delimited
         string of Madrigal mnemonics (example = 'gdlat,ti,dti')

     user_fullname - is user name (string)

     user_email - is user email address (string)

     user_affiliation - is user affiliation (string)

     filters is the optional filters requested in exactly the form given in isprint
         command line (example = 'time1=15:00:00 date1=01/20/1998
                       time2=15:30:00 date2=01/20/1998 filter=ti,500,1000')
         See:  http://www.haystack.mit.edu/madrigal/ug_commandLine.html for details

     missing is an optional double to represent missing values.  Defaults to NaN

     assumed is an optional double to represent assumed values.  Defaults to NaN

     knownbad is an optional double to represent knownbad values.  Defaults to NaN

     The returned records is a three dimensional array of double with the dimensions:

         [Number of rows, number of parameters requested, number of records]

     If error or no data returned, will return error explanation string instead.
```

```
   Example: data = isprintWeb('http://www.haystack.mit.edu/cgi-bin/madrigal/', ...
                              '/opt/madrigal/experiments/1998/mlh/07jan98/mil980107g.001', ..
                              'gdlat,ti,dti', ...
                              'Bill Rideout', 'wrideout@haystack.mit.edu', 'MIT');

   For now avoids limits with urlread by only asking for maxRecs records at once.  Repeatedly
   calls isprintUnguarded, which is a method without any additional filtering.

    $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $


   ************************************************************
function records = isprintUnguarded(cgiUrl, file, parms, user_fullname, user_email, user_affili
  isprintUnguarded - Create an isprint-like 3D array of doubles via a command similar to the is

  This method differs from isprintWeb in that no protection against
  downloading too much data is provided.  If too much data is requested,
  urlread will fail.

  The calling syntax for this method is:

               [records] = isprintUnguarded(cgiurl, file, parms, user_fullname, user_email, us

   where

     cgiurl (string) to Madrigal site cgi directory that has that
      filename.
        (Example: 'http://www.haystack.mit.edu/cgi-bin/madrigal/')
         Note that method getMadrigalCgiUrl converts homepage url into cgiurl.

     file is path to file
         (example = '/home/brideout/data/mlh980120g.001')

     parms is the desired parameters in the form of a comma-delimited
         string of Madrigal mnemonics (example = 'gdlat,ti,dti')

     user_fullname - is user name (string)

     user_email - is user email address (string)

     user_affiliation - is user affiliation (string)

     filters is the optional filters requested in exactly the form given in isprint
         command line (example = 'time1=15:00:00 date1=01/20/1998
                       time2=15:30:00 date2=01/20/1998 filter=ti,500,1000')
         See:  http://www.haystack.mit.edu/madrigal/ug_commandLine.html for details

     missing is an optional double to represent missing values.  Defaults to NaN

     assumed is an optional double to represent assumed values.  Defaults to NaN

     knownbad is an optional double to represent knownbad values.  Defaults to NaN

     The returned records is a three dimensional array of double with the dimensions:

         [Number of rows, number of parameters requested, number of records]

     If error or no data returned, will return error explanation string instead.

   Example: data = isprintUnguarded('http://www.haystack.mit.edu/cgi-bin/madrigal/', ...
                              '/opt/madrigal/experiments/1998/mlh/07jan98/mil980107g.001', ..
```

```
                                'gdlat,ti,dti', ...
                                'Bill Rideout', 'wrideout@haystack.mit.edu', 'MIT');

   Unlike isprintWeb, no effort is made to protect the user from
   requesting too more data than the Matlab method urlread can handle

    $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $


   ********************************************************
function  result  = madDownloadFile(cgiurl,  fullFilename, outputFile, format )
madDownloadFile downloads a Madrigal file to local computer in various
formats

    The calling syntax is:
       result  = madDownloadFile(cgiurl,  fullFilename, outputFile, [ format ] )

 Inputs:

      1. cgiurl (string) to Madrigal site cgi directory
        (Example: 'http://www.haystack.mit.edu/cgi-bin/madrigal/')
         Note that method getMadrigalCgiUrl converts homepage url into
         cgiurl.

      2. fullFilename - file to download as returned by getExperimentFilesWeb.m

      3. outputFile - name to save new file to

      4. format - one of the following strings:
          'simple', 'cedar_ascii'
              'simple' is the default if not specified, and
              is recommended for all but the most advanced users
              because it is a simple ascii space delimited column
              format, and is the easiest to parse.  See Cedar
              standard for description of cedar_ascii

    $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $



   ********************************************************
function record = madCalculatorWeb(cgiUrl, ...
                                    time, ...
                                    startLat, ...
                                    endLat, ...
                                    stepLat, ...
                                    startLong, ...
                                    endLong, ...
                                    stepLong, ...
                                    startAlt, ...
                                    endAlt, ...
                                    stepAlt, ...
                                    parms)
  madCalculatorWeb      Create a matrix of doubles via a the Madrigal derivation engine for a t

  The calling syntax is:

                [record] = madCalculatorWeb(cgiurl, time, startLat, endLat, stepLat, startLong,
                                    startAlt, endAlt, stepAlt, parms)
```

```
   where

     cgiurl (string) to Madrigal site cgi directory that has that
      filename.
        (Example: 'http://www.haystack.mit.edu/cgi-bin/madrigal/')
         Note that method getMadrigalCgiUrl converts homepage url into cgiurl.

     time - Matlab datenum double (must be UTC) 7. startLat - Starting geodetic latitude, -90 t

     endLat - Ending geodetic latitude, -90 to 90 (required)

     stepLat - Latitude step (0.1 to 90) (required)

     startLong - Starting geodetic longitude, -180 to 180 (required)

     endLong - Ending geodetic longitude, -180 to 180 (required)

     stepLong - Longitude step (0.1 to 180) (required)

     startAlt - Starting geodetic altitude, >= 0 (required)

     endAlt - Ending geodetic altitude, > 0 (required)

     stepAlt - Altitude step (>= 0.1) (required)

     parms is the desired parameters in the form of a comma-delimited
         string of Madrigal mnemonics (example = 'gdlat,ti,dti')


     The returned record is a matrix of doubles with the dimensions:

         [(num lat steps * num long steps * num alt steps), 3 + num of parms]

     The first three columns will always be lat, long, and alt, so there are three
     additional columns to the number of parameters requested via the parms argument.

     If error or no data returned, will return error explanation string instead.

     Example: result = madCalculatorWeb('http:/grail.haystack.mit.edu/cgi-bin/madrigal', ...
                                        now,45,55,5,45,55,5,200,300,50,'bmag,bn');


   ************************************************************
function [] = globalIsprint(url, ...
    parms, ...
    output, ...
    user_fullname, ...
    user_email, ...
    user_affiliation, ...
    startTime, ...
    endTime, ...
    inst, ...
    filters, ...
    kindats, ...
    expName, ...
    fileDesc)
 globalIsprint is a script to search through the entire Madrigal database
 for appropriate data to print in ascii to a file

    Inputs:
```

```
      url - url to homepage of site to be searched (Example:
            'http://www.haystack.mit.edu/madrigal/'

      parms - a comma delimited string listing the desired Madrigal
          parameters in mnemonic form.
          (Example: 'year,month,day,hour,min,sec,gdalt,dte,te').
          Ascii space-separated data will be returned in the same
          order as given in this string. See
          http://madrigal.haystack.mit.edu/cgi-bin/madrigal/getMetadata
          "Parameter code table" for all possible parameters.

      output - the local file name to store the resulting ascii data.
               (Example: '/tmp/isprint.txt')

      user_fullname - the full user name (Example: 'Bill Rideout')

      user email -  Example: 'brideout@haystack.mit.edu'

      user_affiliation - Example: 'MIT'

      startTime - a Matlab time to begin search at. Example:
                  datenum('20-Jan-1998 00:00:00') Time in UT

      endTime - a Matlab time to end search at. Example:
                datenum('21-Jan-1998 23:59:59') Time in UT

      inst - instrument code (integer).  See
          http://madrigal.haystack.mit.edu/cgi-bin/madrigal/getMetadata
          "Instrument Table" for this list. Examples: 30 for Millstone
          Hill Incoherent Scatter Radar, 80 for Sondrestrom Incoherent
          Scatter Radar

  Optional inputs

      filters - is the optional filters requested in exactly the form given in isprint
       command line (example = 'filter=gdalt,,500 filter=ti,500,1000')
       See:  http://www.haystack.mit.edu/madrigal/ug_commandLine.html for details

      kindats - is an optional array of kindat (kinds of data) codes to accept.
         The default is an empty array, which will accept all kindats.

      expName - a case insensitive regular expression that matches the experiment
         name.  Default is zero-length string, which matches all experiment names.

      fileDesc - a case insensitive regular expression that matches the file description.
         Default is zero-length string, which matches all file descriptions.

  Returns: Nothing.

  Affects: Writes results to output file



Example: globalIsprint('http://www.haystack.mit.edu/madrigal/', ...
                       'year,month,day,hour,min,sec,gdalt,dte,te', ...
                       '/tmp/isprint.txt', ...
                       'Bill Rideout', ...
                       'brideout@haystack.mit.edu', ...
                       'MIT', ...
                       datenum('20-Jan-1998 00:00:00'), ...
                       datenum('21-Jan-1998 23:59:59'), ...
```

```
                        30);

  $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $
```

Example Matlab code use this API

```
  % demo program of madmatlab running on a pc or linux

madurl = 'http://madrigal.haystack.mit.edu/madrigal';

cgiurl = getMadrigalCgiUrl(madurl)

'List all instruments, and their latitudes and longitudes:'
instArray = getInstrumentsWeb(cgiurl);
for i = 1:length(instArray)
    [s,errmsg] = sprintf('Instrument: %s, at lat %f and long %f', ...
            instArray(i).name, ...
            instArray(i).latitude, ...
            instArray(i).longitude);
     s
end
% now list all experiments from local Madrigal site with mlh (code 30) in
% 1998 - should be data if default files installed...
startdate = datenum('01/01/1998');
enddate = datenum('12/31/1998');
expArray = getExperimentsWeb(cgiurl, 30, startdate, enddate, 1);
for i = 1:length(expArray)
    [s,errmsg] = sprintf('Experiment name: %s, at url %s and id %i', ...
            expArray(i).name, ...
            expArray(i).url, ...
            expArray(i).id);
     s
end

% now list all files in the first experiment
expFileArray = getExperimentFilesWeb(cgiurl, expArray(1).id);
for i = 1:length(expFileArray)
    [s,errmsg] = sprintf('File name: %s, with kindat %i', ...
            expFileArray(i).name, ...
            expFileArray(i).kindat);
     s
end
% now first 2 parameters in the last file
parmArray = getParametersWeb(cgiurl, expFileArray(end).name)
for i = 1:10
    [s,errmsg] = sprintf('Parameter mnemonic: %s, description "%s" -- isMeasured = %i', ...
            parmArray(i).mnemonic, ...
            parmArray(i).description, ...
            parmArray(i).isMeasured);
     s
end
%  run isprintWeb for that file for first two parameters
parmStr = sprintf('%s,%s', parmArray(1).mnemonic, parmArray(2).mnemonic);
data = isprintWeb(cgiurl, expFileArray(1).name, parmStr, 'Bill Rideout', 'wrideout@haystack.mit
% print first 10 records
data(:,:,1:10)

% download that data file in simple format
```

```
result = madDownloadFile(cgiurl, expFileArray(1).name, '/tmp/junk.txt');
'downloaded file to /tmp/junk.txt'

% run globalIsprint, which can gather data from multiple files at once
globalIsprint('http://www.haystack.mit.edu/madrigal/', ...
              'year,month,day,hour,min,sec,gdalt,dte,te', ...
              '/tmp/isprint.txt', ...
              'Bill Rideout', 'brideout@haystack.mit.edu', 'MIT', ...
              datenum('20-Jan-1998 00:00:00'), datenum('21-Jan-1998 23:59:59'), 30);
'globalIsprint output saved to /tmp/isprint.txt'

% madCalculatorWeb runs the Madrigal derivation engine for any point
data = madCalculatorWeb(cgiurl, datenum(1999,2,15,12,30,0), 45,55,5,-170,-150,10,200,200,0,'sdw
'madCalculator output'
% print data
data

'The following is an example of searching for non-local experiments'
% 61 is the instrument id of Poker Flat ISR - so this will return an
% experiment not from the Millstone Madrigal site
startdate = datenum('04/01/2008');
enddate = datenum('04/30/2008');
expArray = getExperimentsWeb(cgiurl, 61, startdate, enddate, 0);

% calling this now would fail: expFileArray = getExperimentFilesWeb(cgiurl, expArray(1).id);
% Instead, get the cgiurl of the non-local experiment
if (expArray(1).isLocal == 0)
    cgiurl = getMadrigalCgiUrl(expArray(1).madrigalUrl)
    expArray = getExperimentsWeb(cgiurl, 61, startdate, enddate, 0);
end

% now you can get the files
expFileArray = getExperimentFilesWeb(cgiurl, expArray(1).id);
for i = 1:length(expFileArray)
    [s,errmsg] = sprintf('File name: %s, with kindat %i', ...
            expFileArray(i).name, ...
            expFileArray(i).kindat);
     s
end
```

 Matlab remote API reference | Doc home | Madrigal home

Previous: Scripts supporting the cgi interface **Up:** Remote data access reference - toc **Next:** Python remote API reference

 Python remote API reference | Doc home | Madrigal home

Previous: Matlab remote API reference **Up:** Remote data access reference - toc **Next:** IDL remote API reference

Python remote data access API reference guide

This reference guide describes the madrigalWeb python module, which provides remote access to all of Madrigal's data and features.

HappyDoc Generated Documentation

Modules and Packages

madrigalWeb/

globalIsprint	This script runs a global search through Madrigal data from a given URL.
madrigalWeb	The madrigalWeb module provides access to all Madrigal data via web services.

madrigalWebPlot/

__init__	Python remote plotting API for Madrigal data
madrigalPlot	madrigalPlot is the module that produces plots of Madrigal data.

madrigalWebPlot/scripts/

madrigalPColor	madrigalPColor.py is a remote command-line application that creates PColor plots from Madrigal data
madrigalScatter	madrigalScatter.py is a remote command-line application that creates scatter plots from Madrigal data

Table of Contents

Module: globalIsprint **madrigalWeb/globalIsprint.py**

This script runs a global search through Madrigal data from a given URL.

This script is a stand-alone application, and can be run from anywhere with a connection to the internet. It runs on either unix or windows. It requires only the MadrigalWeb python module to be installed.

Usage:

globalIsprint --url=<Madrigal url> --parms=<Madrigal parms> --output=<output file> \
--user_fullname=<user fullname> --user_email=<user email> \ --user_affiliation=<user affiliation> [options]

where:

--url=<Madrigal url> - url to homepage of site to be searched (ie, http://madrigal.haystack.mit.edu/madrigal/) This is required.

--parms=<Madrigal parms> - a comma delimited string listing the desired Madrigal parameters in mnemonic form. (Example: gdalt,dte,te). Data will be returned in the same order as given in this string. See http://madrigal.haystack.mit.edu/cgi-bin/madrigal/getMetadata and choose "Parameter code table" for all possible parameters

--output=<output file name> - the file name to store the resulting data.

--user_fullname=<user fullname> - the full user name (probably in quotes unless your name is Sting or Madonna)

--user_email=<user email>

--user_affiliation=<user affiliation> - user affiliation. Use quotes if it contains spaces.

and options are:

--startDate=<MM/DD/YYY> - start date to filter experiments before. Defaults to allow all experiments.

--endDate=<MM/DD/YYY> - end date to filter experiments after. Defaults to allow all experiments.

--inst=<instrument list> - comma separated list of instrument codes or names. See Madrigal documentation for this list. Defaults to allow all instruments. If names are given, the argument must be enclosed in double quotes. An asterick will perform matching as in glob. Examples: (--inst=10,30 or --inst="Jicamarca IS Radar,Arecibo*")

--expName - filter experiments by the experiment name. Give all or part of the experiment name. Matching is case insensitive and fnmatch characters * and ? are allowed. Default is no filtering by experiment name.

--fileDesc - filter files by their file description string. Give all or part of the file description string. Matching is case insensitive and fnmatch characters * and ? are allowed. Default is no filtering by file description.

--kindat=<kind of data list> - comma separated list of kind of data codes. See Madrigal documentation for this list. Defaults to allow all kinds of data. If names are given, the argument must be enclosed in double quotes. An asterick will perform matching as in glob. Examples: (--kindat=3001,13201 or --kindat="INSCAL Basic Derived Parameters,*efwind*,2001")

--filter=<[mnemonic] or [mnemonic1,[+-*/]mnemonic2]>,<lower limit1>,<upper limit1>[or<lower limit2>,<upper limit2>...] a filter using any measured or derived Madrigal parameter, or two Madrigal parameters either added, subtracted, multiplied or divided. Each filter has one or more allowed ranges. The filter accepts data that is in any allowed range. If the Madrigal parameter value is missing, the filter will always reject that data. Multiple filter arguments are allowed on the command line. To skip either a lower limit or an upper limit, leave it blank. Examples: (--filter=ti,500,1000 (Accept when 500 <= Ti <= 1000) or --filter=gdalt,-,sdwht,0, (Accept when gdalt > shadowheight - that is, point in direct sunlight) or --filter=gdalt,200,300or1000,1200 (Accept when 200 <= gdalt <= 300 OR 1000 <= gdalt <= 1200))

--seasonalStartDate=<MM/DD> - seasonal start date to filter experiments before. Use this to select only part of the year to collect data. Defaults to Jan 1. Example: (--seasonalStartDate=07/01) would only allow experiments after July 1st from each year.

--seasonalEndDate=<MM/DD> - seasonal end date to filter experiments after. Use this to select only part of the year to collect data. Defaults to Dec 31. Example: (--seasonalEndDate=10/31) would only allow experiments before Oct 31 of each year.

--showFiles - if given, show file names. Default is to not show file names.

--showSummary - if given, summarize all arguments at the beginning. Default is to not show summary.

--includeNonDefault - if given, include all files, including history. Default is to search only default files.

--missing=<missing string> (defaults to "missing")

--assumed=<assumed string> (defaults to "assumed")

--knownbad=<knownbad string> (defaults to "knownbad")

--verbose - if given, print each file processed info to stdout. Default is to run silently.

$Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $

Functions

filterExperimentFilesUsingFileDesc
filterExperimentFilesUsingKindat
filterExperimentFilesUsingStatus
filterExperimentsUsingExpName
filterExperimentsUsingSeason
getExperimentFileList
getInstrumentList
getTimeParms
getTimesOfExperiment

filterExperimentFilesUsingFileDesc

`filterExperimentFilesUsingFileDesc ( expFileList, fileDesc )`

filterExperimentFilesUsingFileDesc returns a subset of the experiment files in expFileList with filtered using fileDesc string and case-insensitive fnmatch.

Input:

expFileList - a list of MadrigalExperimentFile objects to be filtered.

Returns:

a subset of expFileList with default status

filterExperimentFilesUsingKindat

`filterExperimentFilesUsingKindat ( expFileList, kindat )`

filterExperimentFilesUsingKindat returns a subset of the experiment files in expFileList whose kindat is found in kinda argument.

Input:

expFileList - a list of MadrigalExperimentFile objects to be filtered

kindat - the kindat argument passed in by the user - comma separated list of kind of da codes. If names are given, the argument must be enclosed in double quotes. An asteric will perform matching as in glob.

Returns:

a subset of expFileList whose kindat values are accepted

filterExperimentFilesUsingStatus

`filterExperimentFilesUsingStatus ( expFileList )`

filterExperimentFilesUsingStatus returns a subset of the experiment files in expFileList with default status.

Input:

expFileList - a list of MadrigalExperimentFile objects to be filtered.

Returns:

a subset of expFileList with default status

filterExperimentsUsingExpName

```
filterExperimentsUsingExpName ( expList,  expName )
```

filterExperimentsUsingExpName returns a subset of the experiments in expList whose name matches.

Input:

expList - a list of MadrigalExperiment objects to be filtered

expName - filter experiments by the experiment name. Can be all or part of the experiment name. Matching is case insensitive.

Returns:

a subset of expList whose names are accepted

filterExperimentsUsingSeason

```
filterExperimentsUsingSeason (
        expList,
        seasonalStartDate,
        seasonalEndDate,
        )
```

filterExperimentsUsingSeason returns a subset of the experiments in expList whose date is within the given seaso

Input:

expList - a list of MadrigalExperiment objects to be filtered

seasonalStartDate - in form MM/DD - seasonal start date to filter experiments before

seasonalEndDate - in form MM/DD - seasonal end date to filter experiments after

Returns:

a subset of expList whose times are accepted

Exceptions

'seasonalEndDate must be in form MM/DD: ' + str(seasonalEndDat
)
'seasonalStartDate must be in form MM/DD: ' + str(
seasonalStartDate)

getExperimentFileList

```
getExperimentFileList ( server,  expList )
```

filterExperimentFilesUsingStatus returns a subset of theexperiment files in expFileList with default status.

getExperimentFileList returns a list of MadrigalExperimentFi objects given an experiment list.

Inputs:

```
server - the active MadrigalData object to get information fr

expList - the list of desired MadrigalExperiment objects
```

Returns:

a list of MadrigalExperimentFile objects

getInstrumentList

```
getInstrumentList ( inst,  server )
```

getInstrumentList takes the user argument inst and coverts i into a list of instrument codes.

Inputs:

inst - a string containing a comma separated list of instrument codes or names. If name are given, the argument must be enclosed in double quotes. An asterick will perform matching as in glob. Both names and codes may be mixed together.

server - the active MadrigalData object to get information from

Returns:

a list of instrument codes (int). Instrument code 0 means all instruments

getTimeParms

```
getTimeParms (
        expTimeList,
        numIter,
        j,
        )
```

getTimeParms creates arguments to be passed to isprint to get only a slice of an experiment's data

Input:

expTimeList: a list of experiment start and end times:startyear, startmonth, startday, starthour, startmin, startsec, endyear, endmonth, endday, endhour, endmin, endsec

numIter - the number of pieces to break the experiment into

j - this iteration

Returns - a string in the form ' date1=01/20/1998 time1=09:00:00 date2=01/20/1998 time2=10:30:00 ' that will cause isprint to only examine a slice of the data.

getTimesOfExperiment

```
getTimesOfExperiment ( expList,  expId )
```

getTimesOfExperiment returns a list of the start and end time of the experiment given expId.

Input:

expList - the list of MadrigalExperiment objects

expId - the experiment id

Returns:

a list of: (startyear, startmonth, startday, starthour, startmin, startsec, endyear, endmon endday, endhour, endmin, endsec)

Table of Contents

This document was automatically generated on Fri Oct 21 16:17:08 2011 by HappyDoc version r1_5

Table of Contents

Module: madrigalWeb — **madrigalWeb/madrigalWeb.py**

The madrigalWeb module provides access to all Madrigal data via web services.

Classes

MadrigalData	MadrigalData is a class that acquires data from a particular Madrigal site.
MadrigalExperiment	MadrigalExperiment is a class that encapsulates information about a Madrigal Experiment.
MadrigalExperimentFile	MadrigalExperimentFile is a class that encapsulates information about a Madrigal ExperimentFile.
MadrigalInstrument	MadrigalInstrument is a class that encapsulates information about a Madrigal Instrument.

	MadrigalParameter	MadrigalParameter is a class that encapsulates information about a Madrigal Parameter.

Table of Contents

This document was automatically generated on Fri Oct 21 16:17:08 2011 by HappyDoc version r1_5

Table of Contents

Class:
MadrigalData

MadrigalData is a class that acquires data from a particular Madrigal si

Usage example:

```
import madrigalWeb.madrigalWeb

test =  madrigalWeb.madrigalWeb.MadrigalData('http://madrigal.haystack.m

instList = test.getInstrumentList()
```

Non-standard Python modules used: None

Change history:

Written by Bill Rideout Feb. 10, 2004

Methods

__getSiteDict	getVersion
__getSiteId	isprint
__init__	listFileTimes
compareVersions	madCalculator
downloadFile	madCalculator2
geodeticToRadar	madCalculator3
getAllInstruments	madTimeCalculator
getExperimentFileParameters	radarToGeodetic
getExperimentFiles	simplePrint
getExperiments	traceMagneticField

__getSiteDict

```
__getSiteDict ( self )
```

__getSiteDict returns a dictionary with key = site id, value= s

Uses getMetadata cgi script

__getSiteId

```
__getSiteId ( self )
```

__getSiteId returns the local site id

Uses getMetadata cgi script

Exceptions

IOError, 'No siteId found'

__init__

```
__init__ ( self,  url )
```

__init__ initializes a MadrigalData object.

Inputs:

```
            url - (string) url of main page of madrigal site. Example
```

Affects: Converts main page to cgi url, and stores that.

Also stores self.siteDict, with key = site id, value= site url, and self.siteId

Exceptions: If url not found.

Exceptions

ValueError, 'invalid url: ' + str(url)
ValueError, 'unable to open url ' + str(url)

compareVersions

```
compareVersions (
        self,
        ver1,
        ver2,
        )
```

compareVersions returns -1 if ver1 < ver2, 0 if equal, 1 if ver1

Inputs: version number strings, in form number dot number (any number of dots)

downloadFile

```
downloadFile (
        self,
        filename,
        destination,
        format='simple',
        )
```

downloadFile will download a Cedar file in the specified form

Inputs:

filename - The absolute filename to as returned via getExperimentFiles.

destination - where the file is to be stored

format - file format desired. May be `simple`, `hdf5` , `madrigal`, `blockedBinary`
`simple` Simple is a simple ascii space delimited column format. Simple and hdf5 are

hdf5 format works for Madrigal 2.6 or later

Exceptions	
	IOError, 'downloadFile with hdf5 format requires Madrig self._madVers) ValueError, 'Illegal format specified: %s' %(str(format)

geodeticToRadar

```
geodeticToRadar (
        self,
        slatgd,
        slon,
        saltgd,
        gdlat,
        glon,
        gdalt,
        )
```

geodeticToRadar converts arrays of points in space to az, el,

Input arguments:

1. slatgd - radar geodetic latitude
2. slon - radar longitude
3. saltgd - radar altitude
4. gdlat - either a single geodetic latitude, or a list of geodetic latitudes
5. glon - either a single longitude, or a list of longitudes. If so, len(gdlat) must =
6. gdalt - either a single deodetic altitude, or a list of geodetic altitudes. If so, len

Returns:

A list of lists, where each list contains 3 floats (az, el, and range)

Exceptions

ValueError, 'No data found at url' + str(url)
ValueError, 'all lists most have same length'
ValueError, 'error raised using url ' + str(url) + ' ' + str(p
ValueError, 'unable to open url ' + str(url)

getAllInstruments

```
getAllInstruments ( self )
```

returns a list of all MadrigalInstruments at the given Madrigal site

Exceptions

ValueError, 'No data found at url' + str(url)
ValueError, 'error raised using url ' + str(url) + ' ' + str(p
ValueError, 'unable to open url ' + str(url)

getExperimentFileParameters

```
getExperimentFileParameters ( self,  fullFilename )
```

getExperimentFileParameters returns a list of all measured a

Inputs:

fullFilename - full path to experiment file as returned by getExperimentFiles.

Outputs:

List of MadrigalParameter objects for that fullFilename. Includes both measured and d

Exceptions

ValueError, 'error raised using url ' + str(url) + ' ' + str(p
ValueError, 'unable to open url ' + str(url)

getExperimentFiles

```
getExperimentFiles (
        self,
        id,
        getNonDefault=False,
        )
```

returns a list of all default MadrigalExperimentFiles for a give

Inputs:

id - Experiment id.

getNonDefault - if False (the default), only get default files, or realtime files if no defa

Outputs:

List of MadrigalExperimentFile objects for that experiment id

Exceptions

ValueError, """Illegal experiment id -1. This is usually c
set to 0. To get the experiment id for a non-local experim
with the url of the non-local experiment (MadrigalExperi
second time using that Madrigal url. This is because whil
real experiment ids are only known by the individual Ma
examples/exampleMadrigalWebServices.py for an exam
ValueError, 'error raised using url ' + str(url) + ' ' + str(
ValueError, 'unable to open url ' + str(url)

getExperiments

```
getExperiments (
        self,
        code,
        startyear,
        startmonth,
        startday,
        starthour,
        startmin,
        startsec,
        endyear,
        endmonth,
        endday,
        endhour,
        endmin,
        endsec,
        local=1,
        )
```

returns a list of all MadrigalExperiments that meet criteria at

Inputs:

code - int or list of ints representing instrument code(s). Special value of 0 selects all i

startyear - int or string convertable to int

startmonth - int or string convertable to int

startday - int or string convertable to int

starthour - int or string convertable to int

startmin - int or string convertable to int

startsec - int or string convertable to int

endyear - int or string convertable to int

endmonth - int or string convertable to int

endday - int or string convertable to int

endhour - int or string convertable to int

endmin - int or string convertable to int

endsec - int or string convertable to int

local - 0 if all sites desired, 1 (default) if only local experiments desired

Outputs:

List of MadrigalExperiment objects that meet the criteria. Note that if the returned Ma
-1. This means that you will need to create a new MadrigalData object with the url of t
(MadrigalExperiment.madrigalUrl), and then call getExperiments a second time using
share metadata about experiments, the real experiment ids are only known by the indiv
examples/exampleMadrigalWebServices.py for an example of this.

Exceptions

ValueError, 'error raised using url ' + str(url) + ' ' + str(p
ValueError, 'unable to open url ' + str(url)

getVersion

```
getVersion ( self )
```

getVersion gets the version number of Madrigal in form num

Assumes version is 2.5 if no getVersionService.py installed

isprint

```
isprint (
        self,
        filename,
        parms,
        filters,
        user_fullname,
```

```
        user_email,
        user_affiliation,
        )
```

returns as a string the isprint output given filename, parms, f

Inputs:

filename - The absolute filename to be analyzed by isprint.

parms - Comma delimited string listing requested parameters (no spaces allowed).

filters - Space delimited string listing filters desired, as in isprint command

user_fullname - full name of user making request

user_email - email address of user making request

user_affiliation - affiliation of user making request

Returns:

a string holding the isprint output

Exceptions

ValueError, 'error raised using url ' + str(url)
ValueError, 'unable to open url ' + str(url)

listFileTimes

```
listFileTimes ( self,  expDir=None )
```

listFileTimes lists the filenames and last modification times f

Inputs: expDir - experiment directory to which to start. May be any directory or subdir
or relative to experiments[0-9]*. If None (the default), include entire experiments[0-9]
(`/opt/madrigal/experiments/1998`, `experiments/2002/gps`)

Returns: a list of tuple of 1. filename relative to experiments[0-9]* directory, and 2) da

Requires: Madrigal 2.6 or greater

Exceptions

IOError, 'listFileTimes requires Madrigal 2.6 or greater, t

madCalculator

```
madCalculator (
        self,
        year,
        month,
        day,
        hour,
        min,
        sec,
        startLat,
        endLat,
        stepLat,
        startLong,
        endLong,
        stepLong,
        startAlt,
        endAlt,
        stepAlt,
        parms,
        oneDParmList=[],
        oneDParmValues=[],
        )
```

Input arguments:

1. year - int
2. month - int
3. day - int
4. hour - int
5. min - int
6. sec - int
7. startLat - Starting geodetic latitude, -90 to 90 (float)
8. endLat - Ending geodetic latitude, -90 to 90 (float)
9. stepLat - Latitude step (0.1 to 90) (float)
10. startLong - Starting geodetic longitude, -180 to 180 (float)
11. endLong - Ending geodetic longitude, -180 to 180 (float)
12. stepLong - Longitude step (0.1 to 180) (float)
13. startAlt - Starting geodetic altitude, >= 0 (float)
14. endAlt - Ending geodetic altitude, > 0 (float)
15. stepAlt - Altitude step (>= 0.1) (float)
16. parms - comma delimited string of Madrigal parameters desired
17. oneDParmList - a list of one-D parameters whose values should be set for the
    empty list.
18. oneDParmValues - a list of values (doubles) associated with the one-D parame

Returns:

A list of lists of doubles, where each list contains 3 + number of parameters doubles. T
and altitude. The rest of the doubles are the values of each of the calculated values. If t

Example:

result = testData.madCalculator(1999,2,15,12,30,0,45,55,5,-170,-150,10,200,200,0,'bn

result = [[45.0, -170.0, 200.0, 4.1315700000000002e-05, 2.1013500000000001e-05] [45.0, -150.0, 200.0, 4.3856400000000002e-05, 1.97411e-05] [50.0, -170.0, 200.0, 4.6337800000000002e-05, 1.80077e-05] [55.0, -170.0, 200.0, 4.6397899999999998e-05, 1.6932500000000001e-05] [55.0, -150.0, 200.0, 4.85495e-05, 1.7265400000000003e-05, 1.6932500000000001e-05] [55.0, -150.0, 200.0, 4.85495e-

Columns: gdlat glon gdalt bmag bn

Exceptions

ValueError, 'No data found at url' + str(url)
ValueError, 'error raised using url ' + str(url) + ' ' + str(
ValueError, 'len(oneDParmList) != len(oneDParmValues
ValueError, 'unable to open url ' + str(url)

madCalculator2

```
madCalculator2 (
        self,
        year,
        month,
        day,
        hour,
        min,
        sec,
        latList,
        lonList,
        altList,
        parms,
        oneDParmList=[],
        oneDParmValues=[],
        twoDParmList=[],
        twoDParmValues=[],
        )
```

madCalculator2 is similar to madCalculator, except that a random collection of points
Also, a user can input 2D data.

Added to Madrigal2.6 as web service - will not run on earlier Madrigal installations.

Input arguments:

1. year - int
2. month - int
3. day - int
4. hour - int
5. min - int
6. sec - int
7. latList - a list of geodetic latitudes, -90 to 90
8. lonList - a list of longitudes, -180 to 180. Length must = lats
9. altList - a list of geodetic altitudes in km. Length must = lats
10. parms - comma delimited string of Madrigal parameters desired

11. oneDParmList - a list of one-D parameters whose values should be set for the
    empty list.
12. oneDParmValues - a list of values (doubles) associated with the one-D parame
13. twoDParmList - a python list of two-D parameters as mnemonics. Defaults to
14. twoDParmValues - a python list of lists of len = len(twoDParmList). Each ind
    two-D parameter set in twoDParmList, with a length = number of points (or ec

Returns:

A list of lists of doubles, where each list contains 3 + number of parameters doubles. T
and altitude. The rest of the doubles are the values of each of the calculated values. If t

Example:

result = testData.madCalculator2(1999,2,15,12,30,0,[45,55],[-170,-150],[200,300],'sdv

result = [[1999.0, 2.0, 15.0, 12.0, 30.0, 0.0, 3.0, 15.0] [1999.0, 2.0, 15.0, 12.0, 45.0, 0.
[1999.0, 2.0, 15.0, 13.0, 15.0, 0.0, 3.0, 15.0]]

Columns: gdlat glon gdalt sdwht kp

Now uses POST to avoid long url issue

Exceptions

	IOError, 'madCalculator2 requires Madrigal 2.6 or greate
	ValueError, 'No data found at url' + str(url)
	ValueError, 'error raised using url ' + str(url) + ' ' + str(
	ValueError, 'len(oneDParmList) != len(oneDParmValues
	ValueError, 'length of latList must be at least one'
	ValueError, 'lengths of latList, lonList, altList must all be
	ValueError, 'unable to open url ' + str(url)

madCalculator3

```
madCalculator3 (
        self,
        yearList,
        monthList,
        dayList,
        hourList,
        minList,
        secList,
        latList,
        lonList,
        altList,
        parms,
        oneDParmList=[],
        oneDParmValues=[],
        twoDParmList=[],
        twoDParmValues=[],
        )
```

madCalculator3 is similar to madCalculator, except that multiple times can be specifie
positions and 1D and 2D parms. It is equivalent to multiple calls to madCalculator2, ex
multiple calls to madCalculator2 are required for different times. The only restriction i
time.

Added to Madrigal2.6 as web service - will not run on earlier Madrigal installations.

Now uses POST to send arguments, due to large volume of data possible

Input arguments:

1. yearList - a list of years, one for each time requested (ints)
2. monthList - a list of months, one for each time requested. (ints)
3. dayList - a list of days, one for each time requested. (ints)
4. hourList - a list of hours, one for each time requested. (ints)
5. minList - a list of minutes, one for each time requested. (ints)
6. secList - a list of seconds, one for each time requested. (ints)
7. latList - a list of lists of geodetic latitudes, -90 to 90. The first list is for the firs
   not need to have the same number of points. The number of times must match
   latList[timeIndex][positionIndex]
8. lonList - a list of lists of longitudes, -180 to 180. The first list is for the first tir
   need to have the same number of points. The number of times must match yea
   lonList[timeIndex][positionIndex]
9. altList - a list of lists of geodetic altitudes in km. The first list is for the first tir
   need to have the same number of points. The number of times must match yea
   altList[timeIndex][positionIndex]
10. parms - comma delimited string of Madrigal parameters desired
11. oneDParmList - a list of one-D parameters whose values should be set for the
    empty list.
12. oneDParmValues - a list of lists of values (doubles) associated with the one-D
    empty list. The first list is for the first 1D parameter in oneDParmList, and mu
    second parameter, etc. Data organization: onDParmValues[parameterIndex][ti
13. twoDParmList - a python list of of two-D parameters as mnemonics. Defaults
14. twoDParmValues - a list of lists of lists of values (doubles) associated with the
    Defaults to empty list. The first list is for the first 2D parameter in oneDParmL
    in that list must be of len(num positions for that time). The second list is for th
    twoDParmValues[parameterIndex][timeIndex][positionIndex]

Returns:

A list of lists of doubles, where each list contains 9 + number of parameters doubles. T
hour, 5) minute, 6) second, 7) input latitude, 8) longitude, and 9) altitude. The rest of t
values. If the value cannot be calculated, it will be set to nan.

Example:

testData.madCalculator3(yearList=[2001,2001], monthList=[3,3], dayList=[19,20], ho
latList=[[45,46,47,48.5],[46,47,48.2,49,50]], lonList=[[-70,-71,-72,-73],[-70,-71,-72,-7
altList=[[145,200,250,300.5],[200,250,300,350,400]], parms='bmag,pdcon,ne_model',
oneDParmValues=[[31.0,31.0],[45.0,50.0]], twoDParmList=['ti','te','ne'],

twoDParmValues=[[[1000,1000,1000,1000],[1000,1000,1000,1000,1000]], [[1100,12
[[1.0e10,1.0e10,1.0e10,1.0e10],[1.0e10,1.0e10,1.0e10,1.0e10,1.0e10]]])

Columns: year month day hour minute second gdlat glon gdalt bmag pdcon ne_model

Exceptions

IOError, 'madCalculator3 requires Madrigal 2.6 or greate
site is version %s' %(self._madVers)
ValueError, 'No data found at url' + str(url)
ValueError, 'Not all time lists have same length'
ValueError, 'error raised using url ' + str(url) + ' ' + str(
ValueError, 'len(oneDParmList) != len(oneDParmValue
ValueError, 'mismatched number of points in altList'
ValueError, 'mismatched number of points in latList'
ValueError, 'mismatched number of points in lonList'
ValueError, 'unable to open url ' + str(url)
ValueError, 'wrong number of 1D parms for %s' %(str(p

madTimeCalculator

```
madTimeCalculator (
        self,
        startyear,
        startmonth,
        startday,
        starthour,
        startmin,
        startsec,
        endyear,
        endmonth,
        endday,
        endhour,
        endmin,
        endsec,
        stephours,
        parms,
        )
```

Input arguments:

1. startyear - int
2. startmonth - int
3. startday - int
4. starthour - int
5. startmin - int
6. startsec - int
7. endyear - int
8. endmonth - int
9. endday - int
10. endhour - int
11. endmin - int

12. endsec - int
13. stephours - float - number of hours per time step
14. parms - comma delimited string of Madrigal parameters desired (must not dep

Returns:

A list of lists, where each list contains 6 ints (year, month, day, hour, min, sec) + numb
will be set to nan.

Example:

result = testData.madTestCalculator(1999,2,15,12,30,0, 1999,2,20,12,30,0, 24.0, kp,

result = [[1999.0, 2.0, 15.0, 12.0, 30.0, 0.0, 3.0, -9.0] [1999.0, 2.0, 16.0, 12.0, 30.0, 0.0
-31.0] [1999.0, 2.0, 18.0, 12.0, 30.0, 0.0, 6.7000000000000002, -93.0] [1999.0, 2.0, 19

Columns: year, month, day, hour, min, sec, kp, dst

Exceptions

ValueError, 'No data found at url' + str(url)
ValueError, 'error raised using url ' + str(url) + ' ' + str(p
ValueError, 'unable to open url ' + str(url)

radarToGeodetic

```
radarToGeodetic (
        self,
        slatgd,
        slon,
        saltgd,
        az,
        el,
        radarRange,
        )
```

radarToGeodetic converts arrays of az, el, and ranges to geo

Input arguments:

1. slatgd - radar geodetic latitude
2. slon - radar longitude
3. saltgd - radar altitude
4. az - either a single azimuth, or a list of azimuths
5. el - either a single elevation, or a list of elevations. If so, len(el) must = len(az)
6. radarRange - either a single range, or a list of ranges. If so, len(radarRange) m

Returns:

A list of lists, where each list contains 3 floats (gdlat, glon, and gdalt)

Exceptions

ValueError, 'No data found at url' + str(url)
ValueError, 'all lists most have same length'
ValueError, 'error raised using url ' + str(url) + ' ' + str(p
ValueError, 'unable to open url ' + str(url)

simplePrint

```
simplePrint (
        self,
        filename,
        user_fullname,
        user_email,
        user_affiliation,
        )
```

simplePrint prints the data in the given file is a simple ascii f

simplePrint prints only the parameters in the file, without filters or derived parameters parameters, or to filter the data, use isprint instead.

Inputs:

filename - The absolute filename to be printed. Returned by getExperimentFiles.

user_fullname - full name of user making request

user_email - email address of user making request

user_affiliation - affiliation of user making request

Returns: string representing all data in the file in ascii, space-delimited form. The first parameters will always be year, month, day, hour, min, sec, representing the middle ti

traceMagneticField

```
traceMagneticField (
        self,
        year,
        month,
        day,
        hour,
        minute,
        second,
        inputType,
        outputType,
        alts,
        lats,
        lons,
        model,
        qualifier,
        stopAlt=None,
        )
```

traceMagneticField returns a point along a magnetic field line for each point specified
point, 2) intersection with a given altitude in the northern or southern hemisphere, 3) t
qualifier argument. Uses Tsyganenko or IGRF fields, depending on model argument. I
on inputType argument. Output arguments are either GSM or Geodetic, depending on

Inputs:

year, month, day, hour, minute, second - time at which to do the trace

inputType - 0 for geodetic, 1 for GSM

outputType - 0 for geodetic, 1 for GSM

The following parameter depend on inputType:

alts - a list of geodetic altitudes or ZGSMs of starting point

lats - a clist of geodetic latitudes or XGSMs of starting point

lons - a list of longitude or YGSM of starting point

Length of all three lists must be the same

model - 0 for Tsyganenko, 1 for IGRF

qualifier - 0 for conjugate, 1 for north_alt, 2 for south_alt, 3 for apex, 4 for GSM XY p

stopAlt - altitude in km to stop trace at, if qualifier is north_alt or south_alt. If other qu
which will raise exception if qualifier is north_alt or south_alt

Returns a tuple of tuples, one tuple for point in (alts, lats, lons) lists, where each tuple

1. geodetic altitude or ZGSM of ending point
2. geodetic latitude or XGSM of ending point
3. longitude or YGSM of ending point

If error, traceback includes error description

Exceptions	
	ValueError, 'No data found at url' + str(url) ValueError, 'error raised using url ' + str(url) + ' ' + str(ValueError, 'stopAlt must be set for qualifer in (1,2)' ValueError, 'unable to open url ' + str(url)

Table of Contents

Class: MadrigalExperiment

MadrigalExperiment is a class that encapsulates information abo

Attributes:

```
id (int) Example: 10000111.  Uniquely identifies experiment.

url (string) Example: 'http://madrigal.haystack.mit.edu/cgi-bin/ma

name (string) Example: 'Wide Latitude Substorm Study'

siteid (int) Example: 1

sitename (string) Example: 'Millstone Hill Observatory'

instcode (int) Code of instrument. Example: 30

instname (string) Instrument name. Example: 'Millstone Hill Incohe

startyear - int

startmonth - int

startday - int

starthour - int

startmin - int

startsec - int

endyear - int

endmonth - int

endday - int

endhour - int

endmin - int

endsec - int

isLocal - True if a local experiment, False if not

madrigalUrl - url of Madrigal site.  Used if not a local experimen
```

Non-standard Python modules used: None

Change history:

Written by Bill Rideout Feb. 10, 2004

Methods

__cmp__
__init__
__str__

__cmp__

```
__cmp__ ( self,  other )
```

__cmp__ compares two MadrigalExperiment objects.

Compared by start time, then by end time.

__init__

```
__init__ (
        self,
        id,
        url,
        name,
        siteid,
        sitename,
        instcode,
        instname,
        startyear,
        startmonth,
        startday,
        starthour,
        startmin,
        startsec,
        endyear,
        endmonth,
        endday,
        endhour,
        endmin,
        endsec,
        isLocal,
        madrigalUrl,
        )
```

__init__ initializes a MadrigalExperiment.

Inputs:

```
            id (int, or string that can be converted) Example:

            url (string) Example: 'http://madrigal.haystack.mit

            name (string) Example: 'Wide Latitude Substorm Stud

            siteid (int, or string that can be converted) Examp

            sitename (string) Example: 'Millstone Hill Observat

            instcode (int, or string that can be converted) Cod
```

```
instname (string) Instrument name. Example: 'Millst
startyear - int, or string that can be converted
startmonth - int, or string that can be converted
startday - int, or string that can be converted
starthour - int, or string that can be converted
startmin - int, or string that can be converted
startsec - int, or string that can be converted
endyear - int, or string that can be converted
endmonth - int, or string that can be converted
endday - int, or string that can be converted
endhour - int, or string that can be converted
endmin - int, or string that can be converted
endsec - int, or string that can be converted
isLocal - True if a local experiment, False if not
madrigalUrl - url of Madrigal site.  Used if not a
```

Returns: void

Affects: Initializes all the class member variables.

Exceptions: If illegal argument passed in.

Exceptions	
	ValueError, 'In MadrigalExperiment, instname not s ValueError, 'In MadrigalExperiment, isLocal not bo ValueError, 'In MadrigalExperiment, madrigalUrl n ValueError, 'In MadrigalExperiment, name not strin ValueError, 'In MadrigalExperiment, sitename not s ValueError, 'In MadrigalExperiment, url not string t

__str__

```
__str__ ( self )
```

return a readible form of this object

Table of Contents

This document was automatically generated on Fri Oct 21 16:17:08 2011 by HappyDoc version r1_5

Table of Contents

Class:
MadrigalExperimentFile

MadrigalExperimentFile is a class that encapsulates informatio ExperimentFile.

Attributes:

```
name (string) Example '/opt/mdarigal/blah/mlh980120g.001'

kindat (int) Kindat code.  Example: 3001

kindatdesc (string) Kindat description: Example 'Basic Derived

category (int) (1=default, 2=variant, 3=history, 4=real-time)

status (string)('preliminary', 'final', or any other descriptio

permission (int)  0 for public, 1 for private

expId - experiment id of the experiment this MadrigalExperiment
```

Non-standard Python modules used: None

Change history:

Written by Bill Rideout Feb. 10, 2004

Methods

__init__
__str__

__init__

```
__init__ (
        self,
        name,
        kindat,
        kindatdesc,
        category,
        status,
        permission,
        expId=None,
        )
```

__init__ initializes a MadrigalExperimentFile.

Inputs:

```
name - (string) Example '/opt/mdarigal/blah/mlh9
kindat - (int, or string that can be converted)
kindatdesc - (string) Kindat description: Exampl
category - (int, or string that can be converted
status - (string)('preliminary', 'final', or any
permission - (int, or string that can be convert
expId - experiment id of the experiment this Mad
```

Returns: void

Affects: Initializes all the class member variables.

Exceptions: If illegal argument passed in.

Exceptions	
	ValueError, 'In MadrigalExperimentFile, kinda ValueError, 'In MadrigalExperimentFile, name ValueError, 'In MadrigalExperimentFile, status

__str__

```
__str__ ( self )
```

return a readible form of this object

Table of Contents

This document was automatically generated on Wed Jul 2 15:58:53 2008 by HappyDoc version r1_5

Table of Contents

Class: MadrigalInstrument — **madrigalWeb/mad**

MadrigalInstrument is a class that encapsulates information abou Madrigal Instrument.

Attributes:

```
name (string) Example: 'Millstone Hill Incoherent Scatter Radar'
code (int) Example: 30
mnemonic (3 char string) Example: 'mlh'
latitude (double) Example: 45.0
```

```
longitude (double) Example: 110.0

altitude (double)  Example: 0.015 (km)
```

Non-standard Python modules used: None

Change history:

Written by Bill Rideout Feb. 10, 2004

Methods

__init__
__str__

__init__

```
__init__ (
        self,
        name,
        code,
        mnemonic,
        latitude,
        longitude,
        altitude,
        )
```

__init__ initializes a MadrigalInstrument.

Inputs:

```
name - (string) Example: 'Millstone Hill Incoherent

code - (int, or string that can be converted) Examp

mnemonic - (3 char string) Example: 'mlh'

latitude - (double, or string that can be converted

longitude (double, or string that can be converted)

altitude (double, or string that can be converted)
```

Returns: void

Affects: Initializes all the class member variables.

Exceptions: If illegal argument passed in.

Exceptions

ValueError, 'In MadrigalInstrument, mnemonic not string
mnemonic))
ValueError, 'In MadrigalInstrument, mnemonic not three
mnemonic))
ValueError, 'In MadrigalInstrument, name not string type

__str__

```
__str__ ( self )
```

return a readible form of this object

Table of Contents

This document was automatically generated on Wed Jul 2 15:58:53 2008 by HappyDoc version r1_5

Table of Contents

Class: MadrigalParameter

MadrigalParameter is a class that encapsulates information about

Attributes:

```
mnemonic (string) Example 'dti'

description (string) Example: &quot;F10.7 Multiday average observed

isError (int) 1 if error parameter, 0 if not

units (string) Example &quot;W/m2/Hz&quot;

isMeasured (int) 1 if measured, 0 if derivable

category (string) Example: &quot;Time Related Parameter&quot;

isSure (int) - 1 if parameter can be found for every record, 0 if c

isAddIncrement - 1 if additional increment, 0 if normal, -1 if unkn
                 only added with Madrigal 2.5)
```

Non-standard Python modules used: None

Change history:

Written by Bill Rideout Aug. 8, 2005

Methods

__init__
__str__

__init__ initializes a MadrigalInstrument.

__init__

```
__init__ (
        self,
        mnemonic,
        description,
        isError,
        units,
        isMeasured,
        category,
        isSure,
        isAddIncrement,
        )
```

__init__ initializes a MadrigalParameter.

Inputs:

```
mnemonic (string) Example 'dti'

description (string) Example: &quot;F10.7 Multiday a

isError (int) 1 if error parameter, 0 if not

units (string) Example &quot;W/m2/Hz&quot;

isMeasured (int) 1 if measured, 0 if derivable

category (string) Example: &quot;Time Related Parame

isSure (int) - 1 if parameter can be found for every

isAddIncrement - 1 if additional increment, 0 if nor
                 only added with Madrigal 2.5)
```

Returns: void

Affects: Initializes all the class member variables.

Exceptions: If illegal argument passed in.

Exceptions	
	ValueError, 'In MadrigalParameter, category not strin ValueError, 'In MadrigalParameter, description not st ValueError, 'In MadrigalParameter, mnemonic not str ValueError, 'In MadrigalParameter, units not string ty

__str__

```
__str__ ( self )
```

return a readible form of this object

Table of Contents

This document was automatically generated on Thu Jul 24 12:42:56 2008 by HappyDoc version r1_5

Table of Contents

Module: madrigalWebPlot/__init__.py

__init__

Python remote plotting API for Madrigal data

$Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $

Table of Contents

This document was automatically generated on Fri Jul 16 09:27:14 2010 by HappyDoc version r1_5

Table of Contents

Module: madrigalWebPlot/madrigalPlot.py

madrigalPlot

madrigalPlot is the module that produces plots of Madrigal data.

It is meant to be included as part of the Remote Python Madrigal API.

Presently based on madplotlib: http://matplotlib.sourceforge.net/

$Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $

Functions

convertToAbsoluteTimeStr
defineHomeEnvVariable
get_vo_cmap

convertToAbsoluteTimeStr

```
convertToAbsoluteTimeStr ( xticks,  noTime=False )
```

convertToAbsoluteTimeStr converts a list of strings containing seconds since 1/1/1950 to datetime string.

Input: xticks - a list of strings containing seconds since 1/1/1950

Returns: a list of strings formated as YYYY-MM-DD HH-MM-SS. If noTime, format as YYYY-MM-DD

defineHomeEnvVariable

```
defineHomeEnvVariable ()
```

defineHomeEnvVariable makes sure HOME env variable is defined, as required by matplotlib.

If not defined, sets HOME to PWD

get_vo_cmap

```
get_vo_cmap ()
```

get_vo_cmap is a function to return a colormap optimized to show sign changes in the middle of the range.

Classes

madPcolorPlot	madPcolorPlot is the class that produces pcolor plots of x versus y with z intensity.
madScatterPlot	madScatterPlot is the class the produces two dimensional scatter plots of x versus y.

Table of Contents

This document was automatically generated on Fri May 1 09:51:15 2009 by HappyDoc version r1_5

Table of Contents

Class: **madrigalWebPlot/madriga**

madPcolorPlot

madPcolorPlot is the class that produces pcolor plots of x versus y w intensity.

Assumes the x axis is time.

Usage example:

```
obj = madPcolorPlot(isprintText,
                    'Nel (log(m^-3)) - Millstone Hill - Oct. 30, 2003 - Alt
                    'Hours since midnight UT Oct. 30, 2003',
                    'Altitude (km)',
                    './isprint.png',
                    minColormap = 9,
                    maxColormap = 12,
                    smoothAltitude = False)
```

Non-standard Python modules used: matplotlib

Change history:

Written by Bill Rideout Mar. 31, 2005

Methods

__filter_missing

```
__filter_missing ( self,  x )
```

__getYIndex

```
__getYIndex ( self,  yvalue )
```

__getYIndex returns the correct index into the y dimen for a given y value.

Input: yvalue - value of y parameter

Returns: the correct index into the y dimension

Algorithm: if self.truncateAlt == False, simple use the dictionary self.yListD
Else loop through self.yListRanges and return the first greater than the reques
value

__init__

```
__init__ (
        self,
        isprintText,
        titleStr,
        xLabelStr,
        yLabelStr,
        fullFilename,
        minColormap=None,
        maxColormap=None,
        smoothAltitude=True,
        insertDataGap=5,
        useAbsoluteTime=False,
        startTime=None,
        endTime=None,
        sortTimeFlag=False,
        maxNumTimes=None,
        maxNumAlt=None,
        truncateIsprint=False,
        colorMap=matplotlib.cm.jet,
        yMinimum=None,
```

```
        yMaximum=None,
        altYTitle=None,
        altYLabels=None,
        )
```

__init__ writes a madPColorPlot to a file.

Inputs:

isprintText - a string giving isprint output without headers. First parameter m
UTH or UT1, depending on whether time scale should be relative to the expe
beginning (UTH) or absolute (UT1). The second must be gdalt, and third para
to be plotted.

titleStr - plot title (string) - should describe parameter being plott

xLabelStr - x label string

yLabelStr - ylabel string

fullFilename - full path of file containing pcolor plot to be saved. Extension r
jpeg or png, or exception thrown.

minColormap - minimum parameter value (defaults to lowest parameter valu

maxColormap - maximum parameter value (defaults to highest parameter val
However, if both minColormap and maxColormap == None, autoscaling app

smoothAltitude - if True, extrapolate between existing data between altitudes
in missing data; if False, leave missing

insertDataGap - this parameter sets the threshold for inserting a data gap. The
intervals being plotted are ordered, and the time gap larger than 90% of the re
determined. Any time interval more than insertDataGap times bigger is then
considered missing data. Defaults to five. If None, no gaps are ever inserted.
data with close to uniform time intervals, no gaps will be inserted.

useAbsoluteTime - if true, print time as YYYY-MM-DD HH:MM:SS. If fals
(default), print time as hour since beginning of experiment (UTH). If
useAbsoluteTime is true, first parameter in isprintText must be UT1, if false,
must be UTH.

startTime - start plot at given time. If useAbsoluteTime == True, then startTi
must be in units of seconds since 1/1/1950. If useAbsoluteTime == False, the
startTime must be in units of UTH (hours since midnight UT of first day of
experiment). Default is None, which means start at lowest time found.

endTime - end plot at given time. If useAbsoluteTime == True, then endTime
be in units of seconds since 1/1/1950. If useAbsoluteTime == False, then end

must be in units of UTH (hours since midnight UT of first day of experiment
Default is None, which means end at largest time found.

sortTimeFlag - if true, check that time is correctly sorted. If false (the default
assume time already sorted

maxNumTimes - if not None, decimate the number of times in the isprint stri
maxNumTimes. If None (the default), plot all times.

maxNumAlt - if not None, reduce the number of altitudes to maxNumAlt. If
(the default), plot all altitudes.

truncateIsprint - if True, and both maxNumTimes and maxNumAlt not = Nor
then truncate the number of isprint lines to be maxNumTimes * maxNumAlt

colorMap - sets colormap. It not given, defaults to matplotlib.cm.jet

yMinimum - minumum y value. If None (default), set by data y minimum.

yMaximum - maximum y value. If None (default), set by data y maximum.

altYTitle - title of right (second) y axis. If None (the default), no Y axis on th
side.

altYLabels - a list of Y labels (strings) for the right axis. If None (the default)
labels on the right axis.

Returns: void

Affects: None

Exceptions

	ValueError, 'No valid z data found' ValueError, 'input text is not parseable'

__truncateIsprint

```
__truncateIsprint (
        self,
        isprintText,
        maxLines,
        )
```

__truncateIsprint truncates isprintText to have maxLines at most.

_removeMissing

```
_removeMissing ( self,  isprintText )
```

returns modified isprintText with all missing data removed

decimateTimes

```
decimateTimes (
        self,
        array_data,
        maxNumTimes,
        insertDataGap,
        )
```

decimateTimes decimates array_data to have at most maxNumTimes times.

Input: array_data - two-dimensional array to be decimated by del times and missing data.

maxNumTimes: int representing the maximum number of times to keep in array_data

insertDataGap - this parameter sets the threshold for inserting a data gap. The intervals being plotted are ordered, and the time gap larger than 90% of the re determined. Note that this parameter is used here to stop the truncation of isp lines that will eventually be considered edge lines.

Returns: new array built from decimated array_data

displayToScreen

```
displayToScreen ( self )
```

to implement this takes a reworking away from pylab to use the underlying matplotlib code

getAverage

```
getAverage ( self,  X )
```

returns the average of items in a float array. Does not including missing data. data missing, returns self.__missing

getFigureHandle

```
getFigureHandle ( self )
```

sortArrayInTime

```
sortArrayInTime ( self,  array_data )
```

sortArrayInTime sorts a two-dimensional array so that first element in each row (time) is in ascending order.

Input: array_data - two-dimensional array to be sorted by rearranging rows so the first element in each row (time) is in ascending order

Returns: new_array

Table of Contents

This document was automatically generated on Thu Aug 25 15:33:04 2011 by HappyDoc version r1_5

Table of Contents

Class: madScatterPlot — **madrigalWebPlot/madrigalPlot.py**

madScatterPlot is the class the produces two dimensional scatter plots of x versus y.

Methods

__filter_missing

```
__filter_missing ( self,  x )
```

__init__

```
__init__ (
        self,
        isprintText,
        titleStr,
        xLabelStr,
        yLabelStr,
        fullFilename,
        useAbsoluteTime=False,
        startTime=None,
        endTime=None,
        maxNumPoints=None,
        yMin=None,
        yMax=None,
        )
```

__init__ writes a madScatter plot to a file.

Inputs:

isprintText - a string giving isprint output without headers. First parameter must be UTH or UT1, depending on whether time scale

should be relative to the experiment beginning (UTH) or absolute (UT1). The second must be the parameter to be plotted. Any missing data should be written as "missing" or other string that cannot be converted to a float.

titleStr - plot title (string) - should describe parameter being plotted

xLabelStr - x label string

yLabelStr - ylabel string

fullFilename - full path of file containing pcolor plot to be saved. Extension must be jpeg or png, or exception thrown.

useAbsoluteTime - if true, print time as YYYY-MM-DD HH:MM:SS. If false (default), print time as hour since beginning of experiment (UTH). If useAbsoluteTime is true, first parameter in isprintText must be UT1, if false, it must be UTH.

startTime - start plot at given time. If useAbsoluteTime == True, then startTime must be in units of seconds since 1/1/1950. If useAbsoluteTime == False, then startTime must be in units of UTH (hours since midnight UT of first day of experiment). Default is None, which means start at lowest time found.

endTime - end plot at given time. If useAbsoluteTime == True, then endTime must be in units of seconds since 1/1/1950. If useAbsoluteTime == False, then endTime must be in units of UTH (hours since midnight UT of first day of experiment). Default is None, which means end at largest time found.

maxNumPoints - maximum number of points to plot. If not None, truncate isprintText if needed to have at most maxNumPoints lines.

Returns: void

Affects: None

Exceptions

ValueError, 'No valid y data found'
ValueError, 'input text is not parseable'

__truncateIsprint

```
__truncateIsprint (
        self,
        isprintText,
        maxLines,
        )
```

__truncateIsprint truncates isprintText to have maxLines at most.

Table of Contents

This document was automatically generated on Fri May 1 09:51:15 2009 by HappyDoc version r1_5

Remote access using IDL

The following are the methods for accessing Madrigal data remotely:

- madgetallinstruments - returns an array of all instruments in Madrigal database
- madgetexperiments - returns an array of experiments for a given instument(s) and date range
- madgetexperimentfiles - returns an array of files in a given experiment
- madgetexperimentfileparameters - returns an array of measured parameters in a file, and derived parameters available
- madsimpleprint - returns all data in a file as a 2-D array
- madprint - returns data from a file as a 2-D array using user specified parameters and filters
- maddownloadfile - downloads a Madrigal file to local computer in a simple ascii format
- madglobalprint - returns user-specified data from multiple experiments
- madcalculator - returns derived parameters for a given time and set of spatial locations

```
   ************************************************************
;+
; NAME:
;    madgetallinstruments(madurl)
;
; PURPOSE:
;    Get information on all instruments with data at the Madrigal site specified
;    by madUrl
;
; INPUTS:
;      madurl - scalar string giving a fully qualified url to the Madrigal site
;
; OUTPUT: an array of structures with the following fields:
;               1. name - (string) Example: 'Millstone Hill Incoherent Scatter Radar'
;               2. code - (int) Example: 30
;               3. mnemonic - (3 char string) Example: 'mlh'
;               4. latitude - (double) Example: 45.0
;               5. longitude (double) Example: 110.0
;               6. altitude (double)  Example: 0.015 (km)
; EXAMPLE:
;     result = madGetAllInstruments('http://madrigal.haystack.mit.edu/madrigal')
;
; $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $
;
FUNCTION madgetallinstruments, madurl


   ************************************************************
;+
; NAME:
;    madgetexperiments(madurl, instcode, startyear, startmonth, startday, starthour, startmin,
;                                       endyear, endmonth, endday, endhour, endmin, endsec, loca
;
; PURPOSE:
;    Get information on experiments for a given instrument and date range
;
; INPUTS:
;      madurl - scalar string giving a fully qualified url to the Madrigal site
;      instcode = int or array of ints representing instrument code(s). Special value of 0 sele
```

```
;      startyear, startmonth, startday, starthour, startmin, startsec - 6 ints representing sta
;      endyear, endmonth, endday, endhour, endmin, endsec - 6 ints representing end time
;      local - 0 if all sites desired, 1 (default) if only local experiments desired
;
; OUTPUT: an array of structures representing experiments between the dates with the following
;   1. strid (string) Example: "10000111".  Uniquely identifies experiment. Can be converted to
;         Not stored as a long long because that just doesn't seem to work, so user needs to co
;   2. url (string) Example: 'http://www.haystack.mit.edu/cgi-bin/madtoc/1997/mlh/03dec97'
;   3. name (string) Example: 'Wide Latitude Substorm Study'
;   4. siteid (int) Example: 1
;   5. sitename (string) Example: 'Millstone Hill Observatory'
;   6. instcode (int) Code of instrument. Example: 30
;   7. instname (string) Instrument name. Example: 'Millstone Hill Incoherent Scatter Radar'
;   8. startyear, startmonth, startday, starthour, startmin, startsec - 6 ints representing sta
;   9. endyear, endmonth, endday, endhour, endmin, endsec - 6 ints representing end time of exp
;   10. isLocal - 1 if a local experiment, 0 if not
;   11. madrigalUrl - url of Madrigal site.  Used if not a local experiment.
;   If no experiements found, returns a Null Pointer (IDL does not allow empty arrays)
;
;    Note that if the returned experiment is not local, the experiment id will be -1.  This mea
;    will need to rerun madgetexperiments with the url of the
;    non-local experiment (experiment.madrigalUrl).
;    This is because
;    while Madrigal sites share metadata about experiments, the real experiment ids are only
;    known by the individual Madrigal sites.  See example_madidl.pro for an example of this
;           for an example of this.
; EXAMPLE:
;     result = madGetExperiments('http://madrigal.haystack.mit.edu/madrigal', 30, 1998,1,1,0,0,
;
; $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $
;
FUNCTION madGetExperiments, madurl,  instcode, startyear, startmonth, startday, starthour, star
                                     endyear, endmonth, endday, endhour, endmin, endsec, local


   ********************************************************
;+
; NAME:
;    madgetexperimentfiles(madurl, expid, getnondefault)
;
; PURPOSE:
;    returns a list of all  Madrigal Experiment Files for a given experiment id
;
; INPUTS:
;      madurl - scalar string giving a fully qualified url to the Madrigal site
;      expid - experiment id as returned by madgetexperiments
;      getnondefault - an optional argument.  If 1, get all files, including non-default.  If 0
;             get only default files
;
; OUTPUT: an array of structures representing experiment files for that experiment with the fol
;     1. name (string) Example '/opt/madrigal/blah/mlh980120g.001'
;     2. kindat (int) Kindat code.  Example: 3001
;     3. kindatdesc (string) Kindat description: Example 'Basic Derived Parameters'
;     4. category (int) (1=default, 2=variant, 3=history, 4=real-time)
;     5. status (string)('preliminary', 'final', or any other description)
;     6. permission (int)  0 for public, 1 for private
;     7. expId - experiment id (int) of the experiment this MadrigalExperimentFile belongs in
;   If no experiement files found, returns a Null Pointer (IDL does not allow empty arrays)
; EXAMPLE:
;     result = madGetExperimentFiles('http://madrigal.haystack.mit.edu/madrigal', 300041, 0)
;
```

```
; $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $
;
FUNCTION madGetExperimentFiles, madurl,  expId, getNonDefault


   ************************************************************
;+
; NAME:
;    madgetexperimentfileparameters(madurl, fullFilename)
;
; PURPOSE:
;    returns a list of all measured and derivable parameters in file
;
; INPUTS:
;      madurl - scalar string giving a fully qualified url to the Madrigal site
;      fullFilename - full path to experiment file as returned by madGetExperimentFiles.
;
; OUTPUT: an array of structures representing parameters for that experiment file with the foll
;     1. mnemonic (string) Example 'dti'
;     2. description (string) Example: "F10.7 Multiday average observed (Ott)"
;     3. isError (int) 1 if error parameter, 0 if not
;     4. units (string) Example "W/m2/Hz"
;     5. isMeasured (int) 1 if measured, 0 if derivable
;     6. category (string) Example: "Time Related Parameter"
;     7. isSure (int)  1 if parameter can be found for every record, 0 if can only be found for
;     8. isAddIncrement - 1 if additional increment, 0 if normal, -1 if unknown (this capabilit
;                         only added with Madrigal 2.5)
; EXAMPLE:
;     result = madGetExperimentFileParameters('http://madrigal.haystack.mit.edu/madrigal', '/op
;
; $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $
;
FUNCTION madGetExperimentFileParameters, madurl,  fullFilename


   ************************************************************
;+
; NAME:
;    madsimpleprint(madurl, fullFilename, user_fullname, user_email, user_affiliation)
;
; PURPOSE:
;    madSimplePrint prints the data in the given file is a simple ascii format.
;
;        madSimplePrint prints only the parameters in the file, without filters or derived
;        parameters.  To choose which parameters to print, to print derived parameters, or
;        to filter the data, use isprint instead.
;
; INPUTS:
;      madurl - scalar string giving a fully qualified url to the Madrigal site
;      fullFilename - full path to experiment file as returned by madGetExperimentFiles.
;      user_fullname - full name of user making request
;      user_email - email address of user making request
;      user_affiliation - affiliation of user making request
;
; OUTPUT: a structure with following fields:
;     1. parameters - an array of strings giving the mnemonics of the parameters.  Equal to the
;     2. data - a two dimensional array of double.  Number of columns = number of parameters ab
;         is the number of measurments.  Note that Madrigal often contains a number a measurmen
;         (such as when a radar makes different range measurements at the same time), so two or
; EXAMPLE:
```

```
;     result = madSimplePrint('http://madrigal.haystack.mit.edu/madrigal', '/opt/madrigal/blah/
;                                                'Bill Rideout', 'brideout@haystack.mit.edu', 'MI
;
; $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $
;
FUNCTION madSimplePrint, madurl,  fullFilename, user_fullname, user_email, user_affiliation


   ************************************************************
;+
; NAME:
;    madprint(madurl, fullFilename, parms, filters, user_fullname, user_email, user_affiliation
;
; PURPOSE:
;    madPrint returns a two-dimensional array of doubles based on user-specified parmeters and
;
;       See madSimplePrint to print a file with all data and just those parameters in the file
;
;        madPrint allows you to choose which parameters to print, to print derived parameters,
;        to filter the data.
;
; INPUTS:
;      madurl - scalar string giving a fully qualified url to the Madrigal site
;      fullFilename - full path to experiment file as returned by madGetExperimentFiles.
;      parms - Comma delimited string listing requested parameters (no spaces allowed).
;      filters - Space delimited string listing filters desired, as in isprint command (see
;                   http://madrigal.haystack.mit.edu/madrigal/ug_commandLine.html#isprint for d
;      user_fullname - full name of user making request
;      user_email - email address of user making request
;      user_affiliation - affiliation of user making request
;      ignore_no_data - if 0 (the default), raises error if no data, If 1, ignores no data
;
; OUTPUT:  a two dimensional array of doubles.  Number of columns = number of parameters reques
;         is the number of measurments.  Note that Madrigal often contains a number a measurmen
;         (such as when a radar makes different range measurements at the same time), so two or
; EXAMPLE:
;     result = madPrint('http://madrigal.haystack.mit.edu/madrigal', '/opt/madrigal/blah/mlh980
;                                    'year,month,day,hour,min,sec,gdalt,ti', 'filter=gdalt,500,6
;                                    'Bill Rideout', 'brideout@haystack.mit.edu', 'MIT')
;
; $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $
;
FUNCTION madPrint, madurl,  fullFilename, parms, filters, user_fullname, user_email, user_affil


   ************************************************************
;+
; NAME:
;    maddownloadfile, madurl, fullFilename, outputFile, user_fullname, user_email, user_affilia
;
; PURPOSE:
;    downloads a Madrigal file in simple ascii format to local computer
;
; INPUTS:
;      madurl - scalar string giving a fully qualified url to the Madrigal site
;      fullFilename - full path to experiment file as returned by madGetExperimentFiles.
;      outputFile - path to save file locally
;      user_fullname - full name of user making request
;      user_email - email address of user making request
;      user_affiliation - affiliation of user making request
```

```
;
; EXAMPLE:
;     maddownloadfile, 'http://madrigal.haystack.mit.edu/madrigal', '/opt/madrigal/blah/mlh9801
;                                     'Bill Rideout', 'brideout@haystack.mit.edu', 'MIT'
;
; $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $
;
PRO maddownloadfile, madurl,  fullFilename, outputFile, user_fullname, user_email, user_affilia


   ************************************************************
; globalIsprint is a procedure to search through the entire Madrigal database
; for appropriate data to print in ascii to a file
;
;    Inputs:
;
;        madurl - url to homepage of site to be searched (Example:
;              'http://www.haystack.mit.edu/madrigal/'
;
;        parms - a comma delimited string listing the desired Madrigal
;            parameters in mnemonic form.
;            (Example: 'year,month,day,hour,min,sec,gdalt,dte,te').
;            Ascii space-separated data will be returned in the same
;            order as given in this string. See
;            http://madrigal.haystack.mit.edu/cgi-bin/madrigal/getMetadata
;            "Parameter code table" for all possible parameters.
;
;        output - the local file name to store the resulting ascii data.
;                 (Example: '/tmp/isprint.txt')
;
;        user_fullname - the full user name (Example: 'Bill Rideout')
;
;        user email -  Example: 'brideout@haystack.mit.edu'
;
;        user_affiliation - Example: 'MIT'
;
;        startDate - a time in IDL Julian Date format at which to begin the search
;
;        endDate - a time in IDL Julian Date format at which to end the search
;
;        inst - instrument code (integer).  See
;            http://madrigal.haystack.mit.edu/cgi-bin/madrigal/getMetadata
;            "Instrument Table" for this list. Examples: 30 for Millstone
;            Hill Incoherent Scatter Radar, 80 for Sondrestrom Incoherent
;            Scatter Radar
;
;    Optional inputs
;
;        filters - is the optional filters requested in exactly the form given in isprint
;         command line (example = 'filter=gdalt,,500 filter=ti,500,1000')
;         See:  http://www.haystack.mit.edu/madrigal/ug_commandLine.html for details
;
;        kindats - is an optional array of kindat (kinds of data) codes to accept.
;           The default is a null pointer, which will accept all kindats.
;
;        expName - a case insensitive string as used by strmatch that matches the experiment
;           name.  Default is zero-length string, which matches all experiment names.
;
;        fileDesc - a case insensitive string as used by strmatch that matches the file descrip
;           Default is zero-length string, which matches all file descriptions.
;
```

```
;    Returns: Nothing.
;
;    Affects: Writes results to output file
;
;
;  Example:
;   madglobalprint, 'http://www.haystack.mit.edu/madrigal/',  $
;                           'year,month,day,hour,min,sec,gdalt,dte,te', $
;                           '/tmp/isprint.txt', $
;                           'Bill Rideout',  'brideout@haystack.mit.edu', 'MIT', $
;                           julday(1,19,1998,0,0,0),  julday(1,21,1998,23,59,59), 30
;
;  $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $
;
pro madglobalprint, madurl, parms, output, user_fullname, user_email, user_affiliation, startDa
                            inst, filters, kindats, expName,  fileDesc


   ************************************************************
;+
; NAME:
;    madcalculator(madurl, year,  month, day, hour, min, sec, startLat, endLat, stepLat,
;                               startLong, endLong, stepLong, startAlt, endAlt, stepAlt,  parms)
;
; PURPOSE:
;    madcalculator calculates requested derived parameter for a given time and grid of spatial
;
; INPUTS:
;      madurl - scalar string giving a fully qualified url to the Madrigal site
;      year,  month, day, hour, min, sec - time at which to run calculation
;      startLat - Starting geodetic latitude, -90 to 90 (float)
;      endLat - Ending geodetic latitude, -90 to 90 (float)
;      stepLat - Latitude step (0.1 to 90) (float)
;      startLong - Starting geodetic longitude, -180 to 180  (float)
;      endLong - Ending geodetic longitude, -180 to 180 (float)
;      stepLong - Longitude step (0.1 to 180) (float)
;      startAlt - Starting geodetic altitude, >= 0 (float)
;      endAlt - Ending geodetic altitude, > 0 (float)
;      stepAlt - Altitude step (>= 0.1) (float)
;      parms - comma delimited string of Madrigal parameters desired
;
; OUTPUT: a two dimensional array of doubles.  Number of columns = 3 + number of parameters req
;                first three parameters are always geodetic latitude, longitude, and altitude.
;                is the number of latitudes * number of longitudes * number of altitudes.
; EXAMPLE:
;     result = madcalculator(1999,2,15,12,30,0,45,55,5,-170,-150,10,200,200,0,'bmag,bn')
;
; $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $
;
FUNCTION madcalculator, madurl, year,  month, day, hour, min, sec, startLat, endLat, stepLat,
                       startLong, endLong, stepLong, startAlt, endAlt, stepAlt,  parms
```

	Madrigal administrator's guide	Doc home	Madrigal home

Previous: Python remote API reference **Up:** Doc home **Next:** Is Madrigal appropriate?

Madrigal administrator's guide

This section of the Madrigal documentation is meant for people considering installing Madrigal to hold data from their instruments, or those who have already installed Madrigal and are responsible for administering or updating it. This guide describes how to determine whether Madrigal is right for your data, and how to install it if it is. It also discusses how to create Madrigal data files, and how to add them to Madrigal.


Is Madrigal appropriate for my instrument(s)? | Doc home | Madrigal home

Previous: Madrigal admin guide **Up:** Madrigal admin guide **Next:** Installing Madrigal

Is Madrigal appropriate for my instrument(s)?

The Madrigal database is designed to hold data about the upper atmosphere. One of the strengths of the Madrigal is that most of the measured parameters it contains are defined in a community standard, the Cedar database format. This standard defines many frames of reference used commonly in atmospheric science, such as geodetic coordinates, geomagnetic coordinates, or radar (azimuth, elevation, and range) coordinates. Upper atmospheric data that can fit into one of these coordinate systems is a good candidate for Madrigal. Madrigal is not presently designed to handle spacecraft ephemeris in order to determine location, and to date Madrigal has only been used to hold spacecraft data that is independent of position (such as solar wind) or can be converted to one of the coordinate systems discussed above (such as total electron concentration in geodetic coordinates from GPS satellites).

Generally, if the parameters found in the Cedar database format are appropriate for you, or only a few additions need to be made, then data from your instrument(s) is probably appropriate for Madrigal. Please feel free to contact the OpenMadrigal administrator if you have any questions about using Madrigal.


Is Madrigal appropriate for my instrument(s)? | Doc home | Madrigal home

Previous: Madrigal admin guide **Up:** Madrigal admin guide **Next:** Installing Madrigal


Installing Madrigal for the first time | Doc home | Madrigal home

Previous: Is Madrigal appropriate? **Up:** Madrigal admin guide **Next:** Upgrading Madrigal

Installing Madrigal for the first time

Prerequisites

Most of the prerequisites for Madrigal are pre-installed in any modern unix distribution. They are:

1. A C and a FORTRAN compiler. The free GNU compilers may be downloaded from the GNU Website .
2. tclsh, which may be downloaded from the Tcl Resource Center .
3. A web server.
4. The environmental variable MADROOT should be set to the path to the Madrigal installation directory. Its recommended this be done in your login script (eg, .bash_profile) because you will need it to run administrative scripts later.
5. The environmental variable LD_LIBRARY_PATH should be set to $LD_LIBRARY_PATH:$MADROOT/lib. Its recommended this be done in your login script (eg, .bash_profile) because you will need it to run administrative scripts later.

The installation script will stop and issue a warning if any prerequisite is missing.

Installation instructions

Madrigal can be installed on any unix server with a web server. If you want to link your data in with data on other Madrigal servers, please notify the OpenMadrigal administrator. In general you do not need root permission to install madrigal once the prerequisites listed above are installed. You may need to be root to create the html and cgi directories in the web server. It is generally easiest to then chown for those two directories to the user that will be installing Madrigal.

1. Create a directory to be used as your Madrigal root directory on your unix server. This directory will be referred to as MADROOT.
2. Create the environment variable MADROOT with that directory path.
3. Download the latest Madrigal distribution file, madrigal*.tar.Z from the OpenMadrigal distribution page to MADROOT.
4. uncompress madrigal*.tar.Z
5. tar -xf madrigal*.tar
6. Repeat the 3 steps above to get and untar the Sample Experiments file *experiments.tar.Z*.
7. Create two virtual directories on your webserver for your madrigal cgi files (which must allow scripts to run) and for your madrigal html files.
8. In madroot, there will now be a file called madrigal.cfg.template. Copy it to madrigal.cfg, and edit all the configuration parameters, as described in the Editing madrigal.cfg section below.
9. Edit the file MADROOT/metadata/siteTab.txt with a unique id to include your site. The format of siteTab.txt is described here. If you want to be an official Madrigal site that other Madrigal sites share data with, request the OpenMadrigal administrator to assign you this id. If you do not want all the sites in siteTab.txt included in your combined inventory listing, just remove these sites from siteTab.txt. Links to data from sites you leave in the siteTab.txt file will appear in your Madrigal web site.
10. With Madrigal 2.6, users can now download cached HDF5 files. These cached HDF5 files are created during installation. You will want to make sure you have enough room in your experiments directory (typically about double the existing space used). You can create an ini file to modify how these HDF5

files are created. Instructions below describe how to added derived parameters and/or additional layouts for an given instrument, and optionally by kindat.

11. Be sure to cd to MADROOT before running the following step. Then you should be able to complete the installation simply by typing

```
bash installMadrigal  &> install.log &
```

    There may be a long pause when running updateMaster near the end of the installation since the instParmTab.txt metadata file is being built for the first time by examining every data file, but future calls to updateMaster will be much faster since only new experiments are examined. Help with any installation errors is available from the OpenMadrigal administrator.
12. If there were no errors, your madrigal installation should be running at the url given by MADSERVERROOT. You can also test it by running MADROOT/bin/python testMadrigalBuild.py, which will test the ability to create plots.
13. If you want to receive notices about updates to Madrigal, sign up for the openmadrigal-admin mailing list under www.openmadrigal.org.
14. Set the script *madroot*/bin/updateMaster up as a cron job to run once a day.
15. If you want to add any documentation pages specific to your site to the Madrigal documentation pages, see the other admin tasks section of this manual.
16. If you want to add your own rules of the road to the Madrigal experiment page, see the other admin tasks section of this manual.

Editing the madrigal.cfg file

The madrigal.cfg file contains all the configuration information specific to your installation. This section discusses how to edit that file for each parameter. The madrigal.cfg.template file contains examples of each parameter.

- MADROOT - Madrigal root directory (absolute). This must be set as an environmental variable.
- MADSERVER - Web server for accessing Madrigal
- MADSERVERROOT - document virtual directory relative to MADSERVER. If the full Madrigal Url is the same as the MADSERVER field (for example, if the url is *http://madrigal.hao.ucar.edu/*), then set this field to a period. (Example: *MADSERVERROOT* = .) This directory should not allow files to be executed.
- MADSERVERCGI - script virtual directory relative to MADSERVER. The web server will need write permission in this directory. This directory should be set to execute only.
- MADSERVERDOCABS - Directory (absolute) where Madrigal documentation and images necessary for the Web software should be installed. You will need write permission to install files in this directory. You may need to be root to create this directory.
- MADSERVERCGIABS - Directory (absolute) where Madrigal CGI scripts will be installed. You will need write permission to install files in this directory. You may need to be root to create this directory.
- SITEID - Site ID - Must be unique and same as in siteTab.txt
- HTMLSTYLE - Body style of html pages
- INDEXHEAD - Heading in the top-level Madrigal page
- CONTACT - Mailto link of contact person(s) for this madrigal installation. Multiple email addresses may be included if separated by commas.
- MAILSERVER - Name of mailserver (possibly localhost if running sendmail)
- NOTESMANAGER - If you want someone to be notified whenever a user adds a note to an experiment, uncomment this optional line and add the email address. See below for a discussion of the optional notes feature and how to configure it after installation.

- MAXGLOBALQUERIES - The maximum number of global queries you want to allow to run on your webserver at any one time. Since a global query can take minutes or even hours to run, setting this value will limit the number of these background jobs running on your webserver. Users who request a global query when the server is at this maximum already will get a message requesting them to resubmit the query later.
- MAXTEMPREPORTS - Sets the maximum size of the tempReports directory in GB. If not set, defaults to 2 GB.

Creating or modifying MADROOT/cachedFiles.ini

By default, Madrigal will create cached HDF5 versions of all default files when Madrigal 2.6 or later is installed. These cached HDF5 files are created so that the user will get a fast response when an HDF5 file is requested for download. Large HDF5 files can take minutes to create from the Cedar source file. By default, these cached HDF5 files only contain the parameters in the Cedar file, and arrange the data only in the default table layout. If you want these cached HDF5 files to have any additional derived parameters or extra layout formats, you will need to create a file called cachedFiles.ini in your MADROOT directory.

With Madrigal 2.6, there is only one additional layout available: 'array'. If you ask for this additional layout, your HDF5 file will contain a subdirectory where each parameter has its own two-dimensional table, where one dimension is the dependent variable (for example, range) and the other dimension is record. This layout is more complex than the standard table layout, but may allow easier plotting of the data.

This ini file has section headings which are simply instrument ids (kinst code). The key value pairs can be:

1. <kindat>_parms = <comma-delimited parameter list>
2. <kindat>_formats = <python dictionary>
3. default_parms = <comma-delimited parameter list>
4. default_formats = <python dictionary>

If an instrument does not have a section, then defaults are used. If a given instrument has a default_parms or default_formats key, then that will be used, unless a key with the matching kindat (kind or data) is found. If an instrument has kindat keys and no default, and the kindat does not match, the default is used.

Here's an example cachedFiles.ini file. In this case, for kinst=30 (the Millstone ISR), kindats 13204 and 13300 are listed separately. If any other kindat is found, the default key is used:

```
# Millstone Hill radars
[30]
# 13204 is F regions winds file

13204_parms =
13204_formats = {}
# 13300 is gridded ISR data
13300_parms =
13300_formats = {}
# all other kindats should be as follows
default_parms = gdlat,glon,gdalt,ne,dne
default_formats = {'array':'range'}
```

Previous: Is Madrigal appropriate? **Up:** Madrigal admin guide **Next:** Upgrading Madrigal

	Upgrading or moving Madrigal	Doc home	Madrigal home

Previous: Installing Madrigal **Up:** Madrigal admin guide **Next:** Madrigal data model

Upgrading or moving Madrigal

With the release of Madrigal 2.5, the procedure to upgrade and/or move Madrigal to a new server was improved. To either upgrade Madrigal on your existing server, or to move it to another server, you can now do a clean installation first. This clean installation will not effect your present Madrigal server. If the clean installation succeeds, then you migrate your Madrigal data and make the new server active. If the migration fails for any reason, it is simple to switch back to the original Madrigal server. Of course you can also directly upgrade your present Madrigal installation also - you simply need to be sure the server is fully backed up in case the installation fails.

To determine what release of Madrigal you are presently running, see the title of the main documentation page (that's the page you get to by clicking on *Documentation* from the home page). The latest release of Madrigal will always be available from the OpenMadrigal web site via the Download/Update link.

This page covers the following administrative tasks:

- Upgrading Madrigal (using the same server)
- Moving Madrigal to a new server (installs the latest Madrigal version)
- Moving the Madrigal madroot directory
- Direct Madrigal upgrade (the old method - be sure to back up your server before attempting)

Setting environmental variables

It is now required that the Madrigal administrator set the following environmental variables in their login script:

- MADROOT - the path to the Madrigal installation directory
- LD_LIBRARY_PATH - set to $LD_LIBRARY_PATH:$MADROOT/lib

The installation script will also tell you if you need to set LD_LIBRARY_PATH for the web server, and give you instructions if its needed.

Upgrading Madrigal (using the same server)

To upgrade Madrigal to the latest release using the same server:

1. Create a new MADROOT directory for your new installation on your existing Madrigal server. After installation this will be your permanent MADROOT directory (unless you then follow the moving MADROOT procedure). Set your MADROOT environmental variable to this new directory.
2. Create two temporary directories on your existing Madrigal server for 1) html files and 2) cgi scripts. For example, if you standard Madrigal server stores its html files under /var/www/html/madrigal and its cgi files under /var/www/cgi-bin/madrigal, you could create the directories /var/www/html/madrigal2 and /var/www/cgi-bin/madrigal2.
3. From OpenMadrigal, download the Madrigal distribution file, madrigal*.tar.Z to your MADROOT directory.

4. From OpenMadrigal, download the sample experiment file, experiments.tar.Z to your MADROOT directory.
5. uncompress experiments.tar.Z and madrigal.tar.Z
6. tar -xf experiments.tar.
7. tar -xf madrigal*.tar.
8. Copy madrigal.cfg from the old MADROOT to the new MADROOT, and edit the following **five** fields:
    1. MADROOT - set to new MADROOT
    2. MADSERVERROOT - New, temporary url directory relative to MADSERVER for the temporary html files. This directory should not allow files to be executed. For the "/var/www/html/madrigal2" example, this would be "madrigal2".
    3. MADSERVERCGI - New, temporary script directory relative to MADSERVER for temporary cgi files. The web server will need write permission in this directory. This directory should be set to execute only. For the "/var/www/cgi-bin/madrigal2" example , this would be "madrigal2".
    4. MADSERVERDOCABS - New, temporary directory (absolute) where html files should be installed. This directory was created in step 2.
    5. MADSERVERCGIABS - New, temporary cgi directory. This directory was created in step 2.
9. Copy siteTab.txt from the old MADROOT/metadata directory to the new MADROOT/metadata directory. This file is comma-delimited. For your site, edit field 4 to be the new MADSERVERROOT set in step 8-2, and field 5 to be the new MADSERVERCGI set in step 8-3.
10. With Madrigal 2.6, users can now download cached HDF5 files. These cached HDF5 files are created during installation. You will want to make sure you have enough room in your experiments directory (typically about double the existing space used). You can create an ini file to modify how these HDF5 files are created. Instructions here describe how to added derived parameters and/or additional layouts for an given instrument, and optionally by kindat.
11. Be sure to cd to new MADROOT before running the following step. Run the following command:

    ```
    bash installMadrigal  &> install.log &
    ```

    Help with any installation errors is available from the OpenMadrigal administrator.
12. If there were no errors, test your madrigal installation at the url given by MADSERVERROOT. You can also test it by running MADROOT/bin/python testMadrigalBuild.py, which will test the ability to create plots.
13. In the next step you will run a script that will switch the active Madrigal installation to the new installation. By default, it will copy all the experiment data to the new MADROOT directory, so you'll need to make sure there's enough room on the hard drive for two copies of the experiments directory. If you do not have room for that, you may use the --moveExp flag to cause the experiment directory to be moved to the new MADROOT directory, rather than copied. This script will also modify madrigal.cfg to use the web paths of the original Madrigal server, and will then install the new cgi scripts and html documents to those original web directories. In this way users will not need to enter a new url to visit your Madrigal site. If any problem emerges with the new version, instructions are given in the next step for switching back to the original Madrigal version. To make the switch, cd to the new MADROOT and run:

    ```
    bin/switchMadrigal [--moveExp] <original_madroot> <new_madroot>
    ```

    You should now be able to test the new release of Madrigal at the main Madrigal url.
14. If for any reason you decide you want to switch back to the old version of Madrigal, it is easy to do so. Just export MADROOT to be the old value, cd to the old MADROOT, and run:

```
tclsh configureHtml
tclsh configureScripts
bin/updateMaster
```

You should now see the old version of Madrigal at the main Madrigal url. If you used the --moveExp flag when switching in the previous step, you will need to preceed this procedure by moving the experiments back to the original MADROOT directory and running *./configureExperiments*.

15. There may be reasons you do not want the value of the MADROOT environmental variable to change when you update Madrigal on the same server. For example, you may have written programs using the Madrigal API that hard-coded the MADROOT value. If, after you are done with the procedure above, you decide you want the new version on Madrigal to run using the old MADROOT value, follow the procedure Moving the Madrigal madroot directory.
16. This update procedure creates two copies of Madrigal and all the data. Once you are satisfied with the new version of Madrigal, free up disk space by removing:
    1. The old MADROOT directory
    2. The temporary MADSERVERROOT directory created in step 9-2.
    3. The temporary MADSERVERCGI directory created in step 9-3.

Moving Madrigal to a new server (installs the latest Madrigal version)

If you want to move Madrigal to a new server, and the main Madrigal url is changing, you should notify the OpenMadrigal administrator so that they can update the Madrigal metadata that lists Madrigal sites. If you are moving to a new server but retaining the old Madrigal url, there is no need to change the metadata file siteTab.txt.

In this procedure you will first install a fresh version of Madrigal, and then run a script to import your Madrigal data from your old server.

1. Install a new version of Madrigal as decribed in Installing Madrigal for the first time. You may use your old madrigal.cfg file as a guide, but you will need to change it for the fields that have changed for the new server.
2. Once you are happy with the new installation, run the following script from the MADROOT directory of the new Madrigal server:

```
bin/switchMadrigal.py --newUrl <username>@<server>:<original_madroot> <new_madroot>
```

3. Check that all the data from the old Madrigal server now appears on the new server.
4. Once you are satisfied with the new installation, you may remove the old installation by removing the following three directories on the old server:
    1. The MADSERVERROOT directory on the old server (specified in old_madroot/madrigal.cfg)
    2. The MADSERVERCGI directory on the old server (specified in old_madroot/madrigal.cfg)
    3. The old MADROOT directory

Moving the Madrigal madroot directory

There may be reasons you do not want the value of the MADROOT environmental variable to change when you update Madrigal on the same server. For example, you may have written programs using the Madrigal API that hard-coded the MADROOT value. If, after installing Madrigal as described in the Upgrading Madrigal section, you want to have MADROOT return to its original value, do the following:

1. Rename the old MADROOT directory to a different name
2. Rename the new, active MADROOT directory to the original MADROOT name.
3. Set the MADROOT environmental variable to be the original MADROOT name.
4. Make sure LD_LIBRARY_PATH contains <original MADROOT>:lib
5. Edit the MADROOT line in madrigal.cfg to be the original MADROOT name.
6. Next you will rerun the installation. It should be faster since the creation of metadata and cached HDF5 files will be skipped:

   ```
   bash installMadrigal &> install2.log &
   ```
7. Help with any installation errors is available from the OpenMadrigal administrator.

Direct Upgrade

To install the latest Madrigal release using the Direct Upgrade method:

1. Be sure you have recently backed up your entire server in case a problem arises.
2. Download the Madrigal distribution file, madrigal*.tar.Z to your MADROOT directory.
3. uncompress madrigal*.tar.Z
4. tar -xf madrigal*.tar (This will not overwrite any file meant to be modified by a specific Madrigal installation.)
5. Compare the new version of madrigal.cfg.template file to your existing madrigal.cfg file found under the MADROOT directory. If any new parameters have been added to the end of the file, add them to you madrigal.cfg file and edit just those entries as described in the installation documentation.
6. With Madrigal 2.6, users can now download cached HDF5 files. These cached HDF5 files are created during installation. You will want to make sure you have enough room in your experiments directory (typically about double the existing space used). You can create an ini file to modify how these HDF5 files are created. Instructions here describe how to added derived parameters and/or additional layouts for an given instrument, and optionally by kindat.
7. Be sure to cd to MADROOT before running the following step. Then you should be able to complete the installation simply by typing

   ```
   bash installMadrigal &> install.log &
   ```

   There may be a long pause when running updateMaster near the end of the installation since the instParmTab.txt metadata file is being built for the first time by examining every data file, but future calls to updateMaster will be much faster since only new experiments are examined. There may also be a delay when all the cached HDF5 files are created. Help with any installation errors is available from the OpenMadrigal administrator.
8. If there were no errors, your madrigal installation should be running at the url given by MADSERVERROOT. You can also test it by running MADROOT/bin/python testMadrigalBuild.py, which will test the ability to create plots.
9. If you want to add any documentation pages specific to your site to the Madrigal documentation pages, see the other admin tasks section of this manual.
10. If you want to add your own rules of the road to the Madrigal experiment page, see the other admin tasks section of this manual.

The Madrigal data model and metadata files

Understanding the Madrigal data model is an important step in understanding how Madrigal works. There is a correspondence between each level of the data model and the metadata files that are found in the MADROOT/metadata directory. In this section we describe each level of the Madrigal data model and the corresponding metadata file.

Madrigal site - siteTab.txt

The highest level of Madrigal is a Madrigal site. A Madrigal site is one particular web site controlled by one particular group, that holds all their own data. At the moment, there are Madrigal sites at Millstone Hill, USA, EISCAT, Sweden, Arecibo, Puerto Rico, SRI International, USA, Cornell University, USA, Jicamarca, Peru, The Institute of Solar-Terrestrial Physics, Russia, and Wuhan Ionospheric Observatory, the Chinese Academy of Sciences. While each Madrigal site stores their own data locally, they also share metadata with all the other sites. This makes it possible for users to search for data at all the Madrigal sites at once no matter which site they visit, and simply follow links to the Madrigal site that has the data they are interested in.

Metadata about all sites is stored in MADROOT/metadata/siteTab.txt. When new Madrigal sites are added, this table is updated and all Madrigal sites are notified so they can update this file. If a site is running Madrigal 2.5 or higher, this file will be automatically updated unless the file has been manually modified by the administrator in a way not reported to OpenMadrigal administrator. This file contains the following comma-separated fields:

- Site ID (e.g., 1)
- Site Name (e.g., Millstone Hill Observatory)
- Madrigal server (e.g., www.haystack.mit.edu)
- Madrigal document root relative to server (e.g., madrigal)
- Madrigal CGI directory relative to server (e.g., cgi-bin/madrigal)
- Madrigal servlet directory relative to server (e.g., madrigal/servlets) - This field is no longer used.
- Contact name (e.g., John M. Holt)
- Contact Address 1 (e.g., MIT Haystack Observatory)
- Contact Address 2 (e.g., Route 40)
- Contact Address 3 (e.g., "")
- Contact City (e.g., Westford)
- Contact State/Province (e.g., MA)
- Contact Postal Code (e.g., 01886)
- Contact Country (e.g., USA)

- Contact Telephone (e.g., 1-617-981-5624)
- Contact email (e.g., mailto:jmh@haystack.mit.edu) Multiple addresses may be listed if separated by semicolons.

Instrument - instTab.txt

The next layer of the Madrigal data model is the instrument. All data in Madrigal is associated with one and only one instrument. Any given Madrigal site will hold data from one or more instruments. Since Madrigal focuses on ground-based instruments, most instruments have a particular location associated with them. However, some Madrigal data is based on measurements from multiple instruments, and so have no particular location. Some examples are "EISCAT Scientific Association IS Radars" which combine data from the multiple EISCAT radars, and "World-wide GPS Receiver Network", which consists of over a thousand individual GPS receivers distributed around the globe.

Metadata about all instruments is stored in MADROOT/metadata/instTab.txt. When new Madrigal instruments are added, this table is updated and all Madrigal sites are notified so they can update this file. If a site is running Madrigal 2.5 or higher, this file will be automatically updated unless the file has been manually modified by the administrator in a way not reported to OpenMadrigal administrator. This instrument code list is usually consistent with the Cedar instrument list. This file contains the following comma-separated fields:

- Instrument Code (e.g., 30)
- Instrument 3-letter Mnemonic (e.g., mlh)
- Instrument Name (e.g., Millstone Hill Incoherent Scatter Radar)
- Latitude (e.g., 42.5)
- Longitude (e.g., -71.9)
- Altitude in km above sea level (e.g., 0.146)
- Contact name (e.g., John M. Holt)
- Contact Address 1 (e.g., MIT Haystack Observatory)
- Contact Address 2 (e.g., Route 40)
- Contact Address 3 (e.g., "")
- Contact City (e.g., Westford)
- Contact State/Province (e.g., MA)
- Contact Postal Code (e.g., 01886)
- Contact Country (e.g., USA)
- Contact Telephone (e.g., 1-617-981-5625)
- Contact email (e.g., mailto:jmh@haystack.mit.edu)
- Instrument category id (e.g., 6)

Instrument type - instType.txt

The instrument type table lists categories of instruments, to allow the user to search instruments more easily. If a site is running Madrigal 2.5 or higher, this file will be automatically updated unless the file has been manually modified by the administrator in a way not reported to OpenMadrigal administrator. This file contains the following comma-separated fields:

- Instrument category id (e.g., 6)
- Instrument category description (e.g., Incoherent Scatter Radar)

Experiment - expTab.txt

All the data from a given instrument is organized into experiments. An experiment consists of data from a single instrument covering a limited period of time, and, as a rule, is meant to address a particular scientific goal. Madrigal makes the assumption that instruments may be run in different modes, and so the data generated may vary from one experiment to another. By organizing one instrument's data into experiments, the purpose and limitations of each experiment can be made clearer. As a Madrigal administrator, you can also provide users with supplemental plots and documentation about that experiment, in addition to the standard Madrigal data files. See the section on creating and updating Madrigal experiments for more information.

Madrigal has a number of security codes for different types of experiments. Most experiments are public, which allows them to be accessed by everyone. An experiment can also be made private, which means that only users with set ranges of ip addresses can access them. (See Set private versus public access.) Private experiments are never shared with other Madrigal sites. There is also a hidden experiment state to completely remove an experiment from access by Madrigal (if, for example, the data is discovered to be corrupt, but might be fixed in the future).

There are also two experiment states for archiving experiments. These are *public archive* and *private archive*. These states are meant to support the archiving of Madrigal data at a central Madrigal site, such as the one at NCAR. These archived experiments are duplicates of experiments found at other Madrigal sites. In general these archived experiments are ignored by any part of the user interface that searches all sites, because the user will only want to find the main data source. However, when the user interface is only accessing local data, these archived experiments will appear. A private archived experiment is subject to the same restriction as a regular private experiment.

Metadata about all experiments is stored in MADROOT/metadata/expTab.txt. This file is automatically generated from individual expTab.txt files located in each experiment directory, as will be described in the next section on experiment organization. This file contains the following comma-separated fields:

- Experiment ID (auto-generated)
- Experiment URL (e.g., http://www.haystack.mit.edu/cgi-bin/madtoc/1997/mlh/03dec97g). Note this url is historical, and no longer works.
- Experiment Name (e.g., Wide Latitude Substorm Study)
- Site ID (e.g., 1)
- Start Date (YYYYMMDD) (e.g., 19971203)
- Start Time (HHMMSS)(e.g., 011356)
- End Date (e.g., 19971205) (YYYYMMDD)
- End Time (e.g., 123525) (HHMMSS)
- Instrument Code (e.g., 30)
- Security Code (e.g., 0) - 0 for public, 1 for private, -1 for completely ignored, 2 for public archive, 3 for private archive.
- Principle Investigator (e.g., Phil Erickson) optional field - used to override the instrument PI
- Principle Investigator email (e.g., perickson@haystack.mit.edu) optional - PI must also be given

There is also a file called MADROOT/metadata/expTabAll.txt which is also automatically generated. If differs from expTab.txt in that it contains experiment metadata from all Madrigal sites, not just the local one. Any remote experiment with a non-zero security code will be excluded.

Experiment Files - fileTab.txt

The data from a given experiment is stored in one or more experiment files. There are a number of reasons there may be more than one file for a given experiment. The first is that different kinds of data may be stored in different files. Also, the experimental data may be analyzed in more than one way, leading to files with different sets of measured parameters. For these two cases, each file should have its own kindat code (see below). Another reason for multiple files is that older, historical files can be kept on-line for reference purposes.

The format of these files can be any of the allowed variants of the Cedar database format. Each file may contain only one kindat. The category field is used to distinguish files which are of historical interest only, e.g. a file which have been superseded by a file with an improved electron density calibration. In some cases there may be more than one up-to-date variant of a file, e.g. when different analysis options have been chosen. In this case one of these files is designated the default, and the others are designated as variants.

Metadata about all experiment files is stored in MADROOT/metadata/fileTab.txt. This file is automatically generated from individual fileTab.txt files located in each experiment directory, as will be described in the next section on experiment organization. This file contains the following comma-separated fields:

- File Name (e.g., mil971203g.002)
- Experiment ID (e.g., 10000125)
- Data Type (e.g., 3001)
- Category (1=default, 2=variant, 3=history, 4=real-time)
- Size of File (e.g., 241920)
- File contains at least one Catalog Record File (0 for no, 1 for yes)
- File contains at least one Header Record File (0 for no, 1 for yes)
- Analysis/modify Date (YYYYMMDD) (e.g., 19980101)
- Analysis/modify Time (HHMMSS) (e.g., 115131)
- File processing status description (preliminary, final, or any other description)
- Permission flag: 0 for public, 1 for private
- File analyst - who did the analysis of the data and created this file (e.g. John Holt) optional
- File analyst email (e.g., jholt@haystack.mit.edu) optional - File analyst must also be given

There is also a file called MADROOT/metadata/fileTabAll.txt which is also automatically generated. If differs from fileTab.txt in that it contains experiment file metadata from all Madrigal sites, not just the local one.

Data parameters - parcods.tab

Any given file is made up a series of records holding measured parameters. Note that based on which parameters are in the file, Madrigal will automatically derive a large number of other parameters such as Kp and Magnetic field strength that aren't in the file itself. In the web browser, measured parameters are shown in bold, derived parameters in normal font.

The metadata file parcods.tab contains information about what Madrigal or Cedar parameters are supported. Madrigal parameters are a superset of Cedar parameters. If a Madrigal parameter has a parameter code of 0, it cannot be stored in a Cedar file and is meant the be a derived value only. All Madrigal mnemonics must be unique. All non-zero parameter codes must be unique, and are set by the Cedar standard. If a new parameter is desired, it should be done in coordination with Barbara Emery at NCAR (emery@ucar.edu).

The file parcods.tab is not comma delimited, but instead is fixed length formatted.

- Parameter Code (may be zero) (columns 0-7)
- Description (columns 10-48)
- Int16Desc (columns 50-60)
- ScaleFactor (columns 62-68)
- Units (columns 70-77)
- Mnemonic (columns 81-100)
- Format (Now use C-style formatting) (columns 105-112)
- Width (columns 114-115)
- Category Id (see madCatTab.txt) (columns 118-120)
- Mnemonic has Html description (1 or 0) (columns 122-122) - used to display extra information about the parameter
- Err mnemonic has Html description (1 or 0) (columns 124-124) - used to display extra information about the error parameter

Since data in the Cedar format is presently stored as 16 bit integers, parameters only have a limited dynamic range, and care must be taken in selecting units and scale factor. To increase dynamic range, sometimes an additional, finer scale parameter is also added to the Cedar parameter list, called an additional increment parameter. See, for example, parameters 120 and 121 in the Cedar format, whose descriptions are "Range" and "Additional increment to range". Because 16 bit integers have a dynamic range of 2^16, the scale factor of the additional increment parameter is typically a factor of 10^4 lower than the main parameter.

Madrigal is designed to automatically use additional increment parameters found in parcods.tab. To add a new parameter with an accompanying additional increment parameter, the new additional parameter must follow the following rules:

1. It must have the code (1 + code of main parameter)
2. Its description must begin "Additional increment" (case sensitive)
3. It must use the same units as the main parameter.

Parameter explanations

For parameters that cannot be fully described in the 38 characters allowed in the parcods.tab file, additional explanation about the parameter or its corresponding error parameter can be added to the file madroot/doc/parmDesc.html. Simply create a new named anchor in that file, where the anchor name is the parameter mnemonic in all capitals. Following that, a description of arbitrary length can be given using html. Change one of the last two columns from 0 to 1 for that parameter in parcods.tab to let Madrigal know that this explanation exists. In general, the parameter order in parmDesc.html matches that of parcods.tab, but that is not a functional requirement.

Parameter categories - madCatTab.txt

The Madrigal category metadata file(madCatTab.txt) contains information about what categories Madrigal parameters belong in. The categories are similar to the Cedar categories, but do not follow them exactly. This file does not change. This file contains the following comma-separated fields:

- Category id (integers, starting at 0)
- Category name

- Minimum parameter code (integer). Used only for codes not listed in parcods.tab. If category doesn't have a range of codes, use -1.
- Maximum parameter code (integer). Used only for codes not listed in parcods.tab. If category doesn't have a range of codes, use -1.

Data type (kindat) table - typeTab.txt

The Madrigal data type (also called kind of data or kindat) metadata file(typeTab.txt) contains a list of all data types in the database. The purpose of kindat is to uniquely identify the data processing algorithm used to compute the parameters in the associated Madrigal file. For now this metadata is only used locally; however, Madrigal sites that develop new data processing algorithms should update this table, and then forward the revised typeTab.txt metadata file to the OpenMadrigal administrator. While not required by Madrigal, this updating would allow this table to be consistent with the CEDAR Database list of kindat codes.

- Data Type Code (e.g., 3001)
- Data Type Brief Description (e.g., Basic Derived Parameters)

Instrument parameter table - instParmTab.txt

The instrument parameter metadata file (instParmTab.txt) contains information about what measured parameters are found in the data for any given instrument. This data is used to support the global database query web page, and is rebuild by updateMaster. This file contains the following comma-separated fields:

- Instrument Code (e.g., 30)
- Parameter mnemonic list (e.g., range rangei az1 az2 el1 el2 pl snp3 chisq mhdqc1 systmp systmi power tfreq popl dpopl ti dti tr dtr vo dvo ph+ dph+ pm dpm fa dfa pnrmd pnrmdi vdopp dvdopp)

Instrument kindat table - instKindatTab.txt

The instrument kindat metadata file (instKindatTab.txt) contains information about what kindat codes are used with any given instrument. This data is used to support the global database query web page, and is rebuild by updateMaster. This file contains the following comma-separated fields:

- Instrument Code (e.g., 30)
- Kindat code list (e.g., 3408 13204 13210 3001)

File data

The bottom level of the Madrigal data model is of course the data itself. A Madrigal file is made up of a series of records, each with a start and stop time, representing the integration period of measurement (Madrigal tries to enforce the idea that all measurements take a finite time, but sometimes the start time = the stop time). To get data from a file, simply specify the parameters you want (and optionally, any filters to apply to the data). More details are given later in this tutorial.

Each Madrigal record has two parts - scalar parameters and vector parameters. For historical reasons these two parts are sometimes called one-dimensional and two-dimensional parameters. Scalar parameters are easy to explain - each scalar parameters has one measurement per record. An example might be the azimuth of a radar making a measurement. Vector parameters have multiple values in a given record. The Cedar file format specifies that all vector parameters must have the same number of measurements. One of more of the vector parameters represent the independent spatial variable(s). For radars this variable is typically range, but latitude, longitude, and altitude could just as easily be used as the three independent spatial variables. The

dependent vector variables must all have the same length as the independent variable(s). The independent parameter should never represent time, since the Cedar format specifies that that one record should should cover one period of time.

For example, a radar might store azimuth and elevation as scalar parameters, and range as the independent vector variable. If the electron density and ion temperature are dependent vector variables, and there are ten range measurements, then there must be ten measurements of electron density, and ten measurements of ion temperature. If at certain ranges it is impossible to determine the ion temperature, the Cedar format defines a special value to represent missing data to fill the gap.

The Cedar file format defines the physical meaning of almost every parameter to be found in a Cedar file. The only exceptions are parameters defined by individual groups. Any parameter found in a Cedar file that is not defined in the Cedar file format should be fully defined in the header record of the file. See the experiment page for a description of how to view a Cedar file's header record.

Each Cedar parameter can also have an associated error value. This error value can have the special values "missing", "assumed", or "known bad". If an error parameter is "assumed", the implication is that the measured value itself is assumed, and does not represent a measured value. If the error value is "known bad", the measured data is known to have a problem.


The Madrigal data model and metadata files | Doc home | Madrigal home

Previous: Upgrading Madrigal **Up:** Madrigal admin guide **Next:** Madrigal data organization


How Madrigal data is organized | Doc home | Madrigal home

Previous: Madrigal data model **Up:** Madrigal admin guide **Next:** Creating Madrigal data files

How Madrigal data is organized

In the Madrigal database, the data are organized by experiment. An experiment consists of data from a single instrument, and, as a rule, is meant to address a particular scientific goal. Most often an experiment will correspond to a particular set of operating modes run for a contiguous interval of time. For example, the data collected by an incoherent scatter radar during a world day would constitute an experiment. More complicated situations exist, and judgment may have to be exercised in determining what constitutes an experiment. For example, an experiment might be interrupted temporarily in order to use a different set of operating modes in support of a satellite overpass. In this case there could be two experiments which overlap in time.

To each experiment there corresponds a directory. These directories are of the form *madroot*/experiments[0-9]*/<year>/<instrument>/<*directory*>, where year is the four-digit year. The next level is the experiments[0-9]*, where the directory must begin "experiments" and then may be followed by any number of digits. This is to allow additional hard drives to be mounted to expand the directory space. The next level, instrument, is the 3-letter mnemonic for the instrument (set in the instTab.txt file), and <*directory*> is an arbitrary directory name. An example might be /opt/madrigal/experiments/1997/son/06jan1997_001, which would contain a 1997 Sondrestrom (son) experiment. The use of the starting date in the experiment directory name is not required.

Prior to Madrigal 2.5, the final directory name was required to be in the form DDmmmYY*, where the date was the start date of the experiment and * is an optional character to distinguish different experiments with the same start date. For example, /opt/madrigal/experiments/1997/son/06jan97 contained Sondrestrom data for an experiment beginning on 6 January, 1997, where year =1997, instrument =son, start_date =06jan97, and there was no optional character because there was only one Sondrestrom experiment starting on that date. This convention has been dropped as a requirement, although it is still commonly used as a convention.

Previously, experiments that did not follow the above naming convention were effectively "hidden". With Madrigal 2.5, the security field in the metadata file expTab.txt can be used to hide (or restrict access to) an experiment. (See the change experiment status tool).

Each experiment directory must contain:

- The Experiment Table entry (expTab.txt) for this experiment. This file is created automatically if you use the tools described in the creating experiments section.

Each experiment directory may contain:

- The metadata file fileTab.txt for this experiment. This table must have one entry for each Madrigal format file in the experiment. This file is created automatically if you use the tools described in the creating experiments section.
- One or more datasets in Madrigal (or any valid Cedar) format.
- Subdirectories containing an html file named index.html . Links to these files will show up the experiment page.
- Html pages in the main directory, and again links to these files will show up the experiment page.
- Plots or other files related to individual records in the dataset. Links to these files appear next to the appropriate record in the summarizeCedarFile.cgi page. These files must be located in the subdirectory "plots/[file name]/records" under the experiment directory. In order for the script to determine which file goes with which record, the files must include a five digit number somewhere in its name. For example, plot00027.gif would appear as a link next to record 27 in summarizeCedarFile.cgi. More than one file can be associated with a given record.

An essential role for a Madrigal administrator is to run the script *madroot*/bin/updateMaster on a regular basis. This script performs a number of functions:

- It gathers all the separate metadata files in the individual experiment directories into the central directory *madroot*/metadata.
- It gathers metadata files from other Madrigal sites into the central directory *madroot*/metadata.
- It updates geophysical files from the OpenMadrigal site.

It is recommended that this script be set as a cron job to run once a day.

For Madrigal sites with limited bandwidth, updateMaster can occasionally be run with the -s flag. This will skip any possible download of the latest geophysical files, and so maybe run faster. However, it is still important to run updateMaster without the -s flag on a regular basis.

A complete description of the various ways to add and modify experiments in a Madrigal database is given in the next section. The updateMaster script must be run after each change described in that section for the changes to take effect.

Note that an experiment need not contain any Madrigal files.


How Madrigal data is organized | Doc home | Madrigal home

Previous: Madrigal data model **Up:** Madrigal admin guide **Next:** Creating Madrigal data files


Creating and editing Madrigal data files | Doc home | Madrigal home

Previous: Madrigal data organization **Up:** Madrigal admin guide **Next:** Creating Madrigal experiments

Creating and editing Madrigal data files

A key element in administering the Madrigal database is the ability to create and edit Madrigal data files. An ambitious Madrigal administrator could theoretically read the Cedar database format and write their own code from scratch. However, Madrigal provides API's and examples in three languages, Python, C, and Tcl, to make this chore easier. This section describes how to create Madrigal files using each of those three languages.

Python

This section gives an introduction to using the madrigal python API to create new Cedar files, and to edit existing Cedar files. Examples are given of of both creating new files and editing existing files. Complete documentation is available as part of the Madrigal Python API documentation.

The python cedar module was written to simplify the task of creating Cedar-formatted files as much as possible. The user of this module only needs to know the following about the Cedar format:

- Cedar files are made up of a series of metadata and data records. Each data record consists of a measurement made with a single instrument over a single interval of time.
- A Cedar data record has three parts: a prolog, one-dimensional data, and two-dimensional data. The prolog defines the integration period start and end time, along with the instrument and a kind-of data code that indicates a particular data analysis algorithm. One-dimensional data describes measurements of parameters that do not vary over any spatial dimension. Two-dimensional data describes measurements that do vary over some number of spatial dimensions. Two-dimensional data must have at least one parameter that indicates a spatial dimension.
- A list of Cedar parameters and their units can be found here. A complete description of each parameter can also be seen by clicking on any parameter name in Madrigal. Parameters can be referred to in the python API either by case-insensitive mnemonics (e.g., "Gdalt") or by integer id. Parameters are set either to float values, or to the special values 'missing', 'assumed', or 'knownbad'. The values 'assumed' and 'knownbad' can only be applied to error parameters.
- In addition to data records, the Cedar format also allows two types of records that contain human-readable descriptions of the data: a catalog record which summarizes the entire file, and a header record that summarizes a subset of the file. The text in these records must be padded to a multiple of 80 characters, and should not contain line-feeds. See the Cedar format for the suggested layout of these catalog and header records.

The python cedar module hides the following details from the user:

- The details of the Cedar layout.
- The various 'flavors' of the Cedar format. The use can specify saving a file in one particular 'flavor', if desired

- The limited dynamic range of data fields, given that data is stored as 16 bit integers. An exception is raised if the user tries to set a parameter outside its dynamic range.
- The use of 'additional increment' parameters to increase of precision of a given parameter.

MadrigalCedarFile

The high level object in the cedar module is MadrigalCedarFile. This class emulates a python list, and so users may treat it just like a python list. The restriction enforced is that all items in the list must be either MadrigalCatalogRecords, MadrigalHeaderRecords, or MadrigalDataRecords. Each of these three classes supports the method getType(), which returns 'catalog', 'header', and 'data', respectively.

Using Python to add catalog and header records.

Cedar catalog and header records can be difficult to create. With the cedar module CatalogHeaderCreator, creating these records should now be much easier. Catalog and header records contain a lot of information that can be deduced from the data - this includes which parameters are present, minimum and maximums of certain parameters, and start and stop times. The only parts of the catalog or header record that can't be determined automatically are some optional descriptive text fields. With this new module, you simply pass in strings of any length if you want to fill in one of those optional descriptive text fields - the module will handle all formating for you.

Here's a list of the optional descriptive text fields for a *catalog* record:

- **principleInvestigator:** names of responsible Principal Investigator(s) or others knowledgeable about the experiment
- **expPurpose:** brief description of the experiment purpose
- **expMode:** further elaboration of meaning of experiment mode; e.g. antenna patterns and pulse sequences
- **cycleTime:** minutes for one full measurement cycle (this needs to be a number)
- **correlativeExp:** any correlative experiments (experiments with related data)
- **sciRemarks:** scientific remarks
- **instRemarks:** instrument remarks

Here's a list of the optional descriptive text fields for a *header* record:

- **kindatDesc:** description of how this data was analyzed (the kind of data)
- **analyst:** name of person(s) who analyzed this data
- **comments:** additional comments about data (describe any instrument-specific parameters)
- **history:** a description of the history of the processing of this file

Creating a new file example

```
"""createSample.py shows am example of creating an entirely new Madrigal
file using the Python cedar module.  In particular, it creates a file with
a catalog record, a header record, and two data records.  The data records
contain two 1D parameters (System temperature - SYSTMP and Transmitter
Frequency TFREQ) and five 2D parameters (GDALT, GDLAT, GLON, and TR, and DTR).
"""

import os, os.path
```

```
import types
import datetime

import madrigal.metadata
import madrigal.cedar

################## sample data #################

kinst = 30 # instrument identifier of Millstone Hill ISR
modexp = 230 # id of mode of experiment
kindat = 3408 # id of kind of data processing
nrow = 5 # all data records have 5 2D rows

SYSTMP = (120.0, 122.0)
TFREQ = (4.4E8, 4.4E8)

GDALT = ((70.0, 100.0, 200.0, 300.0, 400.0),
         (70.0, 100.0, 200.0, 300.0, 400.0))

GDLAT = ((42.0, 42.0, 42.0, 42.0, 42.0),
         (42.0, 42.0, 42.0, 42.0, 42.0))

GLON  = ((270.0, 270.0, 270.0, 270.0, 270.0),
         (270.0, 270.0, 270.0, 270.0, 270.0))

TR    = (('missing', 1.0, 1.0, 2.3, 3.0),
         ('missing', 1.0, 1.7, 2.4, 3.1))


DTR   = (('missing', 'assumed', 'assumed', 0.3, 0.7),
         ('missing', 'assumed', 0.7, 0.4, 0.5))

################# end sample data #################

newFile = '/tmp/testCedar.dat'

# create a new Madrigal file
cedarObj = madrigal.cedar.MadrigalCedarFile(newFile, True)

# create all data records -  each record lasts one minute
startTime = datetime.datetime(2005, 3, 19, 12, 30, 0, 0)
recTime = datetime.timedelta(0,60)
for recno in range(2):
    endTime = startTime + recTime
    dataRec = madrigal.cedar.MadrigalDataRecord(kinst,
                                                kindat,
                                                startTime.year,
                                                startTime.month,
                                                startTime.day,
                                                startTime.hour,
                                                startTime.minute,
                                                startTime.second,
                                                startTime.microsecond/10000,
                                                endTime.year,
                                                endTime.month,
                                                endTime.day,
                                                endTime.hour,
                                                endTime.minute,
                                                endTime.second,
                                                endTime.microsecond/10000,
                                                ('systmp', 'tfreq'),
```

```
                                                ('gdalt', 'gdlat', 'glon', 'tr', 'dtr'),
                                                nrow)

    # set 1d values
    dataRec.set1D('systmp', SYSTMP[recno])
    dataRec.set1D('tfreq', TFREQ[recno])

    # set 2d values
    for n in range(nrow):
        dataRec.set2D('gdalt', n, GDALT[recno][n])
        dataRec.set2D('gdlat', n, GDLAT[recno][n])
        dataRec.set2D('glon',  n, GLON[recno][n])
        dataRec.set2D('tr',    n, TR[recno][n])
        dataRec.set2D('dtr',   n, DTR[recno][n])

    # append new data record
    cedarObj.append(dataRec)

    startTime += recTime

# write new file
cedarObj.write()

# next, use the cedar.CatalogHeaderCreator class to add catalog and header
catHeadObj = madrigal.cedar.CatalogHeaderCreator(newFile)
catHeadObj.createCatalog(principleInvestigator="John Holt", sciRemarks="Test data only - do not
catHeadObj.createHeader(analyst="Bill Rideout", comments="Do not use this data")
catHeadObj.write()
```

Editing an existing file example

```
"""editSample.py shows am example of editing existing data in a Madrigal
file using the Python cedar module.  In particular, it edits the sample file
mil980120g.001 to increase all Ti values by a factor of 1.2
"""

import os, os.path
import types

import madrigal.metadata
import madrigal.cedar

metaObj = madrigal.metadata.MadrigalDB()

orgFile = os.path.join(metaObj.getMadroot(), 'experiments/1998/mlh/20jan98/mil980120g.003')
newFile = '/tmp/mil980120g.003'

# read the Madrigal file into memory
cedarObj = madrigal.cedar.MadrigalCedarFile(orgFile)

# loop through each record, increasing all Ti values by a factor of 1.2
for record in cedarObj:
    # skip header and catalog records
    if record.getType() == 'data':
        # loop through each 2D roow
        for row in range(record.getNrow()):
```

```
            presentTi = record.get2D('Ti', row)
            # make sure its not a special string value, eg 'missing'
            if type(presentTi) != types.StringType:
                record.set2D('Ti', row, presentTi*1.2)

# write edited file
cedarObj.write('Madrigal', newFile)
```

C

The C API to create Madrigal data files is more complex than the Python API. The best way to learn how to use it is to follow the example below. All the C methods are documented in the Madrigal C API documention in the Madrigal developer's section.

```
/*
*    Usage: testCreateRecord
*    This program creates a Madrigal file with a catalog, header, and data record
*
*/

#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <madrec.h>
#include <cedar.h>

int
main (argc, argv)
     int     argc;
     char    *argv[];
{
    Madrec *madrecp;     /* Output File - type 0-4 */
    int iotype=0, stat=0, record=0, i=0;
    int lprol=0, jpar=0, mpar=0, nrow=0, krec=0, kinst=0, kindat=0, ibyr=0,
        ibmo=0, ibdy=0, ibh=0, ibm=0, ibs=0, ibcs=0, ieyr=0, iemo=0, iedy=0,
        ieh=0, iem=0, ies=0, iecs=0;
    double kp[8] = {2.0, 3.3, 4.0, 3.0, 8.0, 7.7, 6.3, 5.0};
    double ap[8] = {10.0, 15.0, 16.0, 15.0, 24.0, 21.0, 18.0, 15.0};
    char text[161] = "";


    /* Create a madrec object for the output file */
    if ((madrecp = madrecCreate()) == (Madrec *) NULL) {
        fprintf(stderr, "create madrecw: %s\n", madrecGetError(madrecp));
        return(1);
    }

    /* Connect the output madrec object to a madrigal file */
    iotype = 20;
    /* iotype will set which of the Cedar format variations we will create */
    /* See cedarFromat.pdf for details */
    /* 20 - Madrigal file          */
    /* 21 - Blocked Binary file    */
    /* 22 - Cbf file               */
    /* 23 - Unblocked Binary file  */
    /* 24 - Ascii file             */
```

```
stat = madrecOpen(madrecp, iotype, "madout");
fprintf(stderr, "open madrecw: %s\n", madrecGetError(madrecp));

/* first, add a catalog record - lenght must be 80*n */
/* create some text */
text[160] = '\0';
for (i=0; i<160; i++)
    text[i] = ' ';
strcpy(text, "This is a catalog record line 0");
text[strlen("This is a catalog record line 0")] = ' ';
strcpy(text + 80, "This is a catalog record line 1");
text[80 + strlen("This is a catalog record line 1")] = ' ';
madrecp->recordp = cedarCreateCatalogRecord(31, 30007,
                            2001, 8, 20,
                            0, 0, 0, 0,
                            2001, 8, 21,
                            23, 59, 59, 99,
                            text);

/* append some more text to the catalog record */
for (i=0; i<160; i++)
    text[i] = ' ';
strcpy(text, "This is a catalog record line 2");
text[strlen("This is a catalog record line 2")] = ' ';
strcpy(text + 80, "This is a catalog record line 3");
text[80 + strlen("This is a catalog record line 3")] = ' ';
stat = cedarAppendCatalogRecord(&(madrecp->recordp), text);

stat = madrecPutNextRec(madrecp);
fprintf(stderr, "putNextRec madrecw: %s\n", madrecGetError(madrecp));
free(madrecp->recordp);

/* next, add a header record - lenght must be 80*n */
/* create some text */
for (i=0; i<160; i++)
    text[i] = ' ';
strcpy(text, "This is a header record line 0");
text[strlen("This is a header record line 0")] = ' ';
strcpy(text + 80, "This is a header record line 1");
text[80 + strlen("This is a header record line 1")] = ' ';
madrecp->recordp = cedarCreateHeaderRecord(31, 30007,
                            2001, 8, 20,
                            0, 0, 0, 0,
                            2001, 8, 21,
                            23, 59, 59, 99,
                            2,2,
                            text);

/* append some more text to the header record */
for (i=0; i<160; i++)
    text[i] = ' ';
strcpy(text, "This is a header record line 2");
text[strlen("This is a header record line 2")] = ' ';
strcpy(text + 80, "This is a header record line 3");
text[80 + strlen("This is a header record line 3")] = ' ';
stat = cedarAppendHeaderRecord(&(madrecp->recordp), text);

stat = madrecPutNextRec(madrecp);
fprintf(stderr, "putNextRec madrecw: %s\n", madrecGetError(madrecp));
```

```
    lprol = 16;
    jpar = 3;
    mpar = 2;
    nrow = 8;
    krec = 1002;
    kinst = 210;
    kindat = 30007;
    ibyr = 2001;
    ibmo = 8;
    ibdy = 20;
    ibh = 0;
    ibm = 0;
    ibs = 0;
    ibcs = 0;
    ieyr = 2001;
    iemo = 8;
    iedy = 20;
    ieh = 23;
    iem = 59;
    ies = 59;
    iecs = 59;
    for (record=0; record<5; record++) {

        ibdy++;
        iedy++;

/* Create a Cedar record in the madrec object */
        if (madrecp->recordp != (Int16 *)NULL) {
            free(madrecp->recordp);
        }
        madrecp->recordp = cedarCreateRecord(lprol, jpar, mpar, nrow, krec,
                                             kinst, kindat, ibyr, ibmo, ibdy,
                                             ibh, ibm, ibs, ibcs, ieyr,
                                             iemo, iedy, ieh, iem, ies,
                                             iecs);

/* Set 1d parameters */
        stat = cedarSet1dParm(madrecp->recordp, 340, 12.0, 0);
        stat = cedarSet1dParm(madrecp->recordp, 354, 1.5e-20, 1);
        stat = cedarSet1dParm(madrecp->recordp, 356, 1.2e-20, 2);

/* Set 2d parms */
        stat = cedarSet2dParm(madrecp->recordp, 310, kp, 0);
        stat = cedarSet2dParm(madrecp->recordp, 335, ap, 1);

/* cedarPrintRecord(madrecp->recordp); */

        stat = madrecPutNextRec(madrecp);
        fprintf(stderr, "putNextRec madrecw: %s\n", madrecGetError(madrecp));

    }

    stat = madrecClose(madrecp);
    fprintf(stderr, "close madrecw: %s\n", madrecGetError(madrecp));

    madrecDestroy(madrecp);

    return(0);
}
```

Tcl

The Tcl API is not documented the way the Python and C API's are; users who wish to use it are referred to the source code available from the CVS repository at the OpenMadrigal site.The following example shows the creation of a Madrigal file with tcl:

```
# testWrite

# usage: testWrite

set madfile "madout"

# Create a madrec object
set status [catch mad mad1]
if {$status == 0} {
    puts "mad: [$mad1 get error]"
} else {
    puts "mad Error: [$mad1 get error]"
}

set status [$mad1 open 20 $madfile]
puts "open: [$mad1 get error]"

# Get parameter codes
cedarCode cedarCode

# Create a record
set lprol 16
set jpar 3
set mpar 2
set nrow 8
set krec 1002
set kinst 210
set kindat 30007
set ibyr 2001
set ibmo 8
set ibdy 31
set ibh 0
set ibm 0
set ibs 0
set ibcs 0
set ieyr 2001
set iemo 8
set iedy 31
set ieh 23
set iem 59
set ies 59
set iecs 59

set status [catch {set jdayno [jday 1 11 2001]} result]
puts "status = $status"
puts "result = $result"
puts "jdayno = $jdayno"
puts "date = 1 11 2001"
puts "jdayno = $jdayno"
puts "date = [jdater $jdayno]"

for {set i 0} {$i<5} {incr i} {
```

```
    set status [catch {$mad1 createRecord $lprol $jpar $mpar $nrow $krec \
      kinst $kindat \         $
      ibyr $ibmo $ibdy $ibh $$bm $ibs $ibcs \
      ieyr $iemo $iedy $ieh $$em $ies $iecs} result]
    if {$status != 0} {
        puts "createRecord Error: $result"
        exit
    }


    set status [catch {$mad1 set krec 1002}]
    if {$status == 0} {
puts "set krec: [$mad1 get error]"
    } else {
puts "set krec Error: [$mad1 get error]"
    }

    set status [catch {$mad1 set kinst 2222}]
    if {$status == 0} {
puts "set kinst: [$mad1 get error]"
    } else {
puts "set kinst Error: [$mad1 get error]"
    }

    set status [catch {$mad1 set kindat 3333}]
    if {$status == 0} {
puts "set kindat: [$mad1 get error]"
    } else {
puts "set kindat Error: [$mad1 get error]"
    }

    set status [catch {$mad1 set startTime 2002 9 32 1 1 1 1}]
    if {$status == 0} {
puts "set startTime: [$mad1 get error]"
    } else {
puts "set startTime Error: [$mad1 get error]"
    }

    set status [catch {$mad1 set endTime 2003 10 33 2 2 2 2}]
    if {$status == 0} {
puts "set endTime: [$mad1 get error]"
    } else {
puts "set endTime Error: [$mad1 get error]"
    }

    set status [catch {$mad1 set 1dParm 340 12.0 0}]
    if {$status == 0} {
puts "set 1dParm: [$mad1 get error]"
    } else {
puts "set 1dParm Error: [$mad1 get error]"
    }

    set status [catch {$mad1 set 1dParm 354 1.5e-20 1}]
    if {$status == 0} {
puts "set 1dParm: [$mad1 get error]"
    } else {
puts "set 1dParm Error: [$mad1 get error]"
    }

    set status [catch {$mad1 set 1dParm 356 1.2e-20 2}]
    if {$status == 0} {
```

```
puts "set 1dParm: [$mad1 get error]"
    } else {
puts "set 1dParm Error: [$mad1 get error]"
    }

    set parmvals [list 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0]
    set status [catch {$mad1 set 2dParm 310 $parmvals 0}]
    if {$status == 0} {
puts "set 2dParm: [$mad1 get error]"
    } else {
puts "set 2dParm Error: [$mad1 get error]"
    }

    set parmvals [list 10.0 20.0 30.0 40.0 50.0 60.0 70.0 80.0]
    set status [catch {$mad1 set 2dParm 335 $parmvals 1}]
    if {$status == 0} {
puts "set 2dParm: [$mad1 get error]"
    } else {
puts "set 2dParm Error: [$mad1 get error]"
    }

    # $mad1 printProlog


    $mad1 putNextRecord

    #$mad1 printRecord

}

$mad1 close

$mad1 destroy

exit
```

Scripts to modify existing Cedar files

The following scripts all modify existing Cedar files. All these scripts are installed in *madroot*/bin.

mergeCedarFiles

mergeCedarFiles merges specified records of a set of CEDAR files into a single file. The input file may be any of the 5 supported CEDAR formats (Madrigal, Blocked Binary, Cbf, Unblocked Binary or ASCII), and may include any mixture of prologue, header and data records. The format of the input file is determined automatically.

Usage: mergeCedarFiles [options] Options:

- -i file rec1 rec2 - Add records rec1 to rec2 of file to outFile
- -o outFile
- -t filetype - set type of outFile to fileType
    1. Madrigal
    2. Blocked Binary
    3. Cbf

4. Unblocked binary
5. Ascii

If more than one input file is present, the specified records are added in the order in which they are encountered. If output filetype is not specified, filetype = 1 (Madrigal).

mergeCedarFilesInTime

mergeCedarFilesInTime merges specified records from two CEDAR files into a single file. Unlike mergeCedarFiles, the records are inserted according to their start times, and not file by file. The input file may be any of the 5 supported CEDAR formats (Madrigal, Blocked Binary, Cbf, Unblocked Binary or ASCII), and may include any mixture of prologue, header and data records. The format of the input file is determined automatically.

Usage: mergeCedarFilesInTime [options] Options:

- -i file rec1 rec2 - Add records rec1 to rec2 of file to outFile. Two input files must be specified.
- -o outFile
- -t filetype - set type of outFile to fileType
    1. Madrigal
    2. Blocked Binary
    3. Cbf
    4. Unblocked binary
    5. Ascii

The specified records from the two files are added in order of ascending start times. If output filetype is not specified, filetype = 1 (Madrigal).

removeCedarRecords

removeCedarRecords is used to create a new cedar file from an existing one, with certain records removed. The input file may be any of the 5 supported CEDAR formats (Madrigal, Blocked Binary, Cbf, Unblocked Binary or ASCII), and may include any mixture of prologue, header and data records. The format of the input file is determined automatically.

Usage: removeCedarRecords [options] Options:

- -i infile - Only one input file must be specified.
- -o outFile - Only one output file must be specified.
- -r rec1 rec2 - A range of records to remove (inclusive). More than one range can be given.
- -t filetype - set type of outFile to fileType
    1. Madrigal
    2. Blocked Binary
    3. Cbf
    4. Unblocked binary
    5. Ascii

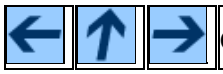
Example:

```
removeCedarRecords -i mil991102g.001 -o mil991102g.002 -r 10 12 -r 100 100
```

In this example mil991102g.002 would be a subset of the original file mil991102g.001, with records 10 through 12 removed, and record 100 removed. If output filetype is not specified, filetype = 1 (Madrigal).


Creating and editing Madrigal data files | Doc home | Madrigal home

Previous: Madrigal data organization **Up:** Madrigal admin guide **Next:** Creating Madrigal experiments


Creating and modifying Madrigal experiments | Doc home | Madrigal home

Previous: Creating Madrigal data files **Up:** Madrigal admin guide **Next:** Other admin tasks

Creating and modifying Madrigal experiments

Adding data to Madrigal involves two steps. The first step is creating Madrigal data files, as discussed in the previous section. The next is installing them on the Madrigal database, with the appropriate metadata and any additional plots or other information you want to supply.

There are a number of scripts supplied with Madrigal to simplify the task of creating or modifying experiments on Madrigal. Use these scripts if you want to:

- Create an experiment using a pre-existing Madrigal file
- Create an experiment when no Madrigal file yet exists (such as to create a real-time file)
- Modify the status of a given experiment
- Add a new file to an experiment
- Modify an existing file in an experiment
- Change the status of a file in an experiment
- Remove a file from an experiment
- Adding auxiliary plots and information to a Madrigal experiment

After changing a Madrigal experiment using one or more of the scripts above, the script *madroot*/bin/updateMaster must be run for the changes to take effect. That's because these scripts change the metadata files in the local directory, and updateMaster combines all the local metadata files into the ones in *madroot*/metadata, which the used by the user interface to Madrigal.

This page also describes how you can add auxiliary information to a given Madrigal experiment.

Create an experiment using a pre-existing Madrigal file

If you have already created a Madrigal file for a given experiment, and want to add that new experiment to Madrigal, use the script createExpWithFile.py, located in *madroot*/bin. Usage:

```
createExpWithFile.py is a script used to create a new Madrigal experiment
   based on an already existing file.  Information such as the duration of the
   experiment is obtained by analyzing the file.

Required arguments:

 --madFilename - full path to the complete Madrigal file. Basename will
   be maintained.

 --expTitle - experiment title. Use quotes if title contains spaces.

   --permission - 0 for public, 1 for private (restricted to certain IP range)
   (both the experiment and the file will set)

 --fileDesc - file decription

Optional arguments:

 --instCode - instrument code. If this argument missing, instrument code is
   taken from file, but error is thrown if more than one kinst found.

 --category - 1=default, 2=variant, 3=history, or 4=realtime.   If this argument is missing,
   1 (default) used.
```

```
 --dirName  - directory name to use for experiment.  If not given, the directory
   name will be the default name DDmmmYY[optChar].  Cannot contain "/"

 --optChar  - optional character to be added to experiment directory if no dirName
   given.  If dirName argument given, this argument ignored.  optChar
   is used if the default directory name DDmmmYY is used for
   more than one experiment created for a given instrument on a given day.
   For example, if --optChar=h for a MLH experiment on September 12, 2005,
   then the experiment directory created would be experiments/2005/mlh/12sep05h.
```

Example: If you have already created the Madrigal file mlh060120g.001 in the /tmp directory, and the experiment name is Calibration, the file should be public, and the description is "Final", you would enter:

```
/opt/madrigal/bin/createExpWithFile.py --madFilename=/tmp/mlh060120g.001 \
    --expTitle="Calibration" --permission=0 --fileDesc="Final"
```

Create an experiment when no Madrigal file yet exists

If you want to create an experiment for a Madrigal data yet to be created (such as when you want to create the Madrigal file in real-time), use the script createRTExp.py, located in *madroot*/bin. Usage:

```
createRTExpWithFile.py is a script used to create a new Madrigal experiment
   that will contain real-time files.  These real-time files are assumed not to exist
   yet.

Required arguments::

 --startDate - experiment start date in form YYYY-MM-DD

 --inst - instrument code or 3-letter Madrigal mnenonic

 --expTitle - experiment title. Use quotes if title contains spaces.

   --rtFiles - comma-separated list of realtime file basenames to be created

 --kindats - comma-separated list of ints or single int of kindats for each realtime file.
   The length and order must be the same as rtFiles.  If only one
   given, it is assumed that all rtFiles have the same kindat.

 --fileDescs - comma-separated list of file descriptions. If the file description contains spac
   quotes must be used.

Optional argument::

 --numDays - number of days the experiment is estimated to run - if not given, and no start
   and end times specified, defaults to one day.  Error raised if endDate and endTime
   also specified.

 --startTime - start time in form HH:MM:DD.  Defaults to 00:00:00

 --endDate - end day in form YYYY-MM-DD. endTime must also be specified

 --endTime - end time in form HH:MM:DD.  endDate must also be specified

   --permissions - comma-separated list of 0 for public, 1 for private (restricted to certain I
```

```
   If only one given, it is assumed it applied to all.  If this argument is not
   given, it defaults to 0 (public)

 --dirName  - directory name to use for experiment.  If not given, the directory
   name will be the default name DDmmmYY[optChar].  Cannot contain "/"

 --optChar  - optional character to be added to experiment directory if no dirName
   given.  If dirName argument given, this argument ignored.  optChar
   is used if the default directory name DDmmmYY is used for
   more than one experiment created for a given instrument on a given day.
   For example, if --optChar=h for a MLH experiment on September 12, 2005,
   then the experiment directory created would be experiments/2005/mlh/12sep05h.

 --security - overall experiment access.  0 for public, 1 for private, -1 for ignore.
 Defaults to public (0)
```

Example: If you plan to create two real-time Madrigal files called mlh021001a.000 and mlh021001b.000 for the Millstone Hill Radar for an experiment planned to run 2 days, with both files having the kindat 3410, you would enter:

```
/opt/madrigal/bin/createRTExp.py --startDate=2002-10-01 --numDays=2 --inst=mlh --expTitle="test
    --rtFiles=mlh021001a.000,mlh021001b.000 --kindats=3410 \
    --fileDescs="preliminary - single pulse,preliminary - alternating code"
```

Modify the status of a given experiment

All of the experiment attributes can be changed by the script changeExpStatus.py, located in *madroot*/bin. The most common reason you'd want to run this script is to change an experiment's security, with the options being 0 (public), 1 (limited by IP address), and -1 (ignored, or hidden from everyone). To completely remove an experiment, simply delete the directory. However, setting security to -1 allows you to bring the experiment back at some later time by running changeExpStatus.py again.

A number of other attributes can also be modified, but most are set automatically, and should not need modification.

Usage:

```
changeExpStatus.py is a script used to change the status of an
 existing Madrigal experiment.  The following attributes can be changed:

 expUrl
   experiment name
   siteID
   start date
   start time
   end date
   end time
   instrument code
   security (public, private. ignore)

Required argument:

 --expDir - full path to experiment directory. Example:
   "/opt/madrigal/experiments/1998/mlh/20jan98"

Optional arguments - set these to change an experiment attribute:
```

```
  --expUrl - must be in form <cgi base>/madtoc/YYYY/<3 letter lower case inst code>/<expDir>
  example: http://www.haystack.mit.edu/cgi-bin/madtoc/1997/mlh/03dec97g

--expName - experiment name.  Quotes required if contains spaces.  Example: "World Day"

--siteID - Madrigal siteID of where data will be stored.  Error raised if not the siteID
  of the local Madrigal site. Example: 4

--startDate - new start date of experiment (UT).  In form YYYY-MM-DD.  Example: 1998-01-20

--startTime - new start time of experiment (UT).  In form HH:MM:DD.  Example: 12:30:00

--endDate - new end date of experiment (UT).  In form YYYY-MM-DD.  Example: 1998-01-21

--endTime - new end time of experiment (UT).  In form HH:MM:DD.  Example: 23:30:00

--inst - new instrument code.  Example: 30

--security - new security code.  Allowed values are 0 for public, 1 for private (limited IP ra
             -1 for ignore, 2 for archived experiment, 3 for private (limited IP range acces
             experiment.
```

Example: to change to experiment /opt/madrigal/experiments/2006/mlh/20jan to be private, you would run:

```
/opt/madrigal/bin/changeExpStatus.py --expDir=/opt/madrigal/experiments/2006/mlh/20jan06 --secu
```

Add a new file to an experiment

If you want to add a new file to an existing Madrigal experiment, use addFileToExp.py, located in *madroot*/bin. You can use this script to add a completely new file, or to update an existing one. It you update an existing file and want to change the status of the older file to history or variant, use the script changeFileStatus.py.

Usage:

```
addFileToExp.py is a script used to add a new Madrigal file to an
existing experiment.  Information such as the duration of the
experiment is updated by analyzing the file.

Required arguments:

    --madFilename - full path to the complete Madrigal file. Basename will
                    be maintained.

    --expDir - full path to experiment directory. Example:
               "/opt/madrigal/experiments/1998/mlh/20jan98"

    --permission - 0 for public, 1 for private (restricted to certain IP range)

    --fileDesc - file decription

Optional arguments:

    --category - 1=default, 2=variant, 3=history, or 4=realtime. If this argument is missing,
                 1 (default) used.
```

Example: If you want to add the new file /tmp/mlh060120g.002 to an existing experiment, you would enter:

```
/opt/madrigal/bin/createExpWithFile.py --madFilename=/tmp/mlh060120g.002 \
     --expDir=/opt/madrigal/experiments/2006/mlh/20jan06 --permission=0 \
     --fileDesc="alternative analysis"  --category=1
```

Modify an existing file in an experiment

If you want to modify an existing file in a Madrigal experiment, use updateFileInExp.py, located in *madroot*/bin. Note that if the modification in the file is significant, it is preferable to make the old file a history file using changeFileStatus.py, and to add a new file with a different name using addFileToExp.py.

Usage:

```
updateFileInExp.py is a script used to update an existing Madrigal file in an
   existing experiment. Information such as the duration of the
   experiment is updated by analyzing the file. This script is use to replace
   an existing Madrigal file. Use addFileToExp.py to add a new file, and
 changeFileStatus.py to change any file attribute.

Required arguments:

 --madFilename - full path to the new version of the Madrigal file. Basename will
   be maintained.

 --expDir - full path to experiment directory. Example:
 "/opt/madrigal/experiments/1998/mlh/20jan98"
```

Example: To modify the existing file /opt/madrigal/experiments/2002/01oct02/mlh021001a.000 with the file /tmp/mlh021001a.000, you would enter:

```
/opt/madrigal/bin/updateFileInExp.py --madFilename=/tmp/mlh021001a.000 \
 --expDir=/opt/madrigal/experiments/2002/01oct02
```

Change the status of a file in an experiment

If you want to change the status of any existing file in an experiment, such as to make a default file into a history file, use changeFileStatus.py, located in *madroot*/bin. Usage:

```
changeFileStatus.py is a script used to change the status of an
existing Madrigal file.  The file permission, the file description,
or the file category can be changed.

Required arguments:

    --filename - basename of existing Madrigal file.

    --expDir - full path to experiment directory. Example:
               "/opt/madrigal/experiments/1998/mlh/20jan98"

Optional arguments - set these to change a file attribute:

    --permission - 0 for public, 1 for private (restricted to certain IP range)
```

```
    --fileDesc - file decription

    --category - 1=default, 2=variant, 3=history, or 4=realtime
```

Example: If you want to change the status of /opt/madrigal/experiments/2002/01oct02/mlh021001a.000 to be a history file, you would enter:

```
/opt/madrigal/bin/changeFileStatus.py --filename=mlh021001a.000 \
    --expDir=/opt/madrigal/experiments/2002/01oct02 \
    --category=3
```

Remove a file from an experiment

To completely remove a file from an existing experiment, rather than simply make it a history file, use removeFileFromExp.py, located in *madroot*/bin. Usage:

```
removeFileFromExp.py is a script used to remove an existing Madrigal file from an
existing experiment. Information such as the duration of the
 experiment is updated by analyzing the remaining files.

Required arguments:

 --filename - basename of the Madrigal file to be removed.

 --expDir - full path to experiment directory. Example:
 "/opt/madrigal/experiments/1998/mlh/20jan98"
```

Example: To remove the file /opt/madrigal/experiments/2002/01oct02/mlh021001a.000, you would enter:

```
/opt/madrigal/bin/removeFileFromExp.py --filename=mlh021001a.000 \
     --expDir=/opt/madrigal/experiments/2002/01oct02
```

Adding auxiliary plots and information to a Madrigal experiment

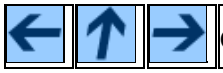
Since Madrigal is a web-based application, you can also display auxiliary plots and information about your experiment as web pages. If you add additional web documents according to the rules below, these documents will also show up through the standard Madrigal interface.

- Subdirectories containing an html file named index.html . Links to these files will show up the experiment page. The page title will be displayed as a link.
- Html pages in the main directory, and again links to these files will show up the experiment page. The page title will be displayed as a link.
- Plots or other files related to individual records in the dataset. Links to these files appear next to the appropriate record in the summarizeCedarFile.cgi page. These files must be located in the subdirectory "plots/[file name]/records" under the experiment directory, where "file name" is the base name of the experiment record.. In order for the script to determine which file goes with which record, the files must include a five digit number somewhere in its name. For example, plot00027.gif would appear as a link next to record 27 in summarizeCedarFile.cgi. More than one file can be associated with a given record.
    - If you want to create individual record plots for an incoherent scatter radar with the standard parameters Te, Ti, Ne, and ion velocity, simply run the script createRecordPlots.py. For example, to create all the individual record plots for the Madrigal file mlh980120g.001, you

would run a command similar to the following:

```
/opt/madrigal/bin/createRecordPlots.py /opt/madrigal/experiments/1998/mlh/20jan98/
```

As an example of plots associated with individual records, lets say you have a Madrigal experiment in the directory /opt/madrigal/experiments/2002/01oct02, with a file named mlh021001a.001. If this file had 10 records, you could then create 10 plot files called plot001.png through plot010.png. You would then create the subdirectories plots/mlh021001a.001/ under the main directory /opt/madrigal/experiments/2002/01oct02/, and put those 10 plot files there.


Creating and modifying Madrigal experiments | Doc home | Madrigal home

Previous: Creating Madrigal data files **Up:** Madrigal admin guide **Next:** Other admin tasks


Other administrative tasks | Doc home | Madrigal home

Previous: Creating Madrigal experiments **Up:** Madrigal admin guide **Next:** User access logging

Other administrative tasks

Some other tasks a Madrigal administrator might need to do:

- Set private versus public access for Madrigal data files
- Add site-specific links or text to the Madrigal homepage
- Allow or disallow user-added notes to experiments
- Modify settings in the madrigal.cfg file
- Add documentation specific to your site to Madrigal documentation page
- Add site-specific rules-of the-road to the Madrigal experiment page
- Copy entire experiment directories from one Madrigal site to another
- Reset experiment start and end times according to data in the experiment files
- Registering interest in an experiment or an intrument
- Mounting additional hard drives to expand the experiments disc capacity
- Rebuild local cached HDF5 files

Contact the OpenMadrigal administrator if some other administrative issue arises.

Set private versus public access for Madrigal data files

This new feature of Madrigal allows data files to be made either publicly available, or restricted to a limited set of IP address listed in a file.

To disable this feature and make all your data public, run the following from the $MADROOT directory:

```
$MADROOT/bin/setAccess $MADROOT/experiments public
```

To enable this feature, you must first create a file called "trustedIPs.txt" under MADROOT. This file should contain the IP addresses of hosts considered to be part of the private group, or partial IPs if whole sub-nets should be included. This file is not created during installation. Anyone whose browser's IP matches this list will be able to view files marked as private. For example, if trustedIPs.txt contains the line 132.197.*, any IP address that begins 132.197 will have private access.

Also, if the fileTab.txt file for that experiment can be overwritten by the web server, anyone whose browser's IP matches this list will be able to to change the access of any file between public and private from the madExperiments listing page.

To change a large number of experiments to be either public or private, run the following script from the $MADROOT directory:

```
$MADROOT/bin/setAccess dirPath [public || private]
```

where dirPath is the full path name of any directory in $MADROOT/experiments, and the second argument is either public or private. This script will set all experiments in dirPath and below to be public or private.

The access permission for any given file is set in the fileTab.txt file. See metadata documentation for details.

Add site-specific links or text to the Madrigal homepage

If you want, you can now add some site-specific links or text to the Madrigal homepage on your site. The html you add will appear under the **OpenMadrigal** link in the left column of the page.

To add html, simply create a file in the MADROOT directory called *siteIndex.html*. This file should not contain any BODY, HEAD, or HTML tags - it should only contain html to be inserted. If you want to integrate your links in with the list above it, make your links list items as in the example below:

```
<LI><A HREF="http://jro.igp.gob.pe/english/index.htm">Jicamarca staff</A></LI>
<LI><A HREF="http://jro.igp.gob.pe/english/index.htm">Jicamarca news</A></LI>
```

After the siteIndex.html file is ready, cd to MADROOT and run the following script to install it:

```
tclsh configureHtml
```

To make a change, simply edit siteIndex.html and rerun the script above.

Allow or disallow user-added notes to experiments

This feature of Madrigal allows users to append notes from the madExperiments search results page. This feature is only enabled if the web-server has write permission in the particular MADROOT/experiments directory where the experiment data is located. To allow this feature for all data, set all directory permissions below MADROOT/experiments to be writable by the web server. To disallow this feature for all data, set all directory permissions below MADROOT/experiments to not be writable by the web server.

As mention under editing madrigal.cfg, the NOTESMANAGER parameter allows a user to be notified each time a note is added. This parameter can be commented out if no notification is required.

Modify settings in the madrigal.cfg file

The following settings in the *madroot*/madrigal.cfg file can be modified at any time and the changes will take effect immediately:

- CONTACT - Mailto link of contact person(s) for this madrigal installation. Multiple email addresses may be included if separated by commas.
- MAILSERVER - Name of mailserver (possibly localhost if running sendmail)
- NOTESMANAGER - If you want someone to be notified whenever a user adds a note to an experiment, uncomment this optional line and add the email address. See below for a discussion of the optional notes feature and how to configure it after installation.
- MAXGLOBALQUERIES - The maximum number of global queries you want to allow to run on your webserver at any one time. Since a global query can take minutes or even hours to run, setting this value will limit the number of these background jobs running on your webserver. Users who request a global query when the server is at this maximum already will get a message requesting them to resubmit the query later.
- MAXTEMPREPORTS - Sets the maximum size of the tempReports directory in GB. If not set, defaults to 2 GB.

The following settings in the *madroot*/madrigal.cfg file can be modified, but the script *madroot*/configureHtml must be run afterwards for the changes to take effect:

- HTMLSTYLE - Body style of html pages
- INDEXHEAD - Heading in the top-level Madrigal page

Add documentation specific to your site to Madrigal documentation page

If you want to add any documentation pages specific to your site to the Madrigal documentation pages, simply put them in the directory *madroot*/doc/siteSpecific. If you do not yet have the file *madroot*/doc/siteSpecific.html, cp *madroot*/doc/siteSpecific_template.html to *madroot*/doc/siteSpecific.html. The edit *madroot*/doc/siteSpecific.html to contain links to your documents. These links will be in the form "siteSpecific/<your document name>".

Run " tclsh configureHtml" from the madroot directory to install these modified files on your web server.

The Site Specific Documentation link on the main documentation page will then take users to your siteSpecific.html page.

Add site-specific rules-of the-road to the Madrigal experiment page

By default, users see only the Cedar rules-of-the-road on the data access page when they first enter the Madrigal site. If you want your users to see rules-of the-road specific to your Madrigal site when the reach the Madrigal experiment page, simply create a text file with your rules called **local_rules_of_the_road.txt** in the *madroot* directory.

Copy entire experiment directories from one Madrigal site to another

If you want to copy data from one Madrigal site into yours (such as is done during the standard installation with the test data from Haystack) copy all the desired data from another Madrigal site into the appropriate directory in $MADROOT/experiments. Then follow these steps:

1. cd $MADROOT
2. ./configureExperiments
3. $MADROOT/bin/updateMaster

The script configureExperiments automatically edits the metadata files, changing the site id to your site id. This same procedure also works for moving experiments between experiments[0-9]* directories.

Reset experiment start and end times according to data in the experiment files

Sometimes the experiment start and end times in the metadata (the expTab.txt file) are manually edited, and do not match the start and end times of the data found in the files. The script updateExpTimes.py will examine the start and end times of all the files in an experiment, and set the experiment metadata start and end times to be the earliest and latest times found. Usage:

```
$MADROOT/bin/updateExpTimes.py [experiment_directory]
```

If optional argument *experiment_directory* given, then it will only update all experiments found in that directory. Default is to update the entire database. Only default or real time files are used to determine the times. Example:

```
$MADROOT/bin/updateExpTimes.py $MADROOT/experiments/1998
```

will update all experiment times for the year 1998.

Registering interest in an experiment or an instrument

With Madrigal 2.6, users can register interest in an existing experiment or in an instrument. When a data file is updated in that experiment using the standard admin scripts, by default all users registered for that experiment or instrument are sent an email. To ensure that is possible, you simply need to make sure the files *$MADROOT/metadata/userdata/regExp.txt* and *$MADROOT/metadata/userdata/regExp.txt* are writable by the web server. If you get a request from a user to unregister them from an experiment or an instrument, simply delete that line with their email from that file.

Mounting additional hard drives to expand the experiments disc capacity

With Madrigal 2.6, administrators can expand the disc space available for experiments by mounting a directory named *experiments[0-9]** in the Madroot directory. For example, you can add a directory called *experiments2* in the Madroot directory and use it for a mount point for additional space for Madrigal experiments. After you create this additional mount, you will need to create a soft link in your web server document root with a soft link to that new directory. For example, say you do the following:

- You create a new mount called *experiments2* in your Madroot directory */opt/madrigal*.
- Be sure to set the permission of this new directory so that the Madrigal administrator and the web server can read and write in it.
- Your web server document root is */usr/local/apache2/htdocs/madrigal* (set by the **MADSERVERDOCABS** field in madrigal.cfg)
- You will need to create a soft link by *cd /usr/local/apache2/htdocs/madrigal* and then *ln -s /opt/madrigal/experiments2 experiments2*.

You can now make use of this new directory in two ways:

1. You can move experiments or whole directory trees of experiments between different *experiments[0-9]** directrories as described in Copy entire experiement directories section (remember to run configureExperiments and updateMaster afterwards). Note, however, the experiments with geophysical data should not be moved (years 1950, 1957, and 1963).
2. When running the scripts createExpWithFile.py or createRTExp.py, use the *--experimentsDirNum* argument to choose the new directory to use. By default the base *experiments* directory will be used. In the example above, the option *--experimentsDirNum*=2 would cause *experiments2* to be used.

Rebuild local cached HDF5 files

As described in the initial installation section, Madrigal creates cached HDF5 versions of all default Cedar files to improve performance. There is a file called MADROOT/cachedFiles.ini that allows an administrator to create these cached HDF5 files with extra derived parameters and formats in them. These can be set according the the instrument (kinst) and kind of data (kindat). If after installation, you decide you want to modify these cached HDF5 files, then do the following:

1. Modify the MADROOT/cachedFiles.ini file (described in initial installation)
2. run *MADROOT/bin/createCachedFiles.py --overwrite*

The full usage of *createCachedFiles.py* is

```
createCachedFiles.py [--inst=<instList>  --path=<expPath> --includeNonDefault --ini=<iniFile> -
   By default all instruments will be included.  Use --inst=<comma delimited kinst list> to onl
   By default, all experiment directories will be included.  Use --path to limit to a particula
   By default only default files will be cached.  Use --includeNonDefault to include all files.
   By default, extra parameters and formats are added by the ini file $MADROOT/cachedFiles.ini.
 --ini=<iniFile> to specify an alternative ini file.  See madrigal.data.MadrigalFile._parseCach
 the ini file format.
 -h or --help - print usage and exit
```



Other administrative tasks | Doc home | Madrigal home

Previous: Creating Madrigal experiments **Up:** Madrigal admin guide **Next:** User access logging


User access logging | Doc home | Madrigal home

Previous: Other admin tasks **Up:** Madrigal admin guide **Next:** Installation script breakdown

User access logging

Madrigal now logs each time a user accesses data from a Madrigal data file. This log file is stored in *madroot*/metadata/userdata/access_<YYYY>.log. Each line in the log file has five comma-separated fields:

1. User name
2. User email
3. User affiliation
4. Time stamp <YYYY-MM-DD HH-MM-SS>
5. Full path the Madrigal file accessed.

Sample:

```
Bill Rideout,brideout@haystack.mit.edu,MIT Haystack,2006-01-03 10-10-09,/opt/madrigal/experimen
Bill Rideout,brideout@haystack.mit.edu,MIT Haystack,2006-01-03 10-10-10,/opt/madrigal/experimen
Bill Rideout,brideout@haystack.mit.edu,MIT Haystack,2006-01-03 10-10-17,/opt/madrigal/experimen
```

When a user is accessing Madrigal through the web, they will be asked for this information the first time they use the Data Access page. No attempt is made to verify the data they enter. This data is then stored as a cookie, and information from that cookie is used for this logging.

All the latest releases of the remote API also require the caller to supply these additional fields. However, if a user makes use of an old version of the remote API without these fields, that call will continue to work. In that case, the access will be logged as "Unknown".

The log file will automatically roll over once a year.


User access logging | Doc home | Madrigal home

Previous: Other admin tasks **Up:** Madrigal admin guide **Next:** Installation script breakdown


Breakdown of the Madrigal installation script | Doc home | Madrigal home

Previous: User access logging **Up:** Madrigal admin guide **Next:** Developer's toc

Breakdown of the Madrigal installation script

The main Madrigal installation script is *madroot*/installMadrigal. On this page the main parts of that script are described.

checkConfig - this tcl script checks the validity of the entries in the madrigal.cfg file.

setPermissions - this tcl script sets appropriate file permissions

configureSource - this tcl script edits any scripts in the source directory that need hard-coded keywords set in the madrigal.cfg file. It also installs some files needed by the IRI model.

Autotools (configure, make, make install) then runs from the *madroot*/source directory and compiles and installs:

> **geo library** - the Fortran geo library is next built using the Fortran compiler specified in madrigal.cfg. This library contains a large number of geometric and scientific routines called by the C library madrec. The build directory is *madroot*/source/madf/geolib. The library is installed in *madroot*/lib. To rebuild this library manually, cd to that directory and run *make* and *make install*. The application looker1 is also built from Fortran. The build directory is *madroot*/source/madf/applications. The executable is installed in *madroot*/bin. To rebuild this program manually, cd to that directory and run *make* and *make install*.
>
> **madrec library** - the C madrec library is next built using the C compiler specified in madrigal.cfg. This library is the heart of Madrigal - it reads and writes Madrigal files, and derives all Madrigal parameters. Python and Tcl all call this library. The build directory is *madroot*/source/madc/madrec. The library is installed in *madroot*/lib. To rebuild this library manually, cd to that directory and run *make* and *make install*.
>
> **madtclsh application** - the madtclsh application is a tclsh extension with capabilities added from the madrec library. It is installed in *madroot*/bin, and is the application that runs the tcl scripts. The build directory is *madroot*/source/madc/madtcl. To rebuild this program manually, cd to that directory and run *make* and *make install*.

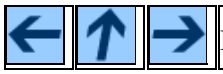
configureHtml - this tcl script edits the documentation files in the doc directory, replacing keywords set in the madrigal.cfg file, and copies them into the web servers document root as set in madrigal.cfg. If you as an administrator modify any documentation in the doc directory (such as to add site-specific documentation), run this script again to install your changes on the web server.

configureScripts - this tcl script edits all the python and tcl scripts in the source directory, putting the modified versions either in the web server's cgi directory, or in *madroot*/bin.

python application - the python application (version 2.5.2) is built next.The large majority of Madrigal is built on python, which is in turn built on the madrec and geo libraries. Madrigal now automatically installs the following external modules in python: pyxml, numpy, and matplotlib.

configureExperiments - this tcl script edits all the metadata in each experiment to be sure the site id is the local one. This script also allows you to copy an experiment directory from one Madrigal site to another.

updateMetadata - the final step in the installation of Madrigal is the updating of metadata through the updateMaster and other scripts.


Breakdown of the Madrigal installation script | Doc home | Madrigal home

Previous: User access logging **Up:** Madrigal admin guide **Next:** Developer's toc


Madrigal developer's guide | Doc home | Madrigal home

Previous: Installation script breakdown **Up:** Doc home **Next:** Internal API overview

Madrigal developer's guide

This section of the Madrigal documentation is meant for developers working to extend or modify Madrigal. It is not meant for users of Madrigal or Madrigal administrators. This section documents the API written in various languages that underlie Madrigal, and the Cedar/Madrigal file formats.



Madrigal internal API overview	Doc home	Madrigal home

Previous: Madrigal developer's guide **Up:** Madrigal developer's guide **Next:** Python API

Madrigal internal API overview

The Madrigal server API's allow programmers to write programs which access CEDAR format files and Madrigal metadata. Unlike the Madrigal remote access API's, these API's run locally on the Madrigal server, and are the heart of the Madrigal server.

There are five applications programming interfaces (API's) to madrigal: C, Python, Fortran, Tcl, and Matlab. The vast majority of Madrigal is based directly on the C-language API, which is the only language to directly read and write Cedar format files. The Cedar format file reading and writing through both Python and Tcl are built on top of the C API. To use these API's you should be familiar with the CEDAR File Format .

The C-language API has two major components: madrec and maddata. The madrec module is file-oriented. It supports all four versions of the CEDAR format in addition to the Madrigal version of the CEDAR format. The maddata module is a higher level interface into Madrigal data, and treats derived parameters and measured parameters from files as the same, and is the basis for isprint output. The maddata module is also meant to be easily extensible as described in the C API documentation. The C API is compiled into the madrec library, which is installed in *madroot*/lib.

The python API is also built on the C API, and is used for the majority of cgi and administrative scripts, and the entire remote access interface. Madrigal installs its own version of python under *madroot*, and the Madrigal python API is installed under site-packages. In general, it is the easiest language to use to implement any functionality needed. With the release of Madrigal 2.4, there is now an easy-to-use python API to create and modify Cedar file formats. See the section of the administrative manual on creating Madrigal files for details. This internal python API should not be confused with the remote python API - the remote python API can be run from anywhere and simply connects over the internet to existing Madrigal servers; this internal python API is part of the Madrigal server itself.

The Tcl API is no longer being expanded, but is still used for a few of the cgi scripts and some of the administrative scripts. The tcl executable with the Madrigal extensions is installed as *madroot*/bin/madtclsh.

The Matlab interface is simply a somewhat higher speed version than the remote Matlab API , but can only run locally. The administrator controls whether and where it is installed, as described in the installation page.

There are two parts to the Fortran API now included in Madrigal. The first part is simply a wrapper around the C API (and indeed is simply Fortran calls that call the C API). The second part of the Fortran API includes a number of scientific methods that the C API calls. This second part of the Fortran API is compiled into the geo library, which is installed in *madroot*/lib.

The following is the Python internal API documentation:

HappyDoc Generated Documentation

Modules and Packages

admin	The admin module contains all administrative classes relating to the madrigal python api.
cedar	cedar is the module that allows the creation and editting of Cedar (and possible future format) files.
data	data is the module that interfaces to madrigal data files, or to Cedar standards about data.
metadata	The metadata module provides access to all metadata about one particular madrigal database.
openmadrigal	The openmadrigal module provides access to all OpenMadrigal installations via http and to OpenMadrigal CVS.
ui/	
__bgReport	__bgReport is the module that runs in a separate process to create a background report.
__init__	Directory of all modules related to madrigal user interface.
isprintExe	isprintExe is a private module designed to get output strings from the maddata engine in isprint format.
madrigalPlot	madrigalPlot is the module that produces plots of Madrigal data.
report	report is the module that creates reports on Madrigal data.
userData	userData is responsible for interfacing to all persisted user data on the madrigal web site.
web	web is the module that interfaces to cgi madrigal web pages.

Python internal API documentation | Doc home | Madrigal home

Previous: Internal API overview **Up:** Madrigal developer's guide **Next:** IMadrigal C API

Table of Contents

admin.py

Module: admin

The admin module contains all administrative classes relating to the madrigal python api.

The main role of this module is to update the data in the Madrigal database. Also contains a notification class and a standard error handing class.

$Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $

Classes	
MadrigalDBAdmin	MadrigalDBAdmin is a class that allows modifications to be made to the Madrigal database
MadrigalError	MadrigalError is an exception class that is thrown for all known errors in using Madrigal Py lib.
MadrigalNotify	MadrigalNotify is an object used to send messages to an administrator about a Madrigal database.

Table of Contents

This document was automatically generated on Fri Dec 30 08:58:50 2005 by HappyDoc version r1_5

Table of Contents

Class: MadrigalDBAdmin **admin.py**

MadrigalDBAdmin is a class that allows modifications to be made to the Madrigal database

dbAdminObj = madrigal.admin.MadrigalDBAdmin()

expDir = dbAdminObj.createMadrigalExperiment(`/home/hyperion/brideout/mlh050429c.0` `Dummy experiment`, 0, `test exp`, 30, 1)

Non-standard Python modules used: None

Change history:

Written by Bill Rideout May. 5, 2005

Methods	
__checkOpenMadrigalMetadata__	createMadrigalExperiment
__init__	createRTExperiment
__updateGlobalMetadata__	overwriteMadrigalFile
__updateLocalMetadata__	removeMadrigalFile
__walkExpDir__	updateExpTab
addMadrigalFile	updateMaster

addWebFile
appendRTMadrigalFile
changeExpStatus
changeFileStatus
writeRTMadrigalFile

__checkOpenMadrigalMetadata__

```
__checkOpenMadrigalMetadata__ ( self )
```

__checkOpenMadrigalMetadata__ is a method that check the openmadrigal site for any updates to the following metadata files:

1. siteTab.txt - the list of all Madrigal installations 2. instTab.txt - the list of all Madrigal instruments 3. instType.txt - the list of all instrument categories

If an update is available, and the existing metadata file is an old one, it will be updated. However, if the local Madrigal adminstrator edits one of these files, then the file will not be updated. If you want to change these files, it is best to contact the OpenMadrigal development administrator (madrigal@haystack.mit.edu)

Exceptions

IOError, 'Problem with url %s' %(thisUrl)

__init__

```
__init__ ( self,  madDB=None )
```

__init__ initializes MadrigalDBAdmin

Inputs: madDB - Existing MadrigalDB object. Default = None.

Returns: void

Affects:

Sets self.__madDB to MadrigalDB object Sets self.__madInst to MadrigalInstrument object

__updateGlobalMetadata__

```
__updateGlobalMetadata__ ( self )
```

__updateGlobalMetadata__ is a private method to update metadata/expTabAll.txt and metadata/fileTabAll.txt from the main madrigal server.

__updateLocalMetadata__

```
__updateLocalMetadata__ ( self )
```

__updateLocalMetadata__ is a private method to update metadata/expTab.txt and metadata/fileTab.txt from the local metadata in the experiments[0-9]* directory

__walkExpDir__

```
__walkExpDir__ (
        self,
        arg,
        dirname,
        names,
        )
```

__walkExpDir__ is a private method called by os.path.walk. arg is a dict with keys: 1. extText = text of combined expTab.txt to be appended to 2. fileText = text of combined fileTab.txt to be appended to 3. presentCount = total experiments done so far 4. localSiteId = local site id (int)

Sets values in arg

addMadrigalFile

```
addMadrigalFile (
        self,
        expDir,
        madFilename,
        permission,
        fileDesc,
        category=1,
        kindat=None,
        notify=True,
        fileAnalyst='',
        fileAnalystEmail='',
        includeCachedHdf5=True,
        iniFile=None,
        )
```

addMadrigalFile adds a new file to an experiment using metadata read from madFilename.

Inputs:

expDir - full path to experiment directory (as returned by createMadriogalExperiment)

madFilename - full path to the complete Madrigal file. Basename will be maintained.

permission - 0 (public) or 1 (private).

fileDesc - file description

category - 1=default, 2=variant, 3=history, or 4=realtime. Default is 1 (default file)

kindat - if not None (the default), use this kindat instead of what is found in the file.

notify - if True (the default), send a message to all registered users. If False, do not.

fileAnalyst - full name of file Analyst. Default is ''

fileAnalystEmail - email of file Analyst. Default is ''

includeCachedHdf5 - if True (the default) create cached Hdf5 file. If False, do not.

iniFile - the ini file to use for creating a cached hdf5 file. If None, uses default ini file $MADROOT/cachedFiles.ini. Ignored if not includeCachedHdf5

Returns: None

Exceptions

'File %s already exists - must be deleted first' %(filename)
ValueError, ' file %s does not yet exist' %(os.path.join(expDir, 'fileTab.txt'))
ValueError, 'More than one kindat value found in file: %s' %(str(kindatList))
ValueError, 'No kindat values found in file'
ValueError, 'category must be 1=default, 2=variant, 3=history, or 4=realtime; not %s' %(str(category))
ValueError, 'fileAnalyst cannot contain a comma'
ValueError, 'fileAnalystEmail cannot contain a comma'
ValueError, 'fileDesc not a string'
ValueError, 'fileDesc string in fileTab.txt cannot contain a comma: is illegal' %(fileDesc)

ValueError, 'permission must be either 0 or 1, not %s' %(str(permission))

addWebFile

```
addWebFile (
        self,
        expDir,
        source,
        relativePath,
        )
```

addWebFile writes a non-Madrigal file meant to be displayed on the web to somewhere within a Madrigal experiment directory.

All needed directories will be created if needed.

Inputs:

expDir - full path to experiment directory

source - local web file to write to Madrigal

relativePath - path relative to expDir to write source file to. If relativePath ends with /, then basename from source used. Otherwise, basename from relativePath used.

Returns: None

Affects: writes a non-Madrigal file to expDir on Madrigal

Exceptions

ValueError, '%s not a valid experiment directory - no fileTab.txt' %(expDir)

appendRTMadrigalFile

```
appendRTMadrigalFile (
        self,
        expDir,
        rtFilename,
        rtFile,
        )
```

appendRTMadrigalFile allows appending to a realtime Madrigal file.

Fails if rtFilename does not match one listed in fileTab.txt. If the file doesn't exist, will create it, and will then check that the format is unblocked binary. This is because unblocked binary is written by logical record, without physical records that may later be modified. If the file does exist, rtFile will be appended with checking the file format.

Inputs:

expDir - full path to experiment directory (as returned by createRTExperiment)

rtFilename - basename of realtime file to be writtem

rtFile - a string containing the new realtime file contents

Returns: None

Raises exception if rtFilename does not match one listed in fileTab.txt, or if format != unblocked binary when new file is created.

Exceptions	
	ValueError, 'Expected file %s to be UnblockedBinary, is %s' %(os.path.join(expDir, rtFilename), filetype) ValueError, 'Filename %s not found in fileTab.txt' %(rtFilename) ValueError, 'Unable to open fileTab.txt in %s' %(expDir)

changeExpStatus

```
changeExpStatus (
        self,
        expDir,
        expUrl=None,
        expName=None,
        siteID=None,
        startDatetime=None,
        endDatetime=None,
        inst=None,
        security=None,
        PI=None,
        PIEmail=None,
        )
```

changeExpStatus is used to change attributes in expTab.txt. If None, no change.

Inputs:

expDir - full path to experiment directory. Required. Example: "/opt/madrigal/experiments/1998/mlh/20jan98". If None, do not change.

expUrl - must be in form <cgi base>/madtoc/YYYY/<3 letter lower case inst code>/<expDir> example: http://www.haystack.mit.edu/cgi-bin/madtoc/1997/mlh/03dec97g. If None, do not change.

expName - experiment name. Quotes required if contains spaces. Example: "World Day" If None, do not change.

siteID - Madrigal siteID (int) of where data will be stored. Error raised if not the siteID of the local Madrigal site. Example: 4. If None, do not change.

startDatetime - new start datetime of experiment (UT). If None, do not change.

endDatetime - new end datetime of experiment (UT). If None, do not change.

inst - new instrument code (int). Example: 30. If None, do not change. Prints warning if not found in instTab.txt

security - new security code. Allowed values are 0 for public, 1 for private (limited IP range access) -1 for ignore, 2 for archived experiment, 3 for private (limited IP range access) archived experiment. If None, do not change.

PI - name of PI. If None, no change

PIEmail - PI email. If None, no change

Exceptions

ValueError, 'Setting experiment to a siteID %i different from this site\'s id %i' %(siteID, self.__madDB.getSiteID())
ValueError, 'The experiment url you are setting this experiment to conflicts with experiment directory %s' %(expUrl, expDir)
ValueError, 'Unable to open expTab.txt in %s' %(expDir)
ValueError, 'expTab.txt in %s has %i experiments, should have exactly 1' %(expDir, expTabInfo.getExpCount())

ValueError, 'security must be in (-1, 0, 1, 2, 3), not %i' %(inst)
ValueError, 'startDatetime %s must be before endDatetime %s' %(str(startDatetime), str(endDatetime))

changeFileStatus

```
changeFileStatus (
        self,
        expDir,
        filename,
        category=None,
        fileDesc=None,
        permission=None,
        kindat=None,
        fileAnalyst=None,
        fileAnalystEmail=None,
        )
```

changeFileStatus is used to change category, fileDesc, or permission of a register file in fileTab.txt.

Inputs:

expDir - full path to experiment directory

filename - basename of existing Madrigal file already registered in fileTab.txt

permission - 0 (public) or 1 (private). If None (default), leave unchanged.

fileDesc - file description. If None (default), leave unchanged.

category - 1=default, 2=variant, or 3=history. If None (default), leave unchanged.

kindat - kindat (int). If None (default), leave unchanged.

fileAnalyst - name of file analyst. If None (default), leave unchanged.

fileAnalystEmail - email of file analyst. If None (default), leave unchanged.

Exceptions

ValueError, 'Madrigal file %s not found in %s' %(filename, os.path.join(expDir, 'fileTab.txt'))
ValueError, 'Unable to open fileTab.txt in %s' %(expDir)

createMadrigalExperiment

```
createMadrigalExperiment (
        self,
        madFilename,
        expTitle,
        permission,
        fileDesc,
        instCode=None,
        category=1,
        optChar='',
        dirName=None,
        kindat=None,
        experimentsDirNum=None,
        PI='',
        PIEmail='',
        fileAnalyst='',
        fileAnalystEmail='',
        includeCachedHdf5=True,
        iniFile=None,
        notify=True,
        )
```

createMadrigalExperiment creates a new experiment on Madrigal using metadata read from madFilename.

Inputs:

madFilename - full path to the complete Madrigal file. Basename will be maintained.

expTitle - experiment title

permission - 0 (public) or 1 (private) or -1 (ignore).

fileDesc - file description

instCode - instrument code. If default (None), instrument code is taken from file, but error is thrown if more than one kinst found.

category - 1=default, 2=variant, 3=history, or 4=realtime. Default is 1 (default file)

optChar - optional character to be added to experiment directory if no dirName given. If dirName argument given, this argument ignored. optChar is used if the default directory name

DDmmmYY is used for more than one experiment created for a given instrument on a given day. For example, if --optChar=h for a MLH experiment on September 12, 2005, then the experiment directory created would be experiments/2005/mlh/12sep05h.

dirName - directory name to use for experiment. If None (the default), the directory name will be the default name DDmmmYY[optChar]. Cannot contain "/"

kindat - if not None (the default), use this kindat instead of what is found in the file.

experimentsDirNum - the number to be appended to the experiments directory, if experiments directory being used is of the form experiments[0-9]* instead of just experiments. For example, if experimentsDirNum is 7, then the experiment would be created in MADROOT/experiments7 instead of MADROOT/experiments.

PI- full name of principal investigator. The default is ''

PIEmail - email of principal investigator. The default is ''

fileAnalyst -full name of file analyst. The default is ''

fileAnalystEmail - email of file analyst,. The default is ''

includeCachedHdf5 - if True (the default) create cached Hdf5 file. If False, do not.

iniFile - the ini file to use for creating a cached hdf5 file. If None, uses default ini file $MADROOT/cachedFiles.ini. Ignored if not includeCachedHdf5

notify - if True (the default), send a message to all registered users. If False, do not.

Returns:

Full path to directory created

Exceptions

IOError, 'Directory %s already exists' %(expDir)	ValueError, 'category must be 1=default, 2=variant, 3=history, or 4=realtime; not	ValueError, 'no such directory %s' %(experimentsDir) ValueError,

		'More than one kindat value found in file: %s' %(str(kindatList)) ValueError, 'More than one kinst value found in file: %s' %(str(kinstList)) ValueError, 'No kindat values found in file' ValueError, 'No kinst values found in file' ValueError, 'PI cannot contain a comma' ValueError, 'PI not a string' ValueError, 'PIEmail cannot contain a comma' ValueError, 'PIEmail not a string' ValueError, 'Unable to find mnemonic for kinst %i' %(instCode)	%s' %(str(category)) ValueError, 'dirName must be base directory name, not %s' %(dirName) ValueError, 'expTitle cannot contain a comma' ValueError, 'expTitle not a string' ValueError, 'fileAnalyst cannot contain a comma' ValueError, 'fileAnalyst not a string' ValueError, 'fileAnalystEmail cannot contain a comma' ValueError, 'fileAnalystEmail not a string' ValueError, 'fileDesc cannot contain a comma' ValueError, 'fileDesc not a string'	'optChar must be an empty or a one character string, not %s' %(str(optChar)) ValueError, 'permission must be either 0 or 1 or -1, not %s' %(str(permission))

createRTExperiment

createMadrigalExperiment creates a new experiment on Madrigal using metadata read from madFilename.

```
createRTExperiment (
        self,
        startTime,
        numDays,
        instrument,
        expTitle,
        rtFilenameList,
        kindatList,
        permissionList,
        fileDescList,
        optChar='',
        endTime=None,
        security=0,
        dirName=None,
        experimentsDirNum=None,
        PI='',
        PIEmail='',
        fileAnalystList=None,
        fileAnalystEmailList=None,
        notify=True,
        )
```

createRTExperiment creates a new experiment on Madrigal in preparation for realtime data.

Since the experiment is presumably not yet complete, metadata such as the duration of the experiment must be estimated. This metadata will be overwritten when the first batch file is added.

Inputs:

startTime - experiment start time. If a number, assumed to be seconds since 1/1/1970. May also be a datetime.datetime object

numDays - number of days the experiment is estimated to run. Ignored if optional endTime given.

instrument - instrument code or 3-letter Madrigal mnenonic

expTitle - experiment title

rtFilenameList - list of realtime filenames to be created

kindatList - list of ints of kindats for each realtime file. Len = len(rtFilenameList)

permissionList - list of 0 (public) or 1 (private). Len = len(rtFilenameList)

fileDescList - list of realtime file descriptions

optChar - optional character to be added to experiment directory if no dirName given. If dirName argument given, this argument ignored. optChar is used if the default directory name DDmmmYY is used for more than one experiment created for a given instrument on a given day. For example, if --optChar=h for a MLH experiment on September 12, 2005, then the experiment directory created would be experiments/2005/mlh/12sep05h.

endTime - optional end date and time of experiment. If a number, assumed to be seconds since 1/1/1970. May also be a datetime.datetime object

security - experiment security setting. If 0 (the default) public. If 1, private. If -1, entire experiment ignored. Any given file permission is the more restricted of experiment permission and file permission.

dirName - directory name to use for experiment. If None (the default), the directory name will be the default name DDmmmYY[optChar]. Cannot contain "/"

experimentsDirNum - the number to be appended to the experiments directory, if experiments directory being used is of the form experiments[0-9]* instead of just experiments. For example, if experimentsDirNum is 7, then the experiment would be created in MADROOT/experiments7 instead of MADROOT/experiments.

PI- full name of principal investigator. The default is ''

PIEmail - email of principal investigator. The default is ''

fileAnalystList - list of full names of file analysts, one for each file. If None, the default, File Analyst = ''

fileAnalystEmailList - list of emails of file analysts, one for each file. If None, the default, File Analyst email = ''

notify - if True (the default), send a message to all registered users. If False, do not.

Returns:

Full path to directory created

Exceptions

'optChar must be an	ValueError, 'expTitle cannot contain a	ValueError, 'kindatList must

		empty or a one character string, not %s' %(str(optChar)) IOError, 'Directory %s already exists' %(expDir) ValueError, '%s not a legal instrument mnemonic or code' %(str(instrument)) ValueError, '%s not a legal instrument mnemonic' %(str(instrument)) ValueError, 'Experiment start time %s after end time %s' %(str(startTime), str(endTime)) ValueError, 'PI cannot contain a comma' ValueError, 'PI not a string' ValueError, 'PIEmail cannot contain a comma'	comma' ValueError, 'expTitle not a string' ValueError, 'fileAnalyst cannot contain a comma' ValueError, 'fileAnalystEmail cannot contain a comma' ValueError, 'fileAnalystEmailList must only contain strings' ValueError, 'fileAnalystList must only contain strings' ValueError, 'fileDesc cannot contain a comma' ValueError, 'fileDescList must only contain strings' ValueError, 'filename cannot contain a comma' ValueError, 'kindatList must contain integers of value 0 (public) or 1 (private)'	contain integers' ValueError, 'length of fileAnalystEmailList not equal length of rtFilenameList' ValueError, 'length of fileAnalystList not equal length of rtFilenameList' ValueError, 'length of fileDescList not equal length of rtFilenameList' ValueError, 'length of kindatList not equal length of rtFilenameList' ValueError, 'length of permissionList not equal length of rtFilenameList' ValueError, 'no such directory %s' %(experimentsDir) ValueError, 'numDays must not be negative' ValueError, 'rtFilenameList must contain strings without /' ValueError, 'rtFilenameList must contain strings'

ValueError, 'PIEmail not a string' ValueError, 'dirName must be base directory name, not %s' %(dirName)

overwriteMadrigalFile

```
overwriteMadrigalFile (
        self,
        expDir,
        madFilename,
        notify=True,
        includeCachedHdf5=True,
        iniFile=None,
        )
```

overwriteMadrigalFile overwrites a file already registered in fileTab.txt.

Automatically updates expTab.txt with any start or end experiment times.

Inputs:

expDir - full path to experiment directory

madFilename - full path to the new Madrigal file. Basename must match that of one in fileTab.txt.

notify - if True (the default), send a message to all registered users. If False, do not.

includeCachedHdf5 - if True (the default) create cached Hdf5 file. If False, do not.

iniFile - the ini file to use for creating a cached hdf5 file. If None, uses default ini file $MADROOT/cachedFiles.ini. Ignored if not includeCachedHdf5

Returns: None

Affects: Overwrites existing Madrigal file. May modify expTab.txt with new start/end times

Exceptions

ValueError, '%s not found in %s' %(filename, os.path.join(expDir, 'fileTab.txt'))
ValueError, 'Unable to open fileTab.txt in %s' %(expDir)

removeMadrigalFile

```
removeMadrigalFile (
        self,
        expDir,
        madFilename,
        )
```

removeMadrigalFile removes a file already registered in fileTab.txt.

Automatically updates expTab.txt with any start or end experiment times.

Inputs:

expDir - full path to experiment directory

madFilename - Name of Madrigal file to be removed. Basename must match that of one in fileTab.txt.

Returns: None

Affects: Removes existing Madrigal file and removes its line from fileTab.txt. May modify expTab.txt with new start/end times

Exceptions

ValueError, 'Unable to open fileTab.txt in %s' %(expDir)

updateExpTab

```
updateExpTab ( self,  expDir )
```

updateExpTab rewrites expTab.txt based on all the Madrigal files registered in fileTab.txt.

Inputs:

expDir - full path to experiment directory

Returns: None

Affects: rewrites expTab.txt in expDir based on all the Madrigal files registered in fileTab.txt.

Exceptions	
	ValueError, 'Unable to open expTab.txt in %s' %(expDir) ValueError, 'Unable to open fileTab.txt in %s' %(expDir)

updateMaster

```
updateMaster ( self,  skipGeo=False )
```

updateMaster is a method to update the local metadata.

Replaces the former tcl script.

Gathers data from experiment directories into metadata/expTab.txt and metadata/fileTab.txt. Also gathers metadata from OpenMadrigal to update metadata/expTabAll.txt and metadata/fileAllTab.txt, to update high level metadata siteTab.txt, instTab.txt, instType.txt, madCatTab.txt, parcods.tab. Also updates geophysical data.

writeRTMadrigalFile

```
writeRTMadrigalFile (
        self,
        expDir,
        rtFilename,
        rtFile,
        )
```

writeRTMadrigalFile writes a realtime Madrigal file to a Madrigal experiment directory.

Fails if rtFilename does not match one listed in fileTab.txt.

Inputs:

expDir - full path to experiment directory (as returned by createRTExperiment)

rtFilename - basename of realtime file to be writtem

rtFile - a string containing the realtime file contents

Returns: None

Raises exception if rtFilename does not match one listed in fileTab.txt.

Exceptions

ValueError, 'Filename %s not found in fileTab.txt' %(rtFilename)
ValueError, 'Unable to open fileTab.txt in %s' %(expDir)

Table of Contents

This document was automatically generated on Thu Oct 20 16:51:50 2011 by HappyDoc version r1_5

Table of Contents

Class:
MadrigalError

MadrigalError is an exception class that is thrown for all known errors Madrigal Py lib.

Usage example:

```
import sys, traceback
import madrigal.admin

try:

    test = open('ImportantFile.txt', 'r')

except:

    raise madrigal.admin.MadrigalError('ImportantFile.txt not opened!',
                                       traceback.format_exception(sys.
                                                                  sys.exc
                                                                  sys.exc
```

Methods

__init__
__str__
getExceptionHtml
getExceptionStr

__init__

```
__init__ (
        self,
        strInterpretation,
        exceptionList,
```

```
        )
```

__init__ gathers the interpretation string along with a information from sys.exc_info().

Inputs: strIntepretation - A string representing the programmer's interpretat exception occurred

exceptionList - a list of strings completely describing the exception. Genera traceback.format_exception(sys.exc_info()[0], sys.exc_info()[1], sys.exc_i

Returns: Void.

Affects: Initializes class member variables _strInterp, _strExcList.

Exceptions: None.

__str__

```
__str__ ( self )
```

getExceptionHtml

```
getExceptionHtml ( self )
```

getExceptionHtml returns an Html formatted string c describing the exception.

Inputs: None

Returns: A formatted string ready for printing completely describing the ex

Affects: None

Exceptions: None.

getExceptionStr

```
getExceptionStr ( self )
```

getExceptionStr returns a formatted string ready for completely describing the exception.

Inputs: None

Returns: A formatted string ready for printing completely describing the ex

Affects: None

Exceptions: None.

Table of Contents

This document was automatically generated on Thu Oct 20 16:51:50 2011 by HappyDoc version r1_5

Table of Contents

Class: MadrigalNotify **admin.py**

MadrigalNotify is an object used to send messages to an administrator about a Madrigal database.

This object provides functions needed to send messages to an administrator about a Madrigal database, for now only sendAlert, which sends an email to the site administrator found is siteTab.txt (or if not possible, the admin in madrigal.cfg, and finally if all else fails, to root).

Usage example:

```
import madrigal.admin

try:

    adminObj =  madrigal.admin.MadrigalNotify()
    adminObj.sendAlert('This is important!', 'Important Message')

except madrigal.admin.MadrigalError, e:

    print e.getExceptionStr()
```

Non-standard Python modules used: None

Exceptions thrown: None - Note that MadrigalNotify tries every trick it knows to avoid throwing exceptions, since this is the class that will generally be called when there is a problem.

Change history:

Written by Bill Rideout Dec. 4, 2001

Methods

__init__
notify
sendAlert

__init__

```
__init__ ( self,  madDB=None )
```

__init__ initializes MadrigalNotify by getting some basic information from MadrigalDB and MadrigalSite.

Note that MadrigalNotify tries every trick it knows to avoid throwing exceptions, since this is the class that will generally be called when there is a problem.

Inputs: Existing MadrigalDB object, by default = None.

Returns: void

Affects: Initializes self.__binDir.

Exceptions: None.

notify

```
notify (
        self,
        email,
        message,
        subject,
        )
```

notify sends an email with the given message and title to email.

Inputs: email (string), message (string), and subject (string)

Returns: void

Affects: none

Exceptions: None.

sendAlert

```
sendAlert (
        self,
        message,
        subject=None,
        )
```

sendAlert sends an email with the given message and optional title.

Inputs: message (string), and optional title (string)

Returns: void

Affects: none

Exceptions: None.

Table of Contents

This document was automatically generated on Thu Oct 20 16:51:50 2011 by HappyDoc version r1_5

Table of Contents

Module: cedar.py

cedar

cedar is the module that allows the creation and editting of Cedar (and possible future format) files.

This module abstracts away many of the details of the Cedar file format, which may change in the future, and instead simply follows the Cedar data model (Prolog-1d parms-2d parms). For example, all values are accessed and set via doubles, so users do not need to deal with scale factors, "additional increment" parameters, etc. The Cedar file format has limited dynamic range, so for now an exception is raised if any value would voilate this limited range. Furthermore, the Cedar data model defines time in the prolog. Unfortunately, time may also be set in the data itself, leading to the possibility of inconsistent data. This module will issue warnings in that case.

This module is built on top of the Madrigal C API as found in libmadrec.so through the python extension class __Madrec

$Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $

Functions

compareParms

```
compareParms ( firstParm,  secondParm )
```

compareParms is used internally by getHeaderKodLines to order the parameters

Inputs:

firstParm - tuple of (code, parameter description, scaleFactor, units) for the first parameter

secondParm - tuple of (code, parameter description, scaleFactor, units) for the second parameter

Returns - -1 if first<second, 0 if first=second, 1 if first>second. First compare abs(code). If equal, positive smaller than negative to make error follow main parm.

floatEquals

```
floatEquals ( first,  second )
```

floatEquals is a method to replace == for comparing two floats. Two floats are equal if they are equal to one part in 10^5

floatIn

```
floatIn ( testFloat,  floatList )
```

floatIn returns True if thisFloat in floatList, where equals defined by floatEquals

isNan

```
isNan ( num )
```

isNan is my attempt to detect IEEE special values such as nan. Returns True if not a float with a real value. Works with both python 2.3 and python 2.4

Algorithm: if both (num < 10.0) and (num > -10.0) are False, return true.

setToMissing

```
setToMissing ( x )
```

a private method to generate default data

Classes

CatalogHeaderCreator	CatalogHeaderCreator is a class that automates the creation of catalog and header records
CedarParameter	CedarParameter is a class with attributes code, mnemonic, and description
MadrigalCatalogRecord	MadrigalCatalogRecord holds all the information in a Cedar catalog record.
MadrigalCedarFile	MadrigalCedarFile is an object that allows the creation and editting of Cedar files.
MadrigalDataRecord	MadrigalDataRecord holds all the information in a Cedar data record.

	MadrigalHeaderRecord	MadrigalHeaderRecord holds all the information in a Cedar header record.

Table of Contents

This document was automatically generated on Thu Apr 16 16:45:18 2009 by HappyDoc version r1_5

Table of Contents

Class: CedarParameter **cedar.py**

CedarParameter is a class with attributes code, mnemonic, and description

Methods

__init__
__str__

__init__

```
__init__ (
        self,
        code,
        mnemonic,
        description,
        )
```

__str__

```
__str__ ( self )
```

Table of Contents

This document was automatically generated on Fri Dec 30 08:58:50 2005 by HappyDoc version r1_5

Table of Contents

Class: MadrigalCatalogRecord **cedar.py**

MadrigalCatalogRecord holds all the information in a Cedar catalog record.

Methods

__init__	setText
__str__	setTimeLists
getEndTimeList	writeRecord
getKinst	
getModexp	
getStartTimeList	

__init__

```
__init__ (
        self,
        kinst,
        modexp,
        sYear,
        sMonth,
        sDay,
        sHour,
        sMin,
        sSec,
        sCentisec,
        eYear,
        eMonth,
        eDay,
        eHour,
        eMin,
        eSec,
        eCentisec,
        text,
        madInstObj=None,
        )
```

__init__ creates a MadrigalCatalogRecord.

Inputs:

kinst - the kind of instrument code. A warning will be raised if not in instTab.txt.

modexp - Code to indicate experimental mode employed. Must be a non-negative integer.

sYear,sMonth,sDay,sHour,sMin,sSec,sCentisec - experiment start time. sCentisec must be 0-99

eYear,eMonth,eDay,eHour,eMin,eSec,eCentisec - experiment end time. eCentisec must be 0-99

text - string containing text in catalog record. Length must be divisible by 80. No linefeeds allowed.

madInstObj - a madrigal.metadata.MadrigalInstrument object. If None, one will be created. Used to verify kinst.

Outputs: None

Returns: None

__str__

```
__str__ ( self )
```

returns a string representation of a MadrigalCatalogRecord

getEndTimeList

```
getEndTimeList ( self )
```

getEndTimeList returns a tuple containing eYear, eMonth, eDay, eHour, eMin, eSec, and eCentisec

Inputs: None

Outputs: a tuple containing eYear, eMonth, eDay, eHour, eMin, eSec, and eCentisec.

getKinst

```
getKinst ( self )
```

getKinst returns the kind of instrument code (int) for a given catalog record.

Inputs: None

Outputs: the kind of instrument code (int) for a given catalog record.

getModexp

```
getModexp ( self )
```

getModexp returns the mode of experiment code (int) for a given catalog record.

Inputs: None

Outputs: the mode of experiment code (int) for a given catalog record.

getStartTimeList

```
getStartTimeList ( self )
```

getStartTimeList returns a tuple containing sYear, sMonth, sDay, sHour, sMin, sSec, and sCentisec

Inputs: None

Outputs: a tuple containing sYear, sMonth, sDay, sHour, sMin, sSec, and sCentisec.

getText

```
getText ( self )
```

getText returns the catalog text.

Inputs: None

Outputs: the catalog text.

getType

```
getType ( self )
```

returns the type `catalog`

setKinst

```
setKinst ( self,  kinst )
```

setKinst sets the kind of instrument code (int) for a given catalog record.

Inputs: kind of instrument code (integer)

Outputs: None

Affects: sets the kind of instrument code (int) (self.__kinst) for a given catalog record. Prints warning if kinst not found in instTab.txt

setModexp

```
setModexp ( self,  modexp )
```

setModexp sets the mode of experiment code (int) for a given catalog record.

Inputs: the mode of experiment code (int)

Outputs: None

Affects: sets the mode of experiment code (int) (self.__modexp)

setText

```
setText ( self,  text )
```

setText sets the catalog text.

Inputs: text: text to be set. Must be length divisible by 80, and not contain line feeds. Also, for now cannot exceed 2^(16) - 80

Outputs: None.

Affects: sets self.__text

Raise TypeError if problem with test

Exceptions

TypeError, 'text exceeds ability of Cedar format to store'
TypeError, 'text length must be divisible by 80: len is %i' %(len(text))
TypeError, 'text must be of type string'
TypeError, 'text must not contain linefeed character'

setTimeLists

```
setTimeLists (
        self,
        sYear,
        sMonth,
        sDay,
        sHour,
        sMin,
        sSec,
        sCentisec,
        eYear,
        eMonth,
        eDay,
        eHour,
        eMin,
        eSec,
        eCentisec,
        )
```

setTimeList resets start and end times

Inputs:

sYear,sMonth,sDay,sHour,sMin,sSec,sCentisec - experiment start time. sCentisec must be 0-99

eYear,eMonth,eDay,eHour,eMin,eSec,eCentisec - experiment end time. eCentisec must be 0-99

Outputs: None

Affects: sets all time attributes (see code).

Exceptions: Raises ValueError if startTime > endTime

Exceptions	
	ValueError, 'Starting time cannot be after ending time'

writeRecord

```
writeRecord (
        self,
        cedarFile,
        format,
        )
```

writeRecord writes a MadrigalCatalogRecord to an open cedarFile.

Users should not call this method directly. Use MadrigalCedarFile.write to persist data.

Inputs:

cedarFile - pointer to madrec as returned by madrigal._Madrec.madrecOpen

format - a format to save the file in. For now, the allowed values are `Madrigal`, `BlockedBinary`, `UnblockedBinary`, `Cbf`, and `Ascii`.

Outputs: None

Affects: writes MadrigalCatalogRecord to cedar file

Exceptions	
	IOError, 'error in madrecCreateCatalogRecord' IOError, 'error in madrecPutNextRec'

Table of Contents

This document was automatically generated on Fri May 30 14:39:57 2008 by HappyDoc version r1_5

Table of Contents

Class: MadrigalCedarFile

MadrigalCedarFile is an object that allows the creation and editting

This class emulates a python list, and so users may treat it just like a python list. The restriction either MadrigalCatalogRecords, MadrigalHeaderRecords, or MadrigalDataRecords (all also de these three classes supports the method getType(), which returns `catalog`, `header`, and `da`

Usage example:

```
# the following example inserts a catalog record at the beginning of

import madrigal.cedar.MadrigalCedarFile, time

cedarObj = madrigal.cedar.MadrigalCedarFile('/opt/madrigal/experimen

startTime = time.mktime((1998,1,20,0,0,0,0,0,0)) - time.timezone

endTime = time.mktime((1998,1,21,23,59,59,0,0,0)) - time.timezone

# catLines is a list of 80 character lines to be included in catalog

catObj = madrigal.cedar.MadrigalCatalogRecord(31, 1000, 1998,1,20,0,
                                              1998,1,21,23,59,59,99,

cedarObj.insert(0, catObj)

cedarObj.write()
```

Non-standard Python modules used: None

Change history:

Written by Bill Rideout April. 6, 2005

Methods

	__contains__	__setslice__
	__delitem__	__str__

__contains__

```
__contains__ ( self,  other )
```

__delitem__

```
__delitem__ ( self,  key )
```

__delslice__

```
__delslice__ (
        self,
        i,
        j,
        )
```

__getitem__

```
__getitem__ ( self,  key )
```

__getslice__

```
__getslice__ (
        self,
        i,
        j,
        )
```

__init__

```
__init__ (
        self,
        fullFilename,
        createFlag=False,
        startDatetime=None,
        endDatetime=None,
        )
```

__init__ initializes MadrigalCedarFile by readir

Inputs:

fullFilename - either the existing Cedar file (in any allowed Cedar

createFlag - tells whether this is a file to be created. If False and f
raised. If True and fullFilename already exists, or fullFilename ca

startDatetime - if not None (the default), reject all input records w
(datetime.datetime object)

endDatetime - if not None (the default), reject all input records wh
(datetime.datetime object)

Affects: populates self.__privList if file exists, sets self.__fullFile

Returns: void

Exceptions

ValueError, 'in MadrigalCedarFile, createF
ValueError, 'in MadrigalCedarFile, endDat
endDatetime))
ValueError, 'in MadrigalCedarFile, fullFile
fullFilename))
ValueError, 'in MadrigalCedarFile, fullFile
fullFilename))
ValueError, 'in MadrigalCedarFile, fullFile
fullFilename))
ValueError, 'in MadrigalCedarFile, startDa
startDatetime))

__iter__

```
__iter__ ( self )
```

__len__

```
__len__ ( self )
```

__parseFile

```
__parseFile ( self )
```

__parseFile reads an existing Cedar file, and populates self.__priv
MadrigalHeaderRecords, or MadrigalDataRecords.

Exceptions

'Illegal return value'
IOError, 'Error parsing Cedar file %s' %(s

__setitem__

```
__setitem__ (
        self,
        key,
        value,
        )
```

Exceptions

ValueError, 'In MadrigalCedarFile, can onl
MadrigalHeaderRecord, or MadrigalDataR

__setslice__

```
__setslice__ (
        self,
        i,
        j,
        seq,
        )
```

Exceptions

ValueError, 'In MadrigalCedarFile, can onl
MadrigalHeaderRecord, or MadrigalDataR

__str__

```
__str__ ( self )
```

append

```
append ( self,  item )
```

Exceptions

ValueError, 'In MadrigalCedarFile, can onl
MadrigalHeaderRecord, or MadrigalDataR

count

```
count ( self,  other )
```

createCatalogTimeSection

```
createCatalogTimeSection ( self )
```

createCatalogTimeSection will return all the lines in the catalog re
the data records.

Inputs: None

Returns: a tuple with three items 1) a string in the format of the tir
datetime, 3) latest datetime

createHeaderTimeSection

```
createHeaderTimeSection ( self,  dataRecList=None )
```

createHeaderTimeSection will return all the lines in the header rec
the data records.

Inputs:

dataRecList - if given, examine only those MadrigalDataRecords
examine all MadrigalDataRecords in this MadrigalCedarFile

Returns: a tuple with three items 1) a string in the format of the ti
datetime, 3) latest datetime

dump

```
dump ( self,  format='UnblockedBinary' )
```

dump appends all the present records in Madr
removes present records from MadrigalCedarF

Can be used to append records to a file. Only works with file form
formats do not have clear record -> file mapping.

Inputs:

format - a format to save the file in. For now, the allowed values a
Defaults to UnblockedBinary. If dump was previously called with

Outputs: None

Affects: writes a MadrigalCedarFile to file

Exceptions

ValueError, 'Format must be UnblockedBi
format))
ValueError, 'Previous dump format was %s
), str(format))

index

```
index ( self,  other )
```

insert

```
insert (
        self,
        i,
        x,
```

```
        )
```

pop

```
pop ( self,  i )
```

remove

```
remove ( self,  x )
```

reverse

```
reverse ( self )
```

write

```
write (
        self,
        format='Madrigal',
        newFilename=None,
        )
```

write persists a MadrigalCedarFile to file.

Inputs:

format - a format to save the file in. For now, the allowed values a
`UnblockedBinary`, `Cbf`, and `Ascii`. Defaults to Madrigal.

newFilename - a filename to save to. Defaults to self.__fullFilena

Outputs: None

Affects: writes a MadrigalCedarFile to file

Exceptions

ValueError, 'Cannot call write method afte

Table of Contents

This document was automatically generated on Tue Dec 2 15:12:20 2008 by HappyDoc version r1_5

Table of Contents

Class: MadrigalDataRecord **cedar.py**

MadrigalDataRecord holds all the information in a Cedar data record.

Methods

__get2DIncrValueList__

```
__get2DIncrValueList__ (
        self,
        code,
        scaleFactor,
        )
```

__get2DIncrValueList__ returns a list containing all the additional increment 2D values of a given main parameter.

Inputs:

parm - parameter code (integer). Must be a parameter with an additional increment parameter.

Outputs: a list containing all the additional increment 2D values of a given main parameter. Special values will be given the values `missing`, `assumed`, or `knownbad`

__get2DMainValueList__

```
__get2DMainValueList__ (
        self,
        code,
        scaleFactor,
        )
```

__get2DMainValueList__ returns a list containing all the 2D values of a given main parameter.

Inputs:

code - parameter code (integer). Must be a parameter with an additional increment parameter.

Outputs: a list containing all the 2D values of a given main parameter that has an additional increment parameter. Special values will be given the values `missing`, `assumed`, or `knownbad`

__get2DValueList__

```
__get2DValueList__ ( self,  parm )
```

__get2DValueList__ returns a list containing all the 2D values of a given parameter.

Inputs:

parm - can be defined as code (integer) or case-insensitive mnemonic string (eg, "Gdalt")

Outputs: a list containing all the 2D values of a given parameter. Special values will be given the values `missing`, `assumed`, or `knownbad`

__init__

```
__init__ (
        self,
        kinst,
        kindat,
        sYear,
        sMonth,
        sDay,
        sHour,
        sMin,
        sSec,
        sCentisec,
        eYear,
        eMonth,
        eDay,
        eHour,
        eMin,
        eSec,
        eCentisec,
        oneDList,
        twoDList,
        nrow,
        madInstObj=None,
        madParmObj=None,
        )
```

__init__ creates a MadrigalDataRecord with all missing data.

Inputs:

kinst - the kind of instrument code. A warning will be raised if not in instTab.txt.

kindat - kind of data code. Must be a non-negative integer.

sYear,sMonth,sDay,sHour,sMin,sSec,sCentisec - record start time. sCentisec must be 0-99

eYear,eMonth,eDay,eHour,eMin,eSec,eCentisec - record end time. eCentisec must be 0-99

oneDList - list of one-dimensional parameters in record. Parameters can be defined as codes (integers) or case-insensitive mnemonic strings (eg, "Gdalt")

twoDList - list of two-dimensional parameters in record. Parameters can be defined as codes (integers) or case-insensitive mnemonic strings (eg, "Gdalt")

nrow - number of rows of 2D data to create. Until set, all values default to missing.

madInstObj - a madrigal.metadata.MadrigalInstrument object. If None, one will be created. Used to verify kinst.

madParmObj - a madrigal.data.MadrigalParameter object. If None, one will be created. Used to verify convert parameters to codes.

Outputs: None

Returns: None

Exceptions

'kindat cannot be negative: %i' %(kindat)
ValueError, 'nrow must not be less that zero: = %i' %(nrow)

__str__

```
__str__ ( self )
```

returns a string representation of a MadrigalDataRecord

add1D

```
add1D ( self,  oneDParm )
```

add1D adds a new one-dim parameter to a MadrigalDataRecord

Input: oneDParm - Parameter can be defined as codes (integer) or case-insensitive mnemonic string (eg, "Gdalt")

Affects: adds tuple to self.__oneDList, where tuple has five elements: (code, lower-case mnemonic, parameter description, scaleFactor, hasAddIncrement)

Also adds one item to self.__oneDData, with value = missing. Creates self.__oneDData if does not exist.

Exceptions

ValueError, 'Illegal use of an "additional increment" Cedar parameter %s, since this module sets values via doubles, not ints' %(mnem)

add2D

```
add2D ( self,  twoDParm )
```

add2D adds a new two-dim parameter to a MadrigalDataRecord

Input: twoDParm - Parameter can be defined as codes (integer) or case-insensitive mnemonic string (eg, "Gdalt")

Affects: adds tuple to self.__twoDList, where tuple has five elements: (code, lower-case mnemonic, parameter description, scaleFactor, hasAddIncrement)

Also adds one column with self.__nrow rows to self.__twoDData, with value = missing. Creates self.__twoDData if does not exist.

Exceptions

ValueError, 'Illegal use of an "additional increment" Cedar parameter %s, since this module sets values via doubles, not ints' %(mnem)

delete1D

```
delete1D ( self,  parm )
```

delete1D removes the given 1D parameter from the record

Inputs:

parm - can be defined as code (integer) or case-insensitive mnemonic string (eg, "Gdalt")

Outputs: None

Raise exception if 1D parm does not exist

Exceptions

ValueError, 'Parameter %s not found as 1D parameter in this data record' %(str(parm))

delete2DParm

```
delete2DParm ( self,  parm )
```

delete2DParm removes the given 2D parameter from every row in the record

Inputs:

parm - can be defined as code (integer) or case-insensitive mnemonic string (eg, "Gdalt")

Outputs: None

Raise exception if 2D parm does not exist

Exceptions

ValueError, '2D Parameter %s not found in this data record' %(str(parm))

delete2DRows

```
delete2DRows ( self,  rows )
```

delete2DRows removes the given 2D row or rows in the record (first is row 0)

Inputs:

row number (integer) or list of row numbers to delete (first is row 0)

Outputs: None

Raise exception if row does not exist

Exceptions

ValueError, 'Illegal value of row %i with nrow = %i' %(thisRow, self.__nrow)

get1D

```
get1D ( self,  parm )
```

get1D returns the 1D value for a given 1D parameter

Inputs:

parm - can be defined as code (integer) or case-insensitive mnemonic string (eg, "Gdalt")

Outputs: double value, or the strings "missing", "assumed", or "knownbad"

Exceptions

ValueError, '1D Parameter %s not found in this data record' %(str(parm))

get1DParms

```
get1DParms ( self )
```

get1DParms returns a list of 1D parameters in the MadrigalDataRecord.

Inputs: None

Outputs: a list of 1D CedarParameter objects in the MadrigalDataRecord.

get2D

```
get2D (
        self,
        parm,
        row,
        )
```

get2D returns the 2D value for a given 2D parameter

Inputs:

parm - can be defined as code (integer) or case-insensitive mnemonic string (eg, "Gdalt")

row - row number to get data. Starts at 0.

Outputs: double value, or the strings "missing", "assumed", or "knownbad"

Exceptions

ValueError, '2D Parameter %s not found in this data record' %(str(parm))
ValueError, 'Illegal value of row %i with nrow = %i' %(row, self.__nrow)

get2DParms

```
get2DParms ( self )
```

get2DParms returns a list of 2D parameters in the MadrigalDataRecord.

Inputs: None

Outputs: a list of 2D CedarParameter objects in the MadrigalDataRecord.

getEndTimeList

```
getEndTimeList ( self )
```

getEndTimeList returns a tuple containing eYear, eMonth, eDay, eHour, eMin, eSec, and eCentisec

Inputs: None

Outputs: a tuple containing eYear, eMonth, eDay, eHour, eMin, eSec, and eCentisec.

getHeaderKodLines

```
getHeaderKodLines ( self )
```

getHeaderKodLines creates the lines in the Madrigal header record that start KOD and describe parms

Inputs: None

Returns: a string of length 80*num 1D parms. Each 80 characters contains a description of a single parm accodring to the Cedar Standard

getKindat

```
getKindat ( self )
```

getKindat returns the kind of data code (int) for a given data record.

Inputs: None

Outputs: the kind of data code (int) for a given data record.

getKinst

```
getKinst ( self )
```

getKinst returns the kind of instrument code (int) for a given data record.

Inputs: None

Outputs: the kind of instrument code (int) for a given data record.

getNrow

```
getNrow ( self )
```

getNrow returns the number of 2D data rows (int) for a given data record.

Inputs: None

Outputs: the number of 2D data rows.

getStartTimeList

```
getStartTimeList ( self )
```

getStartTimeList returns a tuple containing sYear, sMonth, sDay, sHour, sMin, sSec, and sCentisec

Inputs: None

Outputs: a tuple containing sYear, sMonth, sDay, sHour, sMin, sSec, and sCentisec.

getType

```
getType ( self )
```

returns the type `data`

set1D

```
set1D (
        self,
        parm,
        value,
        )
```

set1D sets a 1D value for a given 1D parameter

Inputs:

parm - can be defined as code (integer) or case-insensitive mnemonic string (eg, "Gdalt")

value - double (or string convertable to double) value to set 1D parameter to. To set special Cedar values, the global values missing, assumed, or knownbad may be used, or the strings "missing", "assumed", or "knownbad"

Outputs: None

Note: if value exceeds the dynamic range of given parameter (ie, if value < -32766*scaleFactor or value > 32767*scaleFactor, then WARNING printed to stderr and value set to missing

Exceptions	
	ValueError, 'Cannot set a Madrigal parameter to %s' %(str(value)) ValueError, 'Parameter %s not found as 1D parameter in this data record' %(str(parm))

set2D

```
set2D (
        self,
        parm,
        row,
        value,
        )
```

set2D sets a 2D value for a given 2D parameter and row

Inputs:

parm - can be defined as code (integer) or case-insensitive mnemonic string (eg, "Gdalt")

row - row number to set data. Starts at 0.

value - double (or string convertable to double) value to set 2D parameter to. To set special Cedar values, the global values missing, assumed, or knownbad may be used, or the strings "missing", "assumed", or "knownbad"

Outputs: None

Note: if value exceeds the dynamic range of given parameter (ie, if value < -32766*scaleFactor or value > 32767*scaleFactor, then WARNING printed to stderr and value set to missing

Exceptions

ValueError, 'Cannot set a Madrigal parameter to %s' %(str(value))
ValueError, 'Illegal value of row %i with nrow = %i' %(row, self.__nrow)
ValueError, 'Parameter %s not found as 2D parameter in this data record' %(str(parm))

setEndTimeList

```
setEndTimeList (
        self,
        eYear,
        eMonth,
        eDay,
        eHour,
        eMin,
        eSec,
        eCentisec=0,
        )
```

setEndTimeList changes the data record end time

Inputs: integers eYear, eMonth, eDay, eHour, eMin, eSec. eCentisec defaults to 0

Outputs: None

Affects: changes self.__eYear, self.__eMonth, self.__eDay, self.__eHour, self.__eMin, self.__eSec, and self.__eCentisec. Also self.__eTime

Prints warning if new start time after present end time

Exceptions

ValueError, 'Illegal datetime %s' %(str((eYear, eMonth, eDay, eHour, eMin, eSec)))
ValueError, 'Illegal eCentisec %i' %(eCentisec)

setKindat

```
setKindat ( self,  newKindat )
```

setKindat sets the kind of data code (int) for a given data record.

Inputs: newKindat (integer)

Outputs: None

Affects: sets self.__kindat

Exceptions

'kindat cannot be negative: %i' %(kindat)

setKinst

```
setKinst ( self,  newKinst )
```

setKinst sets the kind of instrument code (int) for a given data record.

Inputs: newKinst - new instrument code (integer)

Outputs: None

Affects: sets self.__kinst

Exceptions

ValueError, 'Kinst must not be less than 0, not %i' %(newKinst)

setStartTimeList

```
setStartTimeList (
        self,
        sYear,
        sMonth,
        sDay,
        sHour,
        sMin,
        sSec,
        sCentisec=0,
        )
```

setStartTimeList changes the data record start time

Inputs: integers sYear, sMonth, sDay, sHour, sMin, sSec. sCentisec defaults to 0

Outputs: None

Affects: changes self.__sYear, self.__sMonth, self.__sDay, self.__sHour, self.__sMin, self.__sSec, and self.__sCentisec. Also self.__sTime

Prints warning if new start time after present end time

Exceptions

ValueError, 'Illegal datetime %s' %(str((sYear, sMonth, sDay, sHour, sMin, sSec)))
ValueError, 'Illegal sCentisec %i' %(sCentisec)

writeRecord

```
writeRecord (
        self,
        cedarFile,
        format,
        )
```

writeRecord writes a MadrigalDataRecord to an open cedarFile.

Users should not call this method directly. Use MadrigalCedarFile.write to persist data.

Inputs:

cedarFile - pointer to madrec as returned by madrigal._Madrec.madrecOpen

format - a format to save the file in. For now, the allowed values are `Madrigal`, `BlockedBinary`, `UnblockedBinary`, `Cbf`, and `Ascii`.

Outputs: None

Affects: writes MadrigalDataRecord to cedar file

Exceptions

IOError, 'Tried to set 1D error parm %s to zero' %(thisParm [1])
IOError, 'Tried to set 2D error parm %s to zero' %(thisParm [0])
IOError, 'error in cedarSet1dParm'

Table of Contents

This document was automatically generated on Thu Oct 20 16:51:50 2011 by HappyDoc version r1_5

Table of Contents

Class: MadrigalHeaderRecord — **cedar.py**

MadrigalHeaderRecord holds all the information in a Cedar header record.

Methods

__init__
__str__
getEndTimeList
getJpar
getKindat
getKinst
getMpar
getStartTimeList
getText
setJpar
setKindat
setKinst
setMpar
setText
setTimeLists
writeRecord

getType

__init__

```
__init__ (
        self,
        kinst,
        kindat,
        sYear,
        sMonth,
        sDay,
        sHour,
        sMin,
        sSec,
        sCentisec,
        eYear,
        eMonth,
        eDay,
        eHour,
        eMin,
        eSec,
        eCentisec,
        jpar,
        mpar,
        text,
        madInstObj=None,
        )
```

__init__ creates a MadrigalCatalogRecord.

Inputs:

kinst - the kind of instrument code. A warning will be raised if not in instTab.txt.

kindat - kind of data code. Must be a non-negative integer.

sYear,sMonth,sDay,sHour,sMin,sSec,sCentisec - experiment start time. sCentisec must be 0-99

eYear,eMonth,eDay,eHour,eMin,eSec,eCentisec - experiment end time. eCentisec must be 0-99

jpar - the number of 1d parameters in the following data records

mpar - the number of 2d parameters in the following data records

text - string containing text in catalog record. Length must be divisible by 80. No linefeeds allowed.

madInstObj - a madrigal.metadata.MadrigalInstrument object. If None, one will be created. Used to verify kinst.

Outputs: None

Returns: None

__str__

```
__str__ ( self )
```

returns a string representation of a MadrigalHeaderRecord

getEndTimeList

```
getEndTimeList ( self )
```

getEndTimeList returns a tuple containing eYear, eMonth, eDay, eHour, eMin, eSec, and eCentisec

Inputs: None

Outputs: a tuple containing eYear, eMonth, eDay, eHour, eMin, eSec, and eCentisec.

getJpar

```
getJpar ( self )
```

returns the number of one-dimensional parameters in the associated data records.

getKindat

```
getKindat ( self )
```

getKindat returns the kind of data code (int) for a given header record.

Inputs: None

Outputs: the kind of data code (int) for a given header record.

getKinst

```
getKinst ( self )
```

getKinst returns the kind of instrument code (int) for a given header record.

Inputs: None

Outputs: the kind of instrument code (int) for a given header record.

getMpar

```
getMpar ( self )
```

returns the number of two-dimensional parameters in the associated data records.

getStartTimeList

```
getStartTimeList ( self )
```

getStartTimeList returns a tuple containing sYear, sMonth, sDay, sHour, sMin, sSec, and sCentisec

Inputs: None

Outputs: a tuple containing sYear, sMonth, sDay, sHour, sMin, sSec, and sCentisec.

getText

```
getText ( self )
```

getText returns the header text.

Inputs: None

Outputs: the header text.

getType

```
getType ( self )
```

returns the type `header`

setJpar

```
setJpar ( self,  jpar )
```

set the number of one-dimensional parameters in the associated data records.

Must not be negative.

Exceptions

TypeError, 'jpar must not be less than 0'

setKindat

```
setKindat ( self,  kindat )
```

setKindat sets the mode of kind of data code (int) for a given header record.

Inputs: the kind of data code (int)

Outputs: None

Affects: sets the kind of data code (int) (self.__kindat)

Exceptions: Raises ValueError if kindat less than 0

Exceptions

ValueError, 'kindat must not be less than 0, not %i' %(self.__kindat)

setKinst

```
setKinst ( self,  kinst )
```

setKinst sets the kind of instrument code (int) for a given header record.

Inputs: kind of instrument code (integer)

Outputs: None

Affects: sets the kind of instrument code (int) (self.__kinst) for a given header record. Prints warning if kinst not found in instTab.txt

setMpar

```
setMpar ( self,  mpar )
```

set the number of two-dimensional parameters in the associated data records.

Must not be negative.

Exceptions

TypeError, 'mpar must not be less than 0'

setText

```
setText ( self,  text )
```

setText sets the header text.

Inputs: text: text to be set. Must be length divisible by 80, and not contain line feeds. For now, must not exceed 2^16 - 80 bytes to be able to be handled by Cedar format.

Outputs: None.

Affects: sets self.__text

Raises TypeError if problem with text

Exceptions

TypeError, 'text exceeds ability of Cedar format to store'
TypeError, 'text length must be divisible by 80: len is %i' %(len(text))
TypeError, 'text must be of type string'
TypeError, 'text must not contain linefeed character'

setTimeLists

```
setTimeLists (
        self,
        sYear,
        sMonth,
        sDay,
        sHour,
        sMin,
        sSec,
        sCentisec,
        eYear,
        eMonth,
        eDay,
        eHour,
        eMin,
        eSec,
```

```
        eCentisec,
        )
```

setTimeList resets start and end times

Inputs:

sYear,sMonth,sDay,sHour,sMin,sSec,sCentisec - experiment start time. sCentisec must be 0-99

eYear,eMonth,eDay,eHour,eMin,eSec,eCentisec - experiment end time. eCentisec must be 0-99

Outputs: None

Affects: sets all time attributes (see code).

Exceptions: Raises ValueError if startTime > endTime

Exceptions	
	ValueError, 'Starting time cannot be after ending time'

writeRecord

```
writeRecord (
        self,
        cedarFile,
        format,
        )
```

writeRecord writes a MadrigalHeaderRecord to an open cedarFile.

Users should not call this method directly. Use MadrigalCedarFile.write to persist data.

Inputs:

cedarFile - pointer to madrec as returned by madrigal._Madrec.madrecOpen

format - a format to save the file in. For now, the allowed values are `Madrigal`, `BlockedBinary`, `UnblockedBinary`, `Cbf`, and `Ascii`.

Outputs: None

Affects: writes MadrigalHeaderRecord to cedar file

Exceptions

IOError, 'error in madrecCreateHeaderRecord'
IOError, 'error in madrecPutNextRec'

Table of Contents

This document was automatically generated on Fri May 30 14:39:57 2008 by HappyDoc version r1_5

Table of Contents

Module: data **data.py**

data is the module that interfaces to madrigal data files, or to Cedar standards about data.

This module includes the api to get information from a single madrigal file, and the api to get information about the Cedar standard (such as parameter and category definitions). It is built on top of the Madrigal C API as found in libmadrec.so through the python extension class __Madrec

$Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $

Functions

_mnemSorter

_mnemSorter

```
_mnemSorter ( mnem1,  mnem2 )
```

_mnemSorter is a helper function that compares two parameter mnemonics based on their order in parcods.tab. Error parameters always directly follow the standard parameter

Classes

MadrigalFile	MadrigalFile is an object that provides access to information in a single Madrigal File.
MadrigalParameters	MadrigalParameters is an object that provides information about Madrigal parameters.

Table of Contents

This document was automatically generated on Thu Oct 20 16:51:50 2011 by HappyDoc version r1_5

Table of Contents

Class: MadrigalFile

MadrigalFile is an object that provides access to information in a single

This object provides access to a single Madrigal file.

Usage example:

```
import os, madrigal.data

filepath = os.environ.get('MAD' + 'ROOT') + '/experiments/1998/mlh/20jan9

test = madrigal.data.MadrigalFile(filepath)

print test.toString()

print test.getMaxValidAltitude()
```

Non-standard Python modules used:

None

MadrigalError exception thrown if:

1. No data records found in file

Notes: This class depends on the python extension class madrigal._Madrec, which in turn depends on implemented in libmadrec.so. When Madrigal is installed, this file is put in $madroot/lib. If madroot i located by python by setting the LD_LIBRARY_PATH to the new location of libmadrec.so.

Change history:

Written by Bill Rideout Nov. 26, 2001

Modified by Bill Rideout June 4, 2002 to use summary information in header records if available.

Modified by Bill Rideout Jan 24, 2003 to use the high level maddata module.

Modified by Bill Rideout Jan 24, 2003 to use overview/[filename].summary file if available.

Modified by Bill Rideout Feb 15, 2005 to add more summary data.

Modified by Miguel Urco May 04, 2011 to handle hdf5 files.

Methods

__getExistingSummary	getCachedHdf5	getMa
__getLayoutDescription	getCatalogHeaderStr	getMe
__init__	getEarliestTime	getMe
__parseSummary	getKindatList	getMe
__setToOne	getKinstList	getMe

__getExistingSummary

```
__getExistingSummary ( self )
```

__getExistingSummary returns a list of strings summarizing a or overview file if possible.

If all the required information is not found in the first two records of the file, the file ov that fails, returns None.

Inputs: None

Returns: A list of strings summarizing the MadrigalFile. These strings are: kinstList: a s kindatList: a string of comma separated integers parameterList: a string of comma separ comma separated integers maxPulseLen: a single integer string minPulseLen: a single i integer string minValidAltitude: a single integer string maxLatitude: a single integer str maxLongitude: a single integer string minLongitude: a single integer string earliestTim integers: [year, month, day, hour, min, sec] latestTime: a string of six comma sepatated sec] param1dList: a string of comma separated integers param2dList: a string of comma

If all required information not found in either source, returns None. The following item missingVal or empty: minPulseLen, minValidAltitude, maxLatitude, minLatitude, max param2dList

Affects: Nothing

Exceptions: None

__getLayoutDescription

```
__getLayoutDescription ( self,  format )
```

__getLayoutDescription returns a description of the layout se

Inputs: format: Alternates format available

Returns: LayoutDescription: A list of strings summarizing the Layout Description

Affects: Nothing

Exceptions: None

__init__

```
__init__ (
        self,
        initFile,
        madDB=None,
        )
```

__init__ initializes MadrigalFile by finding all summary data.

Inputs: self, String representing the full path to the madrigal file.

Existing MadrigalDB object, by default = None.

Returns: void

Affects: Initializes self.__summary, which is a list of summary data about the file. The
header records, and if not found, then the file overview/[filename].summary is used. If
_Madrec.getSummary, which will write its results to overview/[filename].summary. Al
summarized data.

Exceptions: MadrigalError thrown if no data record in file.

__parseSummary

```
__parseSummary ( self,  summaryStr )
```

__parseSummary returns a list of strings summarizing a strin

If all the required information is not found in the string, returns None.

Inputs: summaryStr read from header records or summary file.

Returns: A list of strings summarizing the MadrigalFile. These strings are: kinstList: a s
kindatList: a string of comma separated integers parameterList: a string of comma separ
comma separated integers maxPulseLen: a single integer string minPulseLen: a single i
integer string minValidAltitude: a single integer string maxLatitude: a single integer str
maxLongitude: a single integer string minLongitude: a single integer string earliestTim
integers: [year, month, day, hour, min, sec] latestTime: a string of six comma sepatated
sec] param1dList: a string of comma separated integers param2dList: a string of comma

If all required information not found in either source, returns None. The following item
missingVal or empty list: minPulseLen, minValidAltitude, maxLatitude, minLatitude, m
param1dList, param2dList

Affects: Nothing

Exceptions: None

__setToOne

```
__setToOne ( self,  x )
```

Private function used to initialize a list to ones

__str__

```
__str__ ( self )
```

__writeSummary

```
__writeSummary ( self )
```

__writeSummary writes a summary file to overview/[filename]

Uses data from self.__summary.

Inputs: None.

Returns: None

Affects: writes a summary file to overview/[filename].summary

Exceptions: If file cannot be written

_parseCachedIni

```
_parseCachedIni (
        self,
        kinst,
        kindat,
        iniFile=None,
        )
```

_parseCachedIni parses an ini file for information needed to c

Inputs:

kinst - the instrument kinst (integer)

kindat - the data kindat (integer)

iniFile - the ini file to use. If None, uses default ini file $MADROOT/cachedFiles.ini

Returns: a tuple with two items: 1. a list of extra parameters (string mnemonics) 2. a dic
exportToHdf)

Algorithm:

1. If iniFile == None and no default file, returns ([], {}) 2. Searches ini file for sec
   returns ([], {}) 3. Searches right section for key %i_parms % (kindat). If not fou
   found, extra parameters are [] 4. Searches right section for key %i_formats % (
   default_formats. If not found, alternate format dictionary is {}

basicIsprint

```
basicIsprint (
        self,
        output=None,
        parmLine=True,
        extraParameters=[],
        )
```

basicIsprint writes the file in isprint format, using all the parameters in the file, and the year,month,day,hour,min,sec. No filtering used.

Inputs:

output - file to write isprint output too. If None (the default), write to stdout

parmLine - if True (the default), add text line with parameter names as first line. If Fals

extraParameters - These parameters will be added to the output file if they are not alread

exportToHdf

```
exportToHdf (
        self,
        output=None,
        alternateFormatDict={},
        extraParameters=[],
        filter=None,
        showWarnings=False,
        )
```

exportToHdf will write the Cedar file self.__filename to a hdf5 export format. That hdf listed in alternateFormatDict. For now the available alternate formats are {'array':'range

Inputs:

output - hdf5 file to write. If None (the default), write to self.__filename + `hdf5`

alternateFormatDict - For this dictionary, the keys are always a format name, such as `a` additional arguments might be required, such as `range`

Available alternate formats are:

Array: {'array':<independent parm>} where <independent parm> is independent parm s

extraParameters - These parameters will be added to the output file if they are not alread

filter - Filter argument as in isprint command as string (eg, `ti.500,2000`). Just one f

showWarnings - if True, print warnings about problems to stdout. If False (the default),

Affects: Nothing

Exceptions: Parameter required is not in the file nor can it be derived. No records select

Exceptions

madrigal.admin.MadrigalError('The independent parameter
be derived' % parmMnem.upper(), None)
madrigal.admin.MadrigalError('isprint error: ' + str(line), N

getCachedHdf5

```
getCachedHdf5 (
        self,
        iniFile=None,
        overwrite=False,
        showWarnings=False,
        )
```

getCachedHdf5 will get the full path to the cached version of the HDF5 file. It will crea

Inputs: iniFile - ini file to use to create hdf5 with, if one needs to be created. If None, us
$MADROOT/cachedFiles.ini overwrite - if True, overwrite existing hdf5 file. If False (
showWarnings - if True, print warnings about problems to stdout. If False (the default),
Hdf5 file

getCatalogHeaderStr

```
getCatalogHeaderStr ( self )
```

getCatalogHeaderStr returns a string formatted for printing co header records.

Input: None

Returns: a string formatted for printing containing all catalog and header records. Retur

getEarliestTime

```
getEarliestTime ( self )
```

getEarliestTime returns a list of 6 numbers representing the e

Inputs: self

Returns: a list of 6 numbers representing the earliest time in the file. The format is [Yea

Affects: Nothing

Exceptions: None

getKindatList

```
getKindatList ( self )
```

getKindatList returns a list of integers of all kindat values in fi

Inputs: self

Returns: a list of integers of all kindat values in file.

Affects: Nothing

Exceptions: None

getKinstList

```
getKinstList ( self )
```

getKinstList returns a list of integers of all kinst values in file.

Inputs: self

Returns: a list of integers of all kinst values in file.

Affects: Nothing

Exceptions: None

getKinstListStr

```
getKinstListStr ( self )
```

getKinstListStr returns a comma-separated string with the na

Inputs: self

Returns: a comma-separated string with the names of kinst values in file.

Affects: Nothing

Exceptions: None

getLatestTime

```
getLatestTime ( self )
```

getLatestTime returns a list of 6 numbers representing the lat

Inputs: self

Returns: a list of 6 numbers representing the latest time in the file. The format is [Year,

Affects: Nothing

Exceptions: None

getMaxLatitude

```
getMaxLatitude ( self )
```

getMaxLatitude returns a double representing maximum latitu

Inputs: self

Returns: a double representing maximum latitude in degrees in file.

Affects: Nothing

Exceptions: None

getMaxLongitude

```
getMaxLongitude ( self )
```

getMaxLongitude returns a double representing maximum lon

Inputs: self

Returns: a double representing maximum longitude in degrees in file.

Affects: Nothing

Exceptions: None

getMaxPulseLength

```
getMaxPulseLength ( self )
```

getMaxPulseLength returns a double representing maximum microseconds in file.

Inputs: self

Returns: a double representing maximum pulse length in seconds in file.

Affects: Nothing

Exceptions: None

getMaxValidAltitude

```
getMaxValidAltitude ( self )
```

getMaxValidAltitude returns a double representing maximum

Inputs: self

Returns: a double representing maximum valid altitude in km in file.

Affects: Nothing

Exceptions: None

getMeasDervBothParmLists

```
getMeasDervBothParmLists (
        self,
        parmList,
        measParmList,
        derivedParmList,
        allParmList,
        sureParmList,
        )
```

getMeasDervBothParmLists sets up four lists: measured parm and sure parms given a parm list to verify.

Inputs: parmList: A list of parameters (integers or mnemonics to be consi

measParmList: an empty python list. Will be filled with an list of all measured paramet function returns.

derivedParmList: an empty python list. Will be filled with an list of all parameters (mne derived from file when function returns.

allParmList: an empty python list. Will be filled with an list of all parameters in measPa function returns.

sureParmList: an empty python list. Will be filled with an list of all parameters from the and parameters that can be derived from those. These parameters can then be derived fo the value of the parameter in the record may be "missing").

Returns: void (see Affects below)

Affects: adds items to measParmList, derivedParmList, and allParmList. All items will

Exceptions: None

Usage example:

```
import os, madrigal.data

filepath = os.environ.get('MAD' + 'ROOT') + '/experiments/

test = madrigal.data.MadrigalFile(filepath)

measParmList = []

derivedParmList = []

allParmList = []

sureParmList = []

test.getMeasDervBothParmLists(madrigal.ui.web.MadrigalWebF
                              measParmList,
                              derivedParmList,
                              allParmList,
                              sureParmList)

#print lists

print 'Measured parms are: ' + str(measParmList)

print 'Derived parms are: ' + str(derivedParmList)

print 'All good parms are: ' + str(allParmList)

print 'Parameters sure to exist are: ' + str(sureParmList)
```

getMeasured1dParmList

```
getMeasured1dParmList ( self )
```

getMeasured1dParmList returns a list of integers of all 1d par

Inputs: self

Returns: a list of integers of all 1d parameters found in file.

Affects: Nothing

Exceptions: None

getMeasured2dParmList

```
getMeasured2dParmList ( self )
```

getMeasured2dParmList returns a list of integers of all 2d par

Inputs: self

Returns: a list of integers of all 2d parameters found in file.

Affects: Nothing

Exceptions: None

getMeasuredParmList

```
getMeasuredParmList ( self )
```

getMeasuredParmList returns a list of integers of all paramete

Inputs: self

Returns: a list of integers of all parameters found in file.

Affects: Nothing

Exceptions: None

getMinLatitude

```
getMinLatitude ( self )
```

getMinLatitude returns a double representing minimum latitu

Inputs: self

Returns: a double representing minimum latitude in degrees in file.

Affects: Nothing

Exceptions: None

getMinLongitude

```
getMinLongitude ( self )
```

getMinLongitude returns a double representing minimum lon

Inputs: self

Returns: a double representing minimum longitude in degrees in file.

Affects: Nothing

Exceptions: None

getMinPulseLength

```
getMinPulseLength ( self )
```

getMinPulseLength returns a double representing minimum p microseconds in file.

Inputs: self

Returns: a double representing minimum pulse length in seconds in file.

Affects: Nothing

Exceptions: None

getMinValidAltitude

```
getMinValidAltitude ( self )
```

getMinValidAltitude returns a double representing minimum v

Inputs: self

Returns: a double representing minimum valid altitude in km in file.

Affects: Nothing

Exceptions: None

getMissingParmList

```
getMissingParmList ( self )
```

getMissingParmList returns a list of integers, one for each pa

Inputs: self

Returns: a list of integers, one for each parameters stored in file. If 1, that parameter wa record in the file. If -1, not missing in any data record.

Affects: Nothing

Exceptions: None

toString

```
toString ( self )
```

toString returns a simple string representation of a MadrigalF

Inputs: None

Returns: String describing a simple representation of a MadrigalFile object.

Affects: Nothing

Exceptions: None

Table of Contents

This document was automatically generated on Thu Oct 20 16:51:50 2011 by HappyDoc version r1_5

Table of Contents

Class: MadrigalParameters **data.py**

MadrigalParameters is an object that provides information about Madrigal parameters.

This class provides access to the Cedar/Madrigal standards for parameters (such as getMnemonic, getDescription, getCodeFromMnemonic) and categories. It will also examine an expression (string) and return the parameter mnemonics it contains.

Usage example:

```
import madrigal.data.MadrigalParameters

test = madrigal.data.MadrigalParameters()

parcode = test.getParmCodeFromMnemonic(&amp;quot;YEAR&amp;quot;)

print parcode
```

Non-standard Python modules used: None

Change history:

Written by Bill Rideout Nov. 27, 2001 Added getMnemonicListFromExpression Jul. 16, 2002

Methods

__init__	getParmDescription	hasAddIncre
__reorder	getParmDescriptionList	hasHtmlDesc
__sortList	getParmFormat	isAddIncrem
getCategoryDict	getParmMnemonic	isError
getIsprintHeader	getParmMnemonicList	normalizePa

__init__

```
__init__ ( self,  madDB=None )
```

__init__ initializes MadrigalParameters by getting some basic information from MadrigalDB.

Inputs: Existing MadrigalDB object, by default = None.

Returns: void

Affects: Initializes self.__binDir.

Exceptions: None.

__reorder

```
__reorder (
        self,
        validList,
        sortedParmList,
        parmList,
        )
```

Private function that returns a new valid list sorted in the same parameter order parmList.

__sortList

```
__sortList ( self,  parmList )
```

Private function that returns a new list of parameters sorted as isprint sorts para

getCategoryDict

```
getCategoryDict ( self,  parmList )
```

getCategoryDict returns a python dict with key = category index, item = catego name and ordered parameters Inputs: parmList - the list of parameters (integers mnemonics)

Returns: a python dict, with key = category index. Each item is a list of two ite first item is the category name (string). The second item is a list of parameter mnemonics from parmList belonging in that category. Ordering is alphabetical,

that an error parameter immediately follows its non-error parameter.

Affects: None

Exceptions: none

getIsprintHeader

```
getIsprintHeader ( self,  mnemonicList )
```

getIsprintHeader returns a string with mnemonics as it would appear at the top of isprint.

Inputs: mnemonic: a list of Madrigal mnemonics (string or integer)

Returns: a string with mnemonics as it would appear at the top of isprint

Affects: none

Exceptions: If any mnemonic not found.

getMadCategoryIndex

```
getMadCategoryIndex ( self,  category )
```

getMadCategoryIndex returns the index (order) of a give category.

Inputs: a Madrigal category (string)

Returns: an integer representing the index (order). Returns -32767 if not found.

Affects: none

Exceptions: none

getMnemonicListFromExpression

```
getMnemonicListFromExpression ( self,  expressionStr )
```

getMnemonicListFromExpression returns a list of uniqu cedar mnemonics in a python logical expression.

Inputs: expressionStr - a string containing a valid python logical expression wit mnemonics as variables. Expression can contain any python logical operator (a not, ==, <, >, <=, >=, !=), any operator, any number, and valid python math fur and any variable that is a valid cedar mnemonic. A substring is assumed to be a mnemonic if it begins with a letter, contains only alphanumerics, period, plus, underscore, and is not immediately followed by a open parenthesis. Each poten

cedar mnemonic is verified, and an exception is thrown if it is not valid. The va
of the entire expression is then verified by replacing all the valid cedar mnemor
"1.0" and executing the resulting expression. If any exception besides divide by
or value error occurs, an exception is thrown. Otherwise, the list of cedar mnem
found is returned.

Returns: a list of unique cedar mnemonics (upper case).

Affects: None

Exceptions: Error thrown if any non-valid mnemonic found, or if expression th
exception when run (except divide by zero or value error).

Exceptions

'The expression "' + expressionStr + '" contains an error: ' +
str(sys.exc_info() [1])
'The expression ' + expressionStr + ' contains an illegal mnen
' + thisMnemonic

getParametersForInstruments

```
getParametersForInstruments (
        self,
        instrumentList,
        parmListName='Comprehensive',
        )
```

getParametersForInstruments returns a list of unique Madrigal mnemonics associated with a list of instrumen

This method's purpose is to return a list of parameters appropriate for a user to
from given that a certain list of instruments is under consideration. This method
return a list of all measured parameters found in data files associated with those
instruments, and also all parameters in the parmNameList that can be derived fr
those measured parameters. The passed in parmNameList must be a valid name
parameter list found in the madrigal.ui.web.MadrigalWebFormat class, and def
the "Comprehensive" list of parameters used in the madDataBrowse web page.

Inputs: instrumentList - a python list on instruments as integers (ki values).

parmListName - a name (string) of a list of parameters in the MadrigalWebFor
class. Defaults to "Comprehensive"

Returns: an ordered list of unique Madrigal mnemonics.

Affects: None

Exceptions: None.

getParmCategory

```
getParmCategory ( self,  parm )
```

getParmCategory returns a category (String) given a ce parameter (integer or mnemonic string).

Inputs: a cedar code (integer)

Returns: a category string

Affects: none

Exceptions: none

getParmCodeFromMnemonic

```
getParmCodeFromMnemonic ( self,  mnemonic )
```

getParmCodeFromMnemonic converts a string to the ce code (integer).

Inputs: mnemonic: the cedar mnemonic (string or integer in string form)

Returns: integer (cedar code)

Affects: none

Exceptions: MadrigalError thrown if code not found.

Exceptions	
	madrigal.admin.MadrigalError('Mnemonic: ' + str(mnemon not a legal mnemonic.', None)

getParmDescription

```
getParmDescription ( self,  parm )
```

getParmDescription returns a description including unit and possible links (String) given a parameter (integer or mnemonic).

Inputs: a parameter (integer or mnemonic)

Returns: a description string including units and possible links

Affects: none

Exceptions: none

getParmDescriptionList

```
getParmDescriptionList ( self,  parmList )
```

getParmDescriptionList returns a list of descriptions (St given a list of parameters (integer or mnemonic).

Inputs: a list of parameters (integer or mnemonic)

Returns: a list of descriptions (String) given a list of parameters (integer or mnemonic).

Affects: none

Exceptions: none

getParmFormat

```
getParmFormat ( self,  mnemonic )
```

getParmFormat returns format string from parcods.tab given mnemonic.

Inputs: mnemonic: the cedar mnemonic (string or integer)

Returns: format string from parcods.tab of given mnemonic

Affects: none

Exceptions: If mnemonic not found.

getParmMnemonic

```
getParmMnemonic ( self,  code )
```

getParmMnemonic returns a mnemonic (String) given a parameter (integer or mnemonic).

Inputs: a parameter: integer, an integer in string form, or a mnemonic string

Returns: a mnemonic string. If integer not found, returns integer in string form.

Affects: none

Exceptions: none

getParmMnemonicList

```
getParmMnemonicList ( self,  codeList )
```

getParmMnemonicList returns a list of upper case mnemonics (String) given a list of cedar codes (integer, integer as string, or mnemonic string).

Inputs: a list of cedar codes (integer, integer as string, or mnemonic string)

Returns: a list of upper case mnemonics (String) given a list of cedar codes (int
If illegal value, returns str(code) for that item

Affects: none

Exceptions: none

getParmScaleFactor

```
getParmScaleFactor ( self,  mnemonic )
```

getParmScaleFactor returns scale factor as double of gi mnemonic.

Inputs: mnemonic: the cedar mnemonic (string or integer)

Returns: scale factor as double of given mnemonic

Affects: none

Exceptions: If mnemonic not found.

getParmType

```
getParmType ( self,  mnemonic )
```

getParmType returns 1 if mnemonic is a standard paran 0 if an error parameter, or -1 if not found.

Inputs: mnemonic: the cedar mnemonic (string or integer in string form)

Returns: 1 if mnemonic is a standard parameter, 0 if an error parameter, or -1 if found

Affects: none

Exceptions: If non-string passed in.

getParmUnits

```
getParmUnits ( self,  parm )
```

getParmUnits returns units (String) given a parameter (integer or mnemonic).

Inputs: a parameter (integer or mnemonic)

Returns: units (String)

Affects: none

Exceptions: none

getSimpleParmDescription

```
getSimpleParmDescription ( self,  parm )
```

getSimpleParmDescription returns a description withou units or links (String) given a parameter (integer or mnemonic).

Inputs: a parameter (integer or mnemonic)

Returns: a description string without units or links

Affects: none

Exceptions: none

getStdExpression

```
getStdExpression ( self,  expressionStr )
```

getStdExpression returns an expression in standard for (upper case mnemonic, all else lower case).

Inputs: expressionStr - a string containing a valid python logical expression wit mnemonics as variables. Expression can contain any python logical operator (a not, ==, <, >, <=, >=, !=), any operator, any number, and valid python math fun and any variable that is a valid cedar mnemonic. A substring is assumed to be a mnemonic if it begins with a letter, contains only alphanumerics, period, plus, c underscore, and is not immediately followed by a open parenthesis. Each poten cedar mnemonic is verified, and an exception is thrown if it is not valid.

Returns: an expression (string) in standard form (upper case mnemonic, all else case).

Affects: None

getParmType returns 1 if mnemonic is a standard parameter,0 if an error parameter, or -1 if not fo

Exceptions: Error thrown if any non-valid mnemonic found.

Exceptions	
	'The expression ' + expressionStr + ' contains an illegal mnem ' + thisMnemonic

hasAddIncrement

```
hasAddIncrement ( self,  parm )
```

hasAddIncrement returns True if parm has additional increment parameter, False otherwise

Inputs: a parameter (integer or mnemonic)

Returns: True if parm has additional increment parameter, False otherwise

Affects: none

Exceptions: none

hasHtmlDesc

```
hasHtmlDesc ( self,  parm )
```

hasHtmlDesc returns 1 if that parameter has a html description in parmDesc.html.

Inputs: a Madrigal mnemonic (string) or parameter

Returns: 1 if that parameter has a html description in parmDesc.html, 0 if not

Affects: none

Exceptions: none

isAddIncrement

```
isAddIncrement ( self,  parm )
```

isAddIncrement returns True if parm is an additional increment parameter, False otherwise

Inputs: a parameter (integer or mnemonic)

Returns: True if parm is an additional increment parameter, False otherwise

Affects: none

getStdExpression returns an expression in standard form(upper case mnemonic, all else lower case).

Exceptions: none

isError

```
isError ( self,  parm )
```

isError returns True if parm is an error parameter, False otherwise

Inputs: a parameter (integer or mnemonic)

Returns: True if parm is an error parameter, False otherwise

Affects: none

Exceptions: none

normalizeParmList

```
normalizeParmList ( self,  parmList )
```

normalizeParmList returns an ordered list of parameters all mnemonics changed to integers.

Inputs: parmList - the list of parameters (integers or mnemonics) to convert

Returns: a new parmList that is ordered (negitive values are placed directly afte same positive values) and all parameters are converted to integers

Affects: None

Exceptions: none

orderParms

```
orderParms ( self,  parmList )
```

orderParms sorts mnemonic parameters based on orde parcods.tab file.

Error parameters directly follow standard parameter.

Table of Contents

Table of Contents

Module: metadata	**metadata.py**

The metadata module provides access to all metadata about one particular madrigal database.

This metadata is presently read from the files in the metadata directory, and from the madrigal.cfg file. If that file cannot be found, hard-coded values set during installation are used instead. If the madrigal.cfg file is found at the location specified by either the madroot enviroment variable or at the hard-coded value of madroot set at installation, then the parameters are read from the madrigal.cfg file. Note that madroot with caps is only written once in this file before installation, since it will be automatically replaced, so it is referred to by MAD_ROOT or MAD+ROOT.

$Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $

Classes	
MadrigalDB	MadrigalDB is an object that provides access to an entire Madrigal database.
MadrigalExperiment	MadrigalExperiment is an object that provides access to Madrigal experiment info from the metadata.
MadrigalInstrument	MadrigalInstrument is an object that provides access to Madrigal instrument info from the metadata.
MadrigalInstrumentKindats	MadrigalInstrumentKindats is an object that provides access to the metadata file that summarizes the kindat codes associated with each instrument.
MadrigalInstrumentParameters	MadrigalInstrumentParameters is an object that provides access to the metadata file that summarizes the parameters associated with each instrument.
MadrigalKindat	MadrigalKindat is an object that provides access to Madrigal kind of data info from the metadata.
MadrigalMetaFile	MadrigalMetaFile is an object that provides access to Madrigal file info from the metadata.
MadrigalMetadata	MadrigalMetadata is a private class that parses a Madrigal metadata file.
MadrigalSite	MadrigalSite is an object that provides access to Madrigal site info from the metadata.

Table of Contents

This document was automatically generated on Fri Dec 30 09:02:26 2005 by HappyDoc version r1_5

Table of Contents

Class: MadrigalDB **metadata.py**

MadrigalDB is an object that provides access to an entire Madrigal database.

This object provides complete high-level access to an entire Madrigal database. Presently, all its information comes from madrigal.cfg, or another path passed in by the user. If env variable madroot is not set, or if madrigal.cfg cannot be opened, the default values automatically editted during installation are used

Usage example:

```
import madrigal.metadata

try:

    test =  madrigal.metadata.MadrigalDB()

except madrigal.admin.MadrigalError, e:

    print e.getExceptionStr()

else:

    print test.toString()
```

Non-standard Python modules used: None

MadrigalError exception thrown if:

1. If the constructor is called with an configuration file path as an argument, and that file cannot be read.
2. If the configuration file cannot be parsed by ConfigParser.
3. If any of the following keys cannot be found in the configuration file:

   *__MAD_SERVER

   *__MAD_SERVERROOT

   *__MAD_SERVERCGI

   *__SITE_ID

   *__HTML_STYLE

   *__INDEX_HEAD

*__CON_TACTLINK

*__MAD_SERVERDOCABS

Change history:

Written by Bill Rideout Oct. 4, 2001

Methods

__finishInit

```
__finishInit ( self )
```

__finishInit is a private helper function that finishes initialization by combining attributes to form new ones.

Inputs: None

Returns: Void.

Affects: Initializes class member variables that are combinations of existing ones.

Exceptions: None.

__getExperiments

```
__getExperiments (
        self,
        arg,
        dirname,
        names,
        )
```

__getExperiment is a private helper function called by os.path.walk in getExpList.

__getExperiments is called for each sub-directory in the experiments directory. Its purpose is to add any experiment directory paths from that directory that match the search criteria to the expList.

Inputs:

arg: a tuple containing all the filter arguments, plus the fileList to be appended to. The tuple elements are:

expName: a string defining what experiment names to accept (case insensitive). Use of unix file matching characters * and ? are allowed. If None, any experiment name allowed.

kinstList: a list of kinst (kind of instrument codes) integers that will be accepted. If None, all kinst values are accepted.

startDate: a python datetime in UTC. If None, do not reject any files.

endDate: a python datetime in UTC. If None, do not reject any files.

startDayOfYear: a Julian day number (1-366) after which to accept files from any given year. This can be used for example to only allow files from Spring, which has a start day of year of 80 (March 21). If None, do not reject any files.

endDayOfYear: a Julian day number (1-366) before which to accept files from any given year. This can be used for example to only allow files from Spring, which has a end day of year of 172 (June 21). If None, do not reject any files.

expList: the list of valid experiment paths to which is appended any newly-found valid experiment directories

showIgnoredExperiments: if False (default), ignore experiments where expTab.txt state code is set to ignore. If True, include those experiments.

enforcePathConvention: if True, only return experiments whose path is of the form convention 1999/mlh/20jan98d (YYYY/<three lower case letters>/DDmmmYY<optional single character>. If False (the default), only required path in the form 1999/mlh/*.

dirname: the directory name of the present directory

names: the list of filenames in the present directory

Returns: None.

Affects: Adds full file paths of any data files found that meet all filter criteria

Exceptions: None

__getFiles

```
__getFiles (
        self,
        arg,
        dirname,
        names,
        )
```

__getFiles is a private helper function called by os.path.walk in getFileList.

__getFiles is called for each sub-directory in the experiments directory. Its purpose is to add any file paths from that directory that match the search criteria to the fileList.

Inputs:

arg: a tuple containing all the filter arguments, plus the fileList to be appended to. The tuple elements are:

expName: a string defining what experiment names to accept (case insensitive). Use of unix file matching characters * and ? are allowed. If None, any experiment name allowed.

kinstList: a list of kinst (kind of instrument codes) integers that will be accepted. If None, all kinst values are accepted.

kindatList: a list of kindat (kind of data codes) integers that will be accepted. If None, all kindat values are accepted.

startDate: a python date/time in UTC (see time module - actually a tuple of nine integers) after which to accept files. If None, do not reject any files.

endDate: a python date/time in UTC (see time module - actually a tuple of nine integers) before which to accept files. If None, do not reject any files.

startDayOfYear: a Julian day number (1-366) after which to accept files from any given year. This can be used for example to only allow files from Spring, which has a start day of year of 80 (March 21). If None, do not reject any files.

endDayOfYear: a Julian day number (1-366) before which to accept files from any given year. This can be used for example to only allow files from Spring, which has a end day of year of 172 (June 21). If None, do not reject any files.

fileList: the list of valid file paths to which is appended any newly-found valid file paths

publicAccessOnly: if 1, only return files marked in fileTab.txt and expTab.txt as public. If 0, return all non-archive files. If 2, return public and public archive experiments with public files. If 3, return all files, including public, private, archived, and private archived.

enforcePathConvention: if 1, only return experiments whose path is of the form convention 1999/mlh/20jan98d (YYYY/<three lower case letters>/DDmmmYY<optional single character>. If 0 (the default), only require paths to be in the form 1999/mlh/*.

includeNonDefault: if 1, also include data files that are not default files in they match all other criteria. If 0 (the default), exclude non-default.

includeNonMadrigal: if 1, include all experiment files that are not in fileTab.txt. If 0, (the default) limit data files to those in fileTab.txt.

appendKinst: if 1, append kind of instrument integer to each file name, so that what is returned is a list of (file name, inst code) tuples. If 0 (the default), do not append instrument code; return a list of file names.

appendStartTime: if 1, append start time in seconds since 1950 to each file name, so that what is returned is a list of (file name, startTime) tuples. If 0 (the default), do not append startTimes.

includeRealtime: if 1, include realtime files even if includeNonDefault = 0. If 0 (default), do not include realtime if includeNonDefault = 0.

dirname: the directory name of the present directory

names: the list of filenames in the present directory

Returns: None.

Affects: Adds full file paths of any data files found that meet all filter criteria

Exceptions: None

__initFromHardCode

```
__initFromHardCode ( self )
```

__initFromHardCode is a private helper function that reads information from the automatically editted lines.

Inputs: None

Returns: Void.

Affects: Initializes class member variables that are found in the constants at the top of this file. These constants were set during installation.

Exceptions: None.

__init__

```
__init__ ( self,  initFile=None )
```

__init__ initializes the MadrigalDB by reading from $MAD_ROOT/madrigal.cfg (or initString).

Inputs: String representing the full path to the configuration file. Default is None, in which case configuration file is $(__MAD_ROOT)/__confFileName (now madrigal.cfg).

Returns: void

Affects: Initializes all the class member variables.

Exceptions: MadrigalError thrown if error (see class description for specific conditions).

Notes:

Given the file $MAD_ROOT/madrigal.cfg (these names are set by constants above), or initFile, this constructor goes about loading some basic data about the database, including information such as:

-the database utility directory

-the www home base

-the cgi home base

Implemented using ConfigParser module. This reads ini type files. The only difference between ini files and madrigal.cfg is that ini files are broken into sections of the form:

[sectionTitle]

key1 = value1

key2 = value2

Since madrigal.cfg (or another initFile) has no section header, one section head called "madrigal" is appended to the beginning of the file.

Exceptions

madrigal.admin.MadrigalError("Unable to parse configuration file " + self.__confFilePath, traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

__isValidEmail

```
__isValidEmail ( self,  emailStr )
```

__isValidEmail is a private helper function that does some checking to ensure a valid email address was found.

Inputs: emailStr - email address string to verify

Returns: 1 if no problems, 0 if not valid.

Affects: Nothing

Exceptions: None

__readConfFile

```
__readConfFile ( self )
```

__readConfFile is a private helper function that reads information from the parsed config file.

Inputs: None

Returns: Void.

Affects: Initializes class member variables that are found in the config file.

Exceptions: MadrigalError thrown if any key not found.

Exceptions

madrigal.admin.MadrigalError("Unable to find key " + self.__CON_TACTLINK + " in configuration file due to ConfigParser error", traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))
madrigal.admin.MadrigalError("Unable to find key " + self.__HTML_STYLE + " in configuration file due to ConfigParser error", traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))
madrigal.admin.MadrigalError("Unable to find key " + self.__INDEX_HEAD + " in configuration file due to

ConfigParser error", traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2])) madrigal.admin.MadrigalError("Unable to find key " + self.__MAD_SERVER + " in configuration file due to ConfigParser error", traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2])) madrigal.admin.MadrigalError("Unable to find key " + self.__MAD_SERVERCGI + " in configuration file due to ConfigParser error", traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2])) madrigal.admin.MadrigalError("Unable to find key " + self.__MAD_SERVERCGIABS + " in configuration file due to ConfigParser error", traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2])) madrigal.admin.MadrigalError("Unable to find key " + self.__MAD_SERVERDOCABS + " in configuration file due to ConfigParser error", traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2])) madrigal.admin.MadrigalError("Unable to find key " + self.__MAD_SERVERROOT + " in configuration file due to ConfigParser error", traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2])) madrigal.admin.MadrigalError("Unable to find key " + self.__SITE_ID + " in configuration file due to ConfigParser error", traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

__setFileAccess

```
__setFileAccess (
        self,
        arg,
        dirname,
        names,
        )
```

setFileAccess is a private helper function called by os.path.walk in setFileAccess.

Inputs:

arg: either 0 for public access, or 1 for private access.

dirname: the directory name of the present directory

names: the list of filenames in the present directory

Returns: None

Affects: sets all fileTab.txt files in dirname to be public or private, depending on arg.

Exceptions: None.

__str__

```
__str__ ( self )
```

__str__ simply calls toString

getBinDir

```
getBinDir ( self )
```

getBinDir returns the madrigal bin directory.

Inputs: None

Returns: The madrigal bin directory path. (eg /opt/madrigal/bin)

Affects: Nothing

Exceptions: None

getCGIHomeBase

```
getCGIHomeBase ( self )
```

getCGIHomeBase returns the full url to the cgi directory.

Inputs: None

Returns: String representing the full url to the cgi directory. (eg, http://haystack.mit.edu/cgi-bin/madrigal)

Affects: Nothing

Exceptions: None

getCgiDirPath

```
getCgiDirPath ( self )
```

getCgiDirPath returns the directory path to the main madrigal cgi-bin directory.

Inputs: None

Returns: the directory path to the main madrigal cgi-bin directory. (eg, "/export/home/httpd/apache/cgi-bin/madrigal")

Affects: Nothing

Exceptions: None

getContactEmail

```
getContactEmail ( self )
```

getContactEmail returns the email address of the site administrator.

Inputs: None

Returns: The email address of the site administrator.

Affects: Nothing

Exceptions: None

getContactLink

```
getContactLink ( self )
```

getContactLink returns contact email link tag (see getContactEmail for the email alone).

Inputs: None

Returns: The contact email link tag. (eg. <A HREF="MAILTO:brideout@haystack.mit.edu">madrigal@haystack</A>
)

Affects: Nothing

Exceptions: None

getDatabaseUtilityDirectory

```
getDatabaseUtilityDirectory ( self )
```

getDatabaseUtilityDirectory returns the full path to the database utility directory.

Inputs: None

Returns: String representing the full path to the database utility directory. (eg, /opt/madrigal/bin)

Affects: Nothing

Exceptions: None

getDocDirPath

```
getDocDirPath ( self )
```

getDocDirPath returns the directory path to the main madrigal html directory.

Inputs: None

Returns: the directory path to the main madrigal html directory. (eg, "/export/home/httpd/apache/htdocs/madrigal")

Affects: Nothing

Exceptions: None

getExpList

```
getExpList (
        self,
        expName=None,
        kinstList=None,
        startDate=None,
        endDate=None,
        startDayOfYear=None,
        endDayOfYear=None,
        showIgnoredExperiments=False,
        enforcePathConvention=False,
        )
```

getExpList returns a list of full experiment directory names that match the search arguments.

Inputs:

expName: a string defining what experiment names to accept (case insensitive). Use of unix file matching characters * and ? are allowed. If None (default), any experiment name allowed.

kinstList: a list of kinst (kind of instrument codes) integers that will be accepted. If None (default) or if list contains 0, all kinst values are accepted.

startDate: a datetime date. If None (default), do not reject any experiments.

endDate: a datetime date. If None (default), do not reject any experiments.

startDayOfYear: a Julian day number (1-366) after which to accept experiments from any given year. This can be used for example to only allow files from Spring, which has a start day of year of 80 (March 21). If None (default), do not reject any experiments.

endDayOfYear: a Julian day number (1-366) before which to accept experiments from any given year. This can be used for example to only allow files from Spring, which has a end day of year of 172 (June 21). If None (default), do not reject any experiments.

showIgnoredExperiments: if False (default), ignore experiments where expTab.txt state code is set to ignore. If True, include those experiments.

enforcePathConvention: if True, only return experiments whose path is of the form convention 1999/mlh/20jan98d (YYYY/<three lower case letters>/DDmmmYY<optional single character>. If False (the default), only required path in the form 1999/mlh/*.

Returns: a list of full experiment directory names that match the search arguments

Affects: Nothing

Exceptions: None

getExperimentDir

```
getExperimentDir ( self )
```

getExperimentDir returns the main experiment directory. No longer guarenteed to be unique,but will hold test files and geophysical experiments.

Inputs: None

Returns: The full experiment directory path. (eg. /opt/madrigal/experiments)

Affects: Nothing

Exceptions: None

getExperimentDirs

```
getExperimentDirs ( self )
```

getExperimentDirs returns a list of full paths of all valid experiment directories.

Inputs: None

Returns: list of full paths of all valid experiment directories. Valid experiment

directories are those of the form of the regular expression $/home/grail/brideout/madroot/experiments[0-9]*

Affects: Nothing

Exceptions: None

getFileList

```
getFileList (
        self,
        expName=None,
        kinstList=None,
        kindatList=None,
        startDate=None,
        endDate=None,
        startDayOfYear=None,
        endDayOfYear=None,
        publicAccessOnly=3,
        enforcePathConvention=0,
        includeNonDefault=0,
        includeNonMadrigal=0,
        appendKinst=0,
        appendStartTime=0,
        includeRealtime=0,
        )
```

getFileList returns a list of full file names that match the search arguments.

Inputs:

expName: a string defining what experiment names to accept (case insensitive). Use of unix file matching characters * and ? are allowed. If None (default), any experiment name allowed.

kinstList: a list of kinst (kind of instrument codes) integers that will be accepted. If None (default) or if list contains 0, all kinst values are accepted.

kindatList: a list of kindat (kind of data codes) integers that will be accepted. If None (default) or if list contains 0, all kindat values are accepted.

startDate: a python date (see time module - actually a tuple of nine integers) after which to accept files. If None (default), do not reject any files.

endDate: a python date (see time module - actually a tuple of nine integers) before which to accept files. If None (default), do not reject any files.

startDayOfYear: a Julian day number (1-366) after which to accept files from any given year. This can be used for example to only allow files from Spring, which has a start day of year of 80 (March 21). If None (default), do not reject any files.

endDayOfYear: a Julian day number (1-366) before which to accept files from any given year. This can be used for example to only allow files from Spring, which has a end day of year of 172 (June 21). If None (default), do not reject any files.

publicAccessOnly: if 1, only return files marked in fileTab.txt and expTab.txt as public. If 0, return all non-archive files. If 2, return public and public archive experiments with public files. If 3, return all files, including public, private, archived, and private archived. (the default)

enforcePathConvention: if 1, only return experiments whose path is of the form convention 1999/mlh/20jan98d (YYYY/<three lower case letters>/DDmmmYY<optional single character>. If 0 (the default), only require paths to be in the form 1999/mlh/*.

includeNonDefault: if 1, also include data files that are not default files in they match all other criteria. If 0 (the default), exclude non-default.

includeNonMadrigal: if 1, include all experiment files that are not in fileTab.txt. If 0, (the default) limit data files to those in fileTab.txt.

appendKinst: if 1, append kind of instrument integer to each file name, so that what is returned is a list of (file name, inst code) tuples. If 0 (the default), do not append instrument code; return a list of file names.

appendStartTime: if 1, append start time in seconds since 1950 to each file name, so that what is returned is a list of (file name, startTime) tuples. If 0 (the default), do not append startTimes.

includeRealtime: if 1, include realtime files even if includeNonDefault = 0. If 0 (default), do not include realtime if includeNonDefault = 0.

Returns: a list of full path file names (strings) that match the search arguments

Affects: Nothing

Exceptions: None

getFileListFromMetadata

```
getFileListFromMetadata (
        self,
        expName=None,
        kinstList=None,
        kindatList=None,
        startDate=None,
        endDate=None,
        startDayOfYear=None,
        endDayOfYear=None,
        publicAccessOnly=3,
        includeNonDefault=0,
        appendKinst=0,
        appendStartTime=0,
```

```
        includeRealtime=0,
        )
```

getFileListFromMetadata returns a list of full file names that match the search arguments using metadata only.

This method is very similar to getFileList, except that it assumes the metadata is correct, and doesn't search the actual experiment directories. It is therefore faster, but less robust.

Inputs:

expName: a string defining what experiment names to accept (case insensitive). Use of unix file matching characters * and ? are allowed. If None (default), any experiment name allowed.

kinstList: a list of kinst (kind of instrument codes) integers that will be accepted. If None (default) or if list contains 0, all kinst values are accepted.

kindatList: a list of kindat (kind of data codes) integers that will be accepted. If None (default) or if list contains 0, all kindat values are accepted.

startDate: a python date (see time module - actually a tuple of nine integers) after which to accept files. If None (default), do not reject any files.

endDate: a python date (see time module - actually a tuple of nine integers) before which to accept files. If None (default), do not reject any files.

startDayOfYear: a Julian day number (1-366) after which to accept files from any given year. This can be used for example to only allow files from Spring, which has a start day of year of 80 (March 21). If None (default), do not reject any files.

endDayOfYear: a Julian day number (1-366) before which to accept files from any given year. This can be used for example to only allow files from Spring, which has a end day of year of 172 (June 21). If None (default), do not reject any files.

publicAccessOnly: if 1, only return files marked in fileTab.txt and expTab.txt as public. If 0, return all non-archive files. If 2, return public and public archive experiments with public files. If 3, return all files, including public, private, archived, and private archived. (the default)

includeNonDefault: if 1, also include data files that are not default files in they match all other criteria. If 0 (the default), exclude non-default.

appendKinst: if 1, append kind of instrument integer to each file name, so that what is returned is a list of (file name, inst code) tuples. If 0 (the default), do not append instrument code; return a list of file names.

appendStartTime: if 1, append start time in seconds since 1950 to each file name, so that what is returned is a list of (file name, startTime) tuples. If 0 (the default), do not append startTimes.

includeRealtime: if 1, include realtime files even if includeNonDefault = 0. If 0 (default), do not include realtime if includeNonDefault = 0.

Returns: a list of full path file names (strings) that match the search arguments, and possibly kinst and starttime.

Affects: Nothing

Exceptions: None

getFullPathFromPartial

```
getFullPathFromPartial ( self,  partialPath )
```

getFullPathFromPartial returns the full path to a file or directory based on a partial path.

Follows the rule that if partialPath begins with [/]experiments, then the full path is madroot + partialPath. Otherwise (pre madrigal 2.6) full path is madroot + experiments + partialPath

Input: parial path (eg, 2010/mlh/15jan10 or experiments10/2010/mlh/15jan10)

Returns: full path (eg /opt/madrigal/experiments10/2010/mlh/15jan10)

getHtmlStyle

```
getHtmlStyle ( self )
```

getHtmlSyle returns the default html body tag for the site.

Inputs: None

Returns: The default html body tag for the site. (eg. <BODY BGCOLOR=#FFFF88 LINK=#008000 VLINK=#003366>)

Affects: Nothing

Exceptions: None

getIndexHead

```
getIndexHead ( self )
```

getIndexHead returns the heading of the top level madrigal page.

Inputs: None

Returns: The heading of the top level madrigal page. (eg. Welcome to the Madrigal Database <BR> at Ishtar)

Affects: Nothing

Exceptions: None

getLocalRulesOfRoad

```
getLocalRulesOfRoad ( self )
```

getLocalRulesOfRoad returns the local rules of the road.

Inputs: None

Returns: If the file madroot/local_rules_of_the_road.txt exists, returns the text of that. Else returns a default rules_of_road statement

Affects: Nothing

Exceptions: None

getMadServer

```
getMadServer ( self )
```

getMadServer returns the full name of the madrigal server (eg, haystack.mit.edu).

Inputs: None

Returns: String representing the url to the main database website.

Affects: Nothing

Exceptions: None

getMadroot

```
getMadroot ( self )
```

getMadroot returns the value of the environment variable __MAD_ROOT.

Inputs: None

Returns: The value of the environment variable __MAD_ROOT.

Affects: Nothing

Exceptions: None

getMadrootEnvVarName

```
getMadrootEnvVarName ( self )
```

getMadrootEnvVarName returns the name of the environment variable __MAD_ROOT (presently = MAD_ROOT).

Inputs: None

Returns: The name of the environment variable __MAD_ROOT.

Affects: Nothing

Exceptions: None

getMailserver

```
getMailserver ( self )
```

getMailserver returns the mailserver name.

Inputs: None

Returns: The mailserver name. If this heading is not found in madrigal.cfg, no error is thrown - simply defaults to localhost

Affects: Nothing

Exceptions: None

getMaxGlobalQueries

```
getMaxGlobalQueries ( self )
```

getMaxGlobalQueries returns the maximum number of global queries as an integer.

Inputs: None

Returns: The maximum number of global queries as an int. If this heading is not found in madrigal.cfg, no error is thrown - simply defaults to 2

Affects: Nothing

Exceptions: None

getMaxTempReports

```
getMaxTempReports ( self )
```

getMaxTempReports returns the maximum size of the tempReports dir in GB as a float.

Inputs: None

Returns: The maximum nsize of the tempReports dir in GB as a float. If this heading is not found in madrigal.cfg, no error is thrown - simply defaults to 2

Affects: Nothing

Exceptions: None

getMetadataDir

```
getMetadataDir ( self )
```

getMetadataDir returns the metadata directory.

Inputs: None

Returns: The full metadata directory path. (eg. /opt/madrigal/metadata)

Affects: Nothing

Exceptions: None

getPythonExecutable

```
getPythonExecutable ( self )
```

getPythonExecutable returns the full path to the python executable.

Inputs: None

Returns: the full path to the python executable. If this heading is not found in madrigal.cfg, no error is thrown - simply defaults to madroot/bin/python

Affects: Nothing

Exceptions: None

getRelativeCGI

```
getRelativeCGI ( self )
```

getRelativeCGI returns the relative url to the cgi directory.

Inputs: None

Returns: String representing the relative url to the cgi directory. (eg, cgi-bin/madrigal)

Affects: Nothing

Exceptions: None

getRelativeTopLevel

```
getRelativeTopLevel ( self )
```

getRelativeTopLevel returns the relative url of the top level directory in main database website.

Inputs: None

Returns: String representing the relative url to the top level directory in main database website. (eg, madrigal)

Affects: Nothing

Exceptions: None

getSiteID

```
getSiteID ( self )
```

getSiteID returns the site id number.

Inputs: None

Returns: The site id (integer) of the madrigal installation.

Affects: Nothing

Exceptions: If non-integer found

Exceptions

madrigal.admin.MadrigalError("Site id not an integer in madrigal configuration file " + self.__confFile, traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

getTopLevelUrl

```
getTopLevelUrl ( self )
```

getTopLevelUrl returns the full url of the top level directory in main database website.

Inputs: None

Returns: String representing the full url to the top level directory in main database website. (eg, http://haystack.mit.edu/madrigal)

Affects: Nothing

Exceptions: None

getWWWHomeBase

```
getWWWHomeBase ( self )
```

getWWWHomeBase returns the url to the main database website(eg, http://haystack.mit.edu).

Inputs: None

Returns: String representing the url to the main database website.

Affects: Nothing

Exceptions: None

listFileTimes

```
listFileTimes (
        self,
        expDir=None,
        relative=True,
        )
```

listFileTimes returns a list of (filename, datetime_ut) of all files in the experiment directory

Inputs:

expDir - the particular subdirectory of an experiment directory to list. If None (the default), will list all files for all experiment directories. May be an absolute path, or may start with experiments[0-9]*.

relative - if True (the default) give the path relative to the experiments[0-9]* directory. If False, give full path

Returns: a list of tuples, where each tuple has 1. the file path, and 2. a UT datetime object of the last file modification.

Exceptions: raised if expDir is not a valid experiments directory or subdirectory

Exceptions	
	ValueError, 'expDir %s not a valid experiment directory' %(expDir)

setFileAccess

```
setFileAccess (
        self,
        expDirectory,
        accessMode,
        )
```

setFileAccess sets all fileTab.txt files in all subdirectories of expDirectory to be public or private.

Inputs:

expDirectory: The full path to a directory in the experiment directory. That is, it may be madroot/experiments[0-9]* or any directory under it.

accessMode: either 0 for public access, or 1 for private access.

Returns: None

Affects: sets all fileTab.txt files in all subdirectories of expDirectory to be public

or private.

Exceptions: If accessMode is not 1 or 0.

Exceptions

madrigal.admin.MadrigalError('MadrigalDB.setFileAccess called with accessMode = ' + str(accessMode) + ', must be either 0 or 1', None)

tarExperiments

```
tarExperiments (
        self,
        tarFileName,
        startDate=None,
        endDate=None,
        excludePrivData=0,
        ignoreDirCon=1,
        includeNonDefData=0,
        onlyData=0,
        filetype=0,
        verbose=0,
        )
```

tarExperiments creates a tar file containing files from madroot/experiments[0-9]*.

Note: this method sometimes requires the modification of the fileTab.txt files found in the experiments directory. This is because some data files might be excluded, so that the fileTab.txt file will no longer be accurate. Because of this, all files to be tar'ed will be copied to /tmp/temp<random num>/experiments, where the fileTab.txt files will be modified. When done, this temp dir will be deleted.

Inputs:

tarFileName: The full path to a tar file to be created.

startDate: a python date (see time module - actually a tuple of nine integers) after which to accept files. If None (default), do not reject any files.

endDate: a python date (see time module - actually a tuple of nine integers) before which to accept files. If None (default), do not reject any files.

excludePrivData: if 1, allow data marked as private to be omitted (and the line from the fileTab.txt to be removed). If 0 (the default), all data, public and private, will be included.

ignoreDirCon: if 1, ignore convention that directory must be in form 1999/mlh/03sep99 (the default). If 0 , reject non-standard directories.

includeNonDefData: if 1, include all files listed in fileTab.txt. If 0 (the default), reject non-default files, and modify fileTab.txt to remove non-default listings.

onlyData: if 1, reject all files not listed in fileTab.txt. If 0 (the default), accept all files in a directory not mentioned in fileTab.txt.

filetype: format to save data files as. Default 0 is to leave present format unchanged. <type> is an integer as follows:

type = 0 Leave present format unchanged (default)

type = 1 Madrigal

type = 2 Blocked Binary

type = 3 Cbf

type = 4 Unblocked binary

type = 5 Ascii

verbose: if 1, print to std out the list of files included (relative path). If 0, (the default) print nothing.

Returns: None

Affects: created tar file tarFileName of selected files from madroot/experiments.

Exceptions: If unable to read any experiment file.

Exceptions

madrigal.admin.MadrigalError('Unable to tar experiments because denied write permission ' + 'for ' + str(filename), None)
madrigal.admin.MadrigalError('Unable to tar experiments because denied write permission ' + 'for ' + str(os.path.dirname(filename)), None)
madrigal.admin.MadrigalError('In tarExperiments could not remove dir ' + tempDir, traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

toString

```
toString ( self )
```

toString returns a simple string representation of a MadrigalDB object.

Inputs: None

Returns: String describing a simple representation of a MadrigalDB object.

Affects: Nothing

Exceptions: None

Table of Contents

This document was automatically generated on Thu Oct 20 16:51:50 2011 by HappyDoc version r1_5

Table of Contents

Class: MadrigalExperiment

MadrigalExperiment is an object that provides access to Madriga info from the metadata.

This object provides access to all Madrigal experiment information in the metadata file expT

Usage example:

```
import madrigal.metadata

import madrigal.admin

try:

    test = madrigal.metadata.MadrigalExperiment()

    print test.getExperimentName(3001)

except madrigal.admin.MadrigalError, e:

    print e.getExceptionStr()
```

Non-standard Python modules used: None

MadrigalError exception thrown if:

1. MadrigalMetadata fails to open or parse metadata file
2. Columns expected to be ints or floats cannot be converted

Change history:

Written by Bill Rideout Apr. 17, 2002

Methods

__compareDateSite__

```
__compareDateSite__ (
        self,
        first,
        second,
        )
```

__compareDateSite__ is a private method to help sort b site.

first, second - lists of experiment information as parsed from expTab.txt filesse

__getPositionByExpIdBinarySearch__

```
__getPositionByExpIdBinarySearch__ ( self,  expId )
```

__getPositionByExpIdBinarySearch__ is a private method that does a binary s

May fail if experiment ids are not in order, or if expId does not exist. If fails, re

Input: expId - experiment ID (int) to search for

Returns: position (int) of expId. First position = 0. Returns None if not found.

__init__

```
__init__ (
        self,
        madDB=None,
        initFile=None,
        )
```

__init__ initializes MadrigalExperiment by reading from initFile).

Inputs: Existing MadrigalDB object, by default = None.

String representing the full path to the metadata file. Default is None, in which MadrigalDB.getMetadataDir()/expTab.txt.

Returns: void

Affects: Initializes all the class member variables.

Exceptions: MadrigalError thrown by MadrigalMetadata if file not opened or p

__str__

```
__str__ ( self )
```

return possibly modified file as a string in same format as writeMetadata

getExpCount

```
getExpCount ( self )
```

getExpCount returns number of experiments in MadrigalExperiment object

getExpDirByExpId

```
getExpDirByExpId ( self,  expId )
```

getExpDirByExpId returns the full experiment directory experiment id.

Inputs: Experiment Id (integer).

Returns: the full experiment directory (string). Returns None if experiment id n to determine directory.

Affects: None

Exceptions: None

getExpDirByPosition

```
getExpDirByPosition ( self,  position=0 )
```

getExpDirByPosition returns the full experiment directo at given position.

Inputs: position of experiment in list (first position is zero). Defaults to first.

Returns: the full experiment directory, or None if position >= number of experi determine directory.

Affects: None

Exceptions: None

getExpEndDateTimeByExpId

```
getExpEndDateTimeByExpId ( self,  expId )
```

getExpEndDateTimeByExpId returns the ending date/ti for a given experiment id.

Inputs: Experiment Id (integer).

Returns: the experiment end date/time in the standard python form of 9 item tu experiment id not found. Since mktime does not go before 1970, I use date.c m week and DST flag

Affects: None

Exceptions: None

getExpEndDateTimeByPosition

```
getExpEndDateTimeByPosition ( self,  position=0 )
```

getExpEndDateTimeByPosition returns the ending date experiment at given position.

Inputs: position of experiment in list (first position is zero). Defaults to first.

Returns: the experiment end date/time in the standard python form of 9 item tu position >= number of experiments. Since mktime does not go before 1970, I u only day of week and DST flag

Affects: None

Exceptions: None

getExpIdByPosition

```
getExpIdByPosition ( self,  position=0 )
```

getExpIdByPosition returns the experiment id of the exp position.

Inputs: position of experiment in list (first position is zero). Defaults to first.

Returns: the experiment id (integer), or None if position >= number of experim

Affects: None

Exceptions: MadrigalError if any item in row cannot be cast to correct format

Exceptions	
	madrigal.admin.MadrigalError('Error in expTab.txt pars position), traceback.format_exception(sys.exc_info() [sys.exc_info() [2]))

getExpLinksByExpId

```
getExpLinksByExpId ( self,  expId )
```

getExpLinksByExpId returns a list of (title, url) tuples co this experiment.

Inputs:

Inputs: Experiment Id (integer).

Returns: a list of (title, url) tuples containing all links for this experiment.

In order to be a link, a file must be in the experiment directory in the form *.htm */*/index.html. The title is parsed from the title in the head; if not found, return follow file extensions are also links if found in the main experiment directory: `pdf`, 'gif']. For these files, the title is simply the basename.

Affects: None

Exceptions: None

getExpNameByExpId

```
getExpNameByExpId ( self,  expId )
```

getExpNameByExpId returns the experiment name for a

Inputs: Experiment Id (integer).

Returns: the experiment name (string). Returns None if experiment id not foun

Affects: None

Exceptions: None

getExpNameByPosition

```
getExpNameByPosition ( self,  position=0 )
```

getExpNameByPosition returns the experiment name o given position.

Inputs: position of experiment in list (first position is zero). Defaults to first.

Returns: the experiment name, or None if position >= number of experiments.

Affects: None

Exceptions: None

getExpSiteIdByExpId

```
getExpSiteIdByExpId ( self,  expId )
```

getExpSiteIdByExpId returns the site id (int) for a given

Inputs: Experiment Id (integer).

Returns: the site id for this experiment. Returns None if experiment id not foun

Affects: None

Exceptions: None

getExpSiteIdByPosition

```
getExpSiteIdByPosition ( self,  position=0 )
```

getExpSiteIdByPosition returns the experiment site id c given position.

Inputs: position of experiment in list (first position is zero). Defaults to first.

Returns: the experiment site id (integer), or None if position >= number of exp

Affects: None

Exceptions: MadrigalError if any item in row cannot be cast to correct format

Exceptions

madrigal.admin.MadrigalError('Error in expTab.txt pars
position), traceback.format_exception(sys.exc_info() [
sys.exc_info() [2]))

getExpStartDateTimeByExpId

```
getExpStartDateTimeByExpId ( self,  expId )
```

getExpStartDateTimeByExpId returns the starting date/t for a given experiment id.

Inputs: Experiment Id (integer).

Returns: the experiment start date/time in the standard python form of 9 item tu
experiment id not found. Since mktime does not go before 1970, I use date.c m
week and DST flag

Affects: None

Exceptions: None

getExpStartDateTimeByPosition

```
getExpStartDateTimeByPosition ( self,  position=0 )
```

getExpStartDateTimeByPosition returns the starting da experiment at given position.

Inputs: position of experiment in list (first position is zero). Defaults to first.

Returns: the experiment start date/time in the standard python form of 9 item tu
position >= number of experiments. Since mktime does not go before 1970, I u
only day of week and DST flag

Affects: None

Exceptions: None

getExpUrlByExpId

```
getExpUrlByExpId ( self,  expId )
```

getExpUrlByExpId returns the experiment url for a given

Inputs: Experiment Id (integer).

Returns: the experiment url (string). Returns None if experiment id not found.

Affects: None

Exceptions: None

getExpUrlByPosition

```
getExpUrlByPosition ( self,  position=0 )
```

getExpUrlByPosition returns the experiment url of the e position.

Inputs: position of experiment in list (first position is zero). Defaults to first.

Returns: the experiment url, or None if position >= number of experiments.

Affects: None

Exceptions: None

getKinstByExpId

```
getKinstByExpId ( self,  expId )
```

getKinstByExpId returns the kinst (integer) of the exper experiment id.

Inputs: Experiment Id (integer).

Returns: the experiment kinst (integer). Returns None if experiment id not four

Affects: None

Exceptions: None

Exceptions	
	madrigal.admin.MadrigalError('Error in expTab.txt pars position), traceback.format_exception(sys.exc_info() [sys.exc_info() [2]))

getKinstByPosition

```
getKinstByPosition ( self,  position=0 )
```

getKinstByPosition returns the kinst (kind of instrumen experiment at given position.

Inputs: position of file in list (first position is zero). Defaults to first.

Returns: the kindat (kind of data code) of the file at given position as an intege number of files.

Affects: None

Exceptions: Thrown if kindat column cannot be parsed into an integer

Exceptions

madrigal.admin.MadrigalError('Error in expTab.txt pars position), traceback.format_exception(sys.exc_info() [sys.exc_info() [2]))

getLine

```
getLine ( self,  position )
```

getLine returns the line at a given position. Returns Nor number of lines.

Inputs: position - position in file. First line = 0

getPIByExpId

```
getPIByExpId ( self,  expId )
```

getPIByExpId returns the principal investigator (string) a given experiment id.

Inputs: Experiment Id (integer).

Returns: the principal investigator's name (string). Returns None if experiment expTab.txt files may have this column, returns None if the column does not exi

Affects: None

Exceptions: None

This method added in Madrigal 2.6

getPIByPosition

```
getPIByPosition ( self,  position=0 )
```

getPIByPosition returns the principal investigator of the position.

Inputs: position of file in list (first position is zero). Defaults to first.

Returns: the principal investigator's name (string) of the file at given position a position >= number of files. Since not all expTab.txt files may have this colum does not exist.

Affects: None

Exceptions: None

This method added in Madrigal 2.6

getPIEmailByExpId

```
getPIEmailByExpId ( self,  expId )
```

getPIEmailByExpId returns the principal investigator em experiment for a given experiment id.

Inputs: Experiment Id (integer).

Returns: the principal investigator's email (string). Returns None if experiment expTab.txt files may have this column, returns None if the column does not exi

Affects: None

Exceptions: None

This method added in Madrigal 2.6

getPIEmailByPosition

```
getPIEmailByPosition ( self,  position=0 )
```

getPIEmailByPosition returns the principal investigator experiment at given position.

Inputs: position of file in list (first position is zero). Defaults to first.

Returns: the principal investigator's email (string) of the file at given position a position >= number of files. Since not all expTab.txt files may have this colum does not exist.

Affects: None

Exceptions: None

This method added in Madrigal 2.6

getSecurityByExpId

```
getSecurityByExpId ( self,  expId )
```

getSecurityByExpId returns the security code (integer) a given experiment id.

Inputs: Experiment Id (integer).

Returns: the security code (integer). Returns None if experiment id not found.

Affects: None

Exceptions: None

Exceptions	
	madrigal.admin.MadrigalError('Error in expTab.txt pars position), traceback.format_exception(sys.exc_info() [sys.exc_info() [2]))

getSecurityByPosition

```
getSecurityByPosition ( self,  position=0 )
```

getSecurityByPosition returns the security code of the position.

Inputs: position of file in list (first position is zero). Defaults to first.

Returns: the security code (integer) of the file at given position as an integer. R number of files.

Affects: None

Exceptions: Thrown if security column cannot be parsed into an integer

Exceptions	
	madrigal.admin.MadrigalError('Error in expTab.txt pars position), traceback.format_exception(sys.exc_info() [sys.exc_info() [2]))

setExpEndDateTimeByPosition

```
setExpEndDateTimeByPosition (
        self,
        endDateTime,
        position=0,
        )
```

setExpEndDateTimeByPosition sets a new MadrigalExp time by position.

Inputs:

endDateTime - a python datetime object to set the exp end date and time to.

position - which experiment row to change - defaults to 0

Returns: None.

Affects: sets exp end date and time in self.__fileList.

Exceptions: None

setExpIdByPosition

```
setExpIdByPosition (
        self,
        position,
        expId,
        )
```

setExpIdByPosition sets the experiment id of the experi position.

Inputs:

position - position of experiment in list (first position is zero).

expId - the new experiment id to use

Returns: None.

Affects: sets the experiment id of the experiment at given position

Exceptions: MadrigalError if any item in row cannot be cast to correct format,

Exceptions	
	madrigal.admin.MadrigalError('Error in setExpIdByPos position, expId)), traceback.format_exception(sys.exc_

], sys.exc_info() [2])))

setExpKinstByPosition

```
setExpKinstByPosition (
        self,
        position,
        expKinst,
        )
```

setExpKinstByPosition sets the experiment kinst of the position.

Inputs:

position - position of experiment in list (first position is zero).

expKinst - the new experiment kinst to use

Returns: None.

Affects: sets the experiment kinst of the experiment at given position

Exceptions: MadrigalError if any item in row cannot be cast to correct format,

Exceptions

madrigal.admin.MadrigalError('Error in setExpKinstBy
%(str(position, expKinst), traceback.format_exception
sys.exc_info() [1], sys.exc_info() [2])))

setExpNameByPosition

```
setExpNameByPosition (
        self,
        position,
        expName,
        )
```

setExpNameByPosition sets the experiment name of th position.

Inputs:

position - position of experiment in list (first position is zero).

expName - the new experiment name to use

Returns: None.

Affects: sets the experiment name of the experiment at given position

Exceptions: MadrigalError if any item in row cannot be cast to correct format,

Exceptions	
	madrigal.admin.MadrigalError('Error in setExpNameBy %(str(position, expName), [traceback.format_exc()]) madrigal.admin.MadrigalError('Error in setExpNameBy %(str(position, expName), traceback.format_exception sys.exc_info() [1], sys.exc_info() [2])))

setExpSiteIdByPosition

```
setExpSiteIdByPosition (
        self,
        position,
        expSiteId,
        )
```

setExpSiteIdByPosition sets the experiment site id of th position.

Inputs:

position - position of experiment in list (first position is zero).

expSiteId - the new experiment site id to use

Returns: None.

Affects: sets the experiment site id of the experiment at given position

Exceptions: MadrigalError if any item in row cannot be cast to correct format,

Exceptions	
	madrigal.admin.MadrigalError('Error in setExpSiteIdBy %(str(position, expSiteId), traceback.format_exception sys.exc_info() [1], sys.exc_info() [2])))

setExpStartDateTimeByPosition

```
setExpStartDateTimeByPosition (
        self,
        startDateTime,
        position=0,
        )
```

setExpStartDateTimeByPosition sets a new MadrigalEx and time by position.

Inputs:

startDateTime - a python datetime object to set the exp start date and time to.

position - which experiment row to change - defaults to 0

Returns: None.

Affects: sets exp start date and time in self.__fileList.

Exceptions: None

setExpUrlByPosition

```
setExpUrlByPosition (
        self,
        position,
        expUrl,
        )
```

setExpUrlByPosition sets the experiment url of the exp position.

Inputs:

position - position of experiment in list (first position is zero).

expUrl - the new experiment url to use

Returns: None.

Affects: sets the experiment url of the experiment at given position

Exceptions: MadrigalError if any item in row cannot be cast to correct format,

Exceptions

madrigal.admin.MadrigalError('Error in setExpUrlByPo
%(str(position, expUrl), [traceback.format_ex()]))
madrigal.admin.MadrigalError('Error in setExpUrlByPo
%(str(position, expUrl), [traceback.format_exc()]))

setPIByPosition

```
setPIByPosition (
        self,
        position,
        PI,
```

```
        )
```

setPIByPosition sets the principal investigator (string) o given position.

Inputs:

position - position of experiment in list (first position is zero).

PI - the new experiment principal investigator's name (string) to use

Returns: None.

Affects: sets the principal investigator of the experiment at given position

Exceptions: MadrigalError if any item in row cannot be cast to correct format,

This method added in Madrigal 2.6

Exceptions	
	madrigal.admin.MadrigalError('Error in setPIByPositio position, PI), [traceback.format_exc()]))

setPIEmailByPosition

```
setPIEmailByPosition (
        self,
        position,
        PIEmail,
        )
```

setPIEmailByPosition sets the principal investigator em experiment at given position.

Inputs:

position - position of experiment in list (first position is zero).

PIEmail - the new experiment principal investigator's email (string) to use

Returns: None.

Affects: sets the principal investigator email of the experiment at given positio

Exceptions: MadrigalError if any item in row cannot be cast to correct format,

This method added in Madrigal 2.6

Exceptions

madrigal.admin.MadrigalError('Error in setPIEmailByP
%(str(position, PIEmail), [traceback.format_exc()]))

setSecurityByPosition

```
setSecurityByPosition (
        self,
        position,
        securityCode,
        )
```

setSecurityByPosition sets the security code (integer) c given position.

Inputs:

position - position of experiment in list (first position is zero).

securityCode - the new experiment security code (integer) to use

Returns: None.

Affects: sets the security code of the experiment at given position

Exceptions: MadrigalError if any item in row cannot be cast to correct format,

Exceptions

madrigal.admin.MadrigalError('Error in setSecurityByP
%(str(position, securityCode), traceback.format_excep
sys.exc_info() [1], sys.exc_info() [2])))

sortByDateSite

```
sortByDateSite ( self )
```

sortByDateSite will resort self.__fileList so that experiments are listed first by by site

writeMetadata

```
writeMetadata ( self,  newFullPath=None )
```

writeMetadata writes a new version of the expTab.txt fil

Inputs: newFullPath: a new path to write the expTab.txt file to, if not the same opened. Defaults to None, which overwrites metadata file that was read from.

Returns: None.

Affects: Writes updated version of metadata file.

Exceptions: If unable to write file

Exceptions	
	madrigal.admin.MadrigalError("Unable to write metada traceback.format_exception(sys.exc_info() [0], sys.exc]))

Table of Contents

This document was automatically generated on Thu Oct 20 16:51:50 2011 by HappyDoc version r1_5

Table of Contents

Class: MadrigalInstrument **metadata.py**

MadrigalInstrument is an object that provides access to Madrigal instrument info from the metadata.

This object provides access to all Madrigal instrument information in the metadata files instTab.txt and instType.Tab.

Usage example:

```
import madrigal.metadata

import madrigal.admin

try:

    test = madrigal.metadata.MadrigalInstrument()

    print test.getInstrumentName(30)

except madrigal.admin.MadrigalError, e:

    print e.getExceptionStr()
```

Non-standard Python modules used: None

MadrigalError exception thrown if:

1. MadrigalMetadata fails to open or parse metadata file
2. Columns expected to be ints or floats cannot be converted

Change history:

Written by Bill Rideout Nov. 9, 2001

Methods

__init__

```
__init__ (
        self,
        madDB=None,
        initFile=None,
        init2File=None,
        )
```

__init__ initializes MadrigalInstrument by reading from instTab.txt (or initFile) and instType.txt file (or init2File).

Inputs:

madDB - Existing MadrigalDB object, by default = None.

initFile - String representing the full path to the metadata file. Default is None, in which case file read is MadrigalDB.getMetadataDir()/instTab.txt.

init2File - String representing the full path to the metadata file instType.txt. Default is None, in which case file read is MadrigalDB.getMetadataDir()/instType.txt.

Returns: void

Affects: Initializes all the class member variables.

Exceptions: MadrigalError thrown by MadrigalMetadata if file not opened or parsed successfully. Note that the instTab.txt file was updated with the release of the madrigal python api, and this function will throw an error if used with the old file.

__instrumentSort

```
__instrumentSort (
        self,
```

```
        thisInst,
        otherInst,
        )
```

instrumentSort is a private method used to sort tuples of instrument data

getAltitude

```
getAltitude ( self,  kinst )
```

getAltitude returns the altitude as a float that matches kinst argument, or None if not found or blank.

Inputs: kinst integer to get altitude.

Returns: the altitude in km above sea level as a float that matches kinst argument, or None if not found or blank.

Affects: None

Exceptions: MadrigalError if any item in row cannot be cast to correct format

Exceptions	
	madrigal.admin.MadrigalError('Error in instTab.txt parsing metadata row: ' + str(inst), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

getCategory

```
getCategory ( self,  kinst )
```

getCategory returns the instrument category that matches kinst argument as a string, or None if kinst not found.

Inputs: kinst integer to get altitude.

Returns: the instrument category that matches kinst argument as a string, or None if kinst not found.

Affects: None

Exceptions: MadrigalError if any item in row cannot be cast to correct format

Exceptions

madrigal.admin.MadrigalError('Error in instTab.txt parsing metadata row: ' + str(inst), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))
madrigal.admin.MadrigalError('Error in instTtpe.txt parsing metadata row: ' + str(inst2), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

getCategoryId

```
getCategoryId ( self,  kinst )
```

getCategory returns the instrument category that matches kinst argument as a string, or None if kinst not found.

Inputs: kinst integer to get altitude.

Returns: the instrument category that matches kinst argument as a string, or None if kinst not found.

Affects: None

Exceptions: MadrigalError if any item in row cannot be cast to correct format

Exceptions

madrigal.admin.MadrigalError('Error in instTab.txt parsing metadata row: ' + str(inst), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

getContactEmail

```
getContactEmail ( self,  kinst )
```

getContactEmail returns the contact email as a string that matches kinst argument, or None if not found or blank.

Inputs: kinst integer to get contact email.

Returns: the contact email as a string that matches kinst argument, or None if not found or blank.

Affects: None

Exceptions: MadrigalError if any item in row cannot be cast to correct format

This method added in Madrigal 2.6

Exceptions	
	madrigal.admin.MadrigalError('Error in instTab.txt parsing metadata row: ' + str(inst), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

getContactName

```
getContactName ( self,  kinst )
```

getContactName returns the contact name as a string that matches kinst argument, or None if not found or blank.

Inputs: kinst integer to get contact name.

Returns: the contact name as a string that matches kinst argument, or None if not found or blank.

Affects: None

Exceptions: MadrigalError if any item in row cannot be cast to correct format

This method added in Madrigal 2.6

Exceptions	
	madrigal.admin.MadrigalError('Error in instTab.txt parsing metadata row: ' + str(inst), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

getInstrumentList

```
getInstrumentList ( self )
```

getInstrumentList returns a list of all instrument names, mnemonics, and their kinst values.

Inputs: None.

Returns: a list of all instrument names, mnemonics, and their kinst values. Each item in the list is a tuple of the form (Instrument Name (string), mnemonic (string), kinst (integer)). Example item: (`Millstone Hill UHF Steerable Antenna`, `mlh`, 31)

Affects: None

Exceptions: MadrigalError if any item in row cannot be cast to correct format

Exceptions

madrigal.admin.MadrigalError('Error in instTab.txt parsing metadata row: ' + str(inst), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

getInstrumentMnemonic

```
getInstrumentMnemonic ( self,  kinst )
```

getInstrumentMnemonic returns the 3 char instrument mnemonic that matches kinst argument, or None if not found.

Inputs: kinst integer to get instrument mnemonic.

Returns: the instrument mnemonic that matches kinst argument, or None if not found.

Affects: None

Exceptions: MadrigalError if any item in row cannot be cast to correct format

Exceptions

madrigal.admin.MadrigalError('Error in instTab.txt parsing metadata row: ' + str(inst), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

getInstrumentName

```
getInstrumentName ( self,  kinst )
```

getInstrumentName returns the instrument name that matches kinst argument, or None if not found.

Inputs: kinst integer to get instrument name.

Returns: the instrument name that matches kinst argument, or None if not found.

Affects: None

Exceptions: MadrigalError if any item in row cannot be cast to correct format

Exceptions	
	madrigal.admin.MadrigalError('Error in instTab.txt parsing metadata row: ' + str(inst), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

getLatitude

```
getLatitude ( self,  kinst )
```

getLatitude returns the latitude as a float that matches kinst argument, or None if not found or blank.

Inputs: kinst integer to get latitude.

Returns: the latitude as a float that matches kinst argument, or None if not found or blank.

Affects: None

Exceptions: MadrigalError if any item in row cannot be cast to correct format

Exceptions	
	madrigal.admin.MadrigalError('Error in instTab.txt parsing metadata row: ' + str(inst), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

getLongitude

`getLongitude ( self,  kinst )`

getLongitude returns the longitude as a float that matches kinst argument, or None if not found or blank.

Inputs: kinst integer to get longitude.

Returns: the longitude as a float that matches kinst argument, or None if not found or blank.

Affects: None

Exceptions: MadrigalError if any item in row cannot be cast to correct format

Exceptions

madrigal.admin.MadrigalError('Error in instTab.txt parsing metadata row: ' + str(inst), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

getOrderedInstrumentList

`getOrderedInstrumentList ( self )`

getOrderedInstrumentList returns a list of all (instrument names, mnemonics, kinst, categories, categoryId), ordered by categoryId and then kinst.

Inputs: None.

Returns: a list of tuples of (instrument name, mnemonic, kinst, category, categoryId) ordered by categoryId and then kinst. Example item: (`Millstone Hill UHF Steerable Antenna`, `mlh`, 31, `Incoherent Scatter Radars`, 1)

Affects: None

Exceptions: MadrigalError if any item in row cannot be cast to correct format

Exceptions

madrigal.admin.MadrigalError('Error in instTab.txt parsing metadata row: ' + str(inst), traceback.format_exception(sys.exc_info() [0],

sys.exc_info() [1], sys.exc_info() [2]))

getOrderedInstrumentListWithData

```
getOrderedInstrumentListWithData (
        self,
        isTrusted,
        localOnly=False,
        localExpObj=None,
        globalExpObj=None,
        allowArchive=False,
        requireFiles=False,
        )
```

getInstrumentList returns information about which instruments have local and global data.

Inputs:

isTrusted - True if client is trusted, False otherwise

localOnly - if False (the default), will return information about both local and global data. If True, then global data ignored.

localExpObj - an MadrigalExperiment object for the local data. If None (the default), will be created.

globalExpObj - an MadrigalExperiment object for the global data. If None (the default), will be created if not localOnly.

allowArchive - if True, allow experiments marked as security==2(public arcive), and if isTrusted, allow security==3(private archive)

requireFiles - if True, only include experiments with Madrigal data files. If False (default), not not require that.

Returns: an ordered tuple of two items: 1. a list of tuples of (instName, localStartYear, localEndYear, globalStartYear, globalEndYear, kinst, categoryId), ordered by by categoryId, then kinst. If localOnly, globalStartYear and globalEndYear = 0. If not local only, localStartYear and localEndYear will be zero if no local data for that instrument. 2. categoryDict - key = categoryId, value = category Description

Affects: None

Exceptions: MadrigalError if any item in row cannot be cast to correct format

---

getLongitude returns the longitude as a floatthat matches kinst argument, or None if notfound or blank.

Table of Contents

This document was automatically generated on Thu Oct 20 16:51:50 2011 by HappyDoc version r1_5

Table of Contents

Class: MadrigalInstrumentKindats **metadata.py**

MadrigalInstrumentKindats is an object that provides access to the metadata file that summarizes the kindat codes associated with each instrument.

This object provides access to all Madrigal instrument kindat information in the metadata file instKindatTab.txt. The metadata file instKindatTab.txt lists, for any given instrument, all the kindat codes found in all the data files in the database associated with that instrument.

This class also contains a method to rebuild the table instKindatTab.txt by examining all the metadata in the database. This is presumably a somewhat slow process and should be done in the background.

Usage example:

```
import madrigal.metadata

import madrigal.admin

try:

    test = madrigal.metadata.MadrigalInstrumentKindats()

    print test.getKindatListForInstruments([20,30])

except madrigal.admin.MadrigalError, e:

    print e.getExceptionStr()
```

Non-standard Python modules used: None

MadrigalError exception thrown if:

1. MadrigalMetadata fails to open or parse metadata file

Change history:

Written by Bill Rideout Aug. 15, 2002

Methods

__init__
getKindatListForInstruments

rebuildInstKindatTable

__init__

```
__init__ (
        self,
        madDB=None,
        initFile=None,
        )
```

__init__ initializes MadrigalInstrumentKindats by reading from instKindatTab.txt (or initFile).

Inputs: Existing MadrigalDB object, by default = None.

String representing the full path to the metadata file. Default is None, in which case file read is MadrigalDB.getMetadataDir()/instKindatTab.txt.

Returns: void

Affects: Initializes all the class member variables.

Exceptions: MadrigalError thrown by MadrigalMetadata if file not opened or parsed successfully. Note that the instKindatTab.txt file was new with the release of the madrigal python api, and this function will throw an error if file not there.

getKindatListForInstruments

```
getKindatListForInstruments ( self,  kinstList )
```

getKindatListForInstruments returns a list of kindat codes as integers for the given instrument list.

Inputs: kinstList: a list of kinst integers to get associated kindat list. Also accepts a single integer.

Returns: a list of kindat codes as integers associated with the given instrument list.

Affects: None

Exceptions: if error in metadata file

Exceptions

madrigal.admin.MadrigalError('Error in instKindatTab.txt parsing metadata row: ' + str(inst), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

rebuildInstKindatTable

```
rebuildInstKindatTable ( self )
```

rebuildInstKindatTable rebuilds the instKindatTab.txt metadata file.

The table instKindatTab.txt is a listing of every kindat found for a given instrument. Since these files are constantly updated, this table needs to be updated on a regular basis. Data from other Madrigal sites is also imported into this table.

How it works: For each instrument in instTab.txt, this method first creates a list of all the experiments that used that instrument. It then loops through the file table. For each file listing, if the file is the default one, and its experiment id is in the list of experiments just created, the kindat is added to the list if its not already there. Data from all other Madrigal sites is then added via getMetadata. It then writes the metadata file in the form: 10, 1001 1002 1003, where the first column in the instrument code and the second column is a space delimited list of kindat codes.

Inputs: None.

Returns: None.

Affects: Writes file instKindatTab.txt in metadata directory

Exceptions: If unable to write instKindatTab.txt file.

Exceptions

madrigal.admin.MadrigalError('Unable to write: ' + str(filename), None)

Table of Contents

Table of Contents

Class: MadrigalInstrumentParameters **metadata**

MadrigalInstrumentParameters is an object that provides access to the metadata file that summarizes the parameters associated with each instrument.

This object provides access to all Madrigal instrument parameter information in t metadata file instParmTab.txt. The metadata file instParmTab.txt lists, for any given instrument, all the measured parameters found in all the data files in the database associated with that instrument.

This class also contains a method to rebuild the table instParmTab.txt by examini every data file in the database. This is presumably a slow process and should be done in the background.

Usage example:

```
import madrigal.metadata

import madrigal.admin

try:

    test = madrigal.metadata.MadrigalInstrumentParameters

    print test.getParameters(30)

except madrigal.admin.MadrigalError, e:

    print e.getExceptionStr()
```

Non-standard Python modules used: None

MadrigalError exception thrown if:

1. MadrigalMetadata fails to open or parse metadata file

Change history:

Written by Bill Rideout Jul. 17, 2002

Methods

__init__
getParameters
rebuildInstParmTable

__init__

```
__init__ (
```

```
        self,
        madDB=None,
        initFile=None,
        )
```

__init__ initializes MadrigalInstrumentParameters by reading from instParmTab.txt (or initFile).

Inputs: Existing MadrigalDB object, by default = None.

String representing the full path to the metadata file. Default is None, in which case file read is MadrigalDB.getMetadataDir()/instParmTab.txt.

Returns: void

Affects: Initializes all the class member variables.

Exceptions: MadrigalError thrown by MadrigalMetadata if file not opened or parsed successfully. Note that the instParmTab.txt file was new with the release of the madrigal python api, and thi function will throw an error if file not there.

getParameters

```
getParameters ( self,  kinst )
```

getParameters returns a list of parameters in mnemonic form (strings or unknown integers as strings) that matches kinst argument, or None if not found or blank.

Inputs: kinst integer to get parameters. If 0, get parameters from all instruments.

Returns: a list of mnemonic strings or unknown integer strings, o None if kinst not found or blank.

Affects: None

Exceptions: if error in metadata file

Exceptions

madrigal.admin.MadrigalError('Error in instTab.txt parsing metadata row: ' + str(inst), traceback.format_exception(sys.exc_info() [0],

MadrigalInstrumentParameters is an object that providesaccess to the metadata file that summarizes thepar

sys.exc_info() [1], sys.exc_info() [2]))

rebuildInstParmTable

```
rebuildInstParmTable ( self,  completeRebuildFlag=0
```

rebuildInstParmTable rebuilds the instParmTab.txt metadata file.

The table instParmTab.txt is a listing of every measured parameter found in any data file for a given instrument. It now will also import data from other Madrigal sites instParmTab.txt files. Since these files are constantly updated, this table needs to be updated on a regular basis. This methods works in one of two ways, depending on the value of completeRebuildFlag and whether a file called instParmLastUpdate.txt exists in the metadata directory.

If completeRebuildFlag = 1 or instParmLastUpdate.txt does not exist, the method rebuildInstParmTable loops through each instrument in the instTab.txt. For each instrument, it loops through every data file associated with that instrument. For every data file, it gets the list of parameters in that file, and adds them the list for that instrument if they are unique. Since this process involves every file in the database, it may take a great deal of tim and should be run in the background.

If completeRebuildFlag = 0 and instParmLastUpdate.txt does exist, the method rebuildInstParmTable first stores all the existin parameters from the instParmTab.txt. It reads the date of the last update from instParmLastUpdate.txt, and only reads data files newer than that date that are associated with each instrument. Fo every new data file, it gets the list of parameters in that file, and adds them to the list for that instrument if they are unique. This makes rebuildInstParmTable faster, but possibly keeps invalid parameters if experiments are ever deleted.

Finally, the instParmTab.txt file of every other site is obtained v getMetadata, and those parameters are also added.

Inputs: completeRebuildFlag: if 0 (the default), only add parameters from new files, where new means newer than the dat in the instParmLastUpdate.txt file (stored as a float). If 1, rebuil the table completely. This will eliminate any parameters no long found in files, but will take longer. If no instParmLastUpdate.txt file is found, the table is always rebuilt completely.

Returns: None.

getParameters returns a list of parameters inmnemonic form (strings or unknown integersas strings) that ma

Affects: Writes file instParmTab.txt in metadata directory

Exceptions: If unable to write instParmTab.txt file or the instParmLastUpdate.txt file.

Exceptions	
	madrigal.admin.MadrigalError('No data found for: ' + str(filename), None) madrigal.admin.MadrigalError('Unable to write ' + self.__madDB.getMetadataDir() + '/instParmLastUpdate.txt', None) madrigal.admin.MadrigalError('Unable to write ' + str(filename), None)

Table of Contents

This document was automatically generated on Fri Dec 30 08:58:50 2005 by HappyDoc version r1_5

Table of Contents

Class: MadrigalKindat **metadata.py**

MadrigalKindat is an object that provides access to Madrigal kind of data info from the metadata.

This object provides access to all Madrigal kind of data information in the metadata file typeTab.txt.

Usage example:

```
import madrigal.metadata
import madrigal.admin

try:

    test = madrigal.metadata.MadrigalKindat()

    print test.getKindatDescription(3001)

except madrigal.admin.MadrigalError, e:

    print e.getExceptionStr()
```

Non-standard Python modules used: None

MadrigalError exception thrown if:

1. MadrigalMetadata fails to open or parse metadata file
2. Columns expected to be ints or floats cannot be converted

Change history:

Written by Bill Rideout Nov. 9, 2001

Methods

__init__
getKindatDescription
getKindatList

__init__

```
__init__ (
        self,
        madDB=None,
        initFile=None,
        )
```

__init__ initializes MadrigalKindat by reading from typeTab.txt (or initFile).

Inputs: Existing MadrigalDB object, by default = None.

String representing the full path to the metadata file. Default is None, in which case file read is MadrigalDB.getMetadataDir()/typeTab.txt.

Returns: void

Affects: Initializes all the class member variables.

Exceptions: MadrigalError thrown by MadrigalMetadata if file not opened or parsed successfully.

getKindatDescription

```
getKindatDescription ( self,  code )
```

getKindatDescription returns the kindat description that matches code argument, or None if not found.

Inputs: code integer to get kindat description.

Returns: the kindat description that matches code argument, or None if not found.

Affects: None

Exceptions: MadrigalError if any item in row cannot be cast to correct format

Exceptions

madrigal.admin.MadrigalError('Error in typeTab.txt parsing metadata row: ' + str(type), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

getKindatList

```
getKindatList ( self )
```

getKindatList returns a list of all kindat descriptions and codes.

Inputs: None.

Returns: a list of all kindat descriptions and codes. Each item in the list is a tuple of the form (Kindat description (string), kindat code (integer)). Example item: (`INSCAL Basic Derived Parameters`, 3001)

Affects: None

Exceptions: MadrigalError if any item in row cannot be cast to correct format

Exceptions

madrigal.admin.MadrigalError('Error in typeTab.txt parsing metadata row: ' + str(kindat), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

Table of Contents

This document was automatically generated on Fri Dec 30 08:58:50 2005 by HappyDoc version r1_5

Table of Contents

Class:
MadrigalMetaFile

MadrigalMetaFile is an object that provides access to Madrigal file i metadata.

This object provides access to all Madrigal experiment information in the metadata file fileTab.t

Usage example:

```
import madrigal.metadata

import madrigal.admin

try:

    test = madrigal.metadata.MadrigalMetaFile()

    print test.getFileCount()

except madrigal.admin.MadrigalError, e:

    print e.getExceptionStr()
```

Non-standard Python modules used: None

MadrigalError exception thrown if:

1. MadrigalMetadata fails to open or parse metadata file
2. Columns expected to be ints or floats cannot be converted

Change history:

Written by Bill Rideout May. 7, 2002

Methods

__init__	getExpIdByPosition	getS
__str__	getFileCount	getS
deleteRowByFilename	getFilenameByPosition	setA
getAccessByPosition	getHasCatalogByFilename	setA
getAnalystByFilename	getHasCatalogByPosition	setA
getAnalystByPosition	getHasHeaderByFilename	setA
getAnalystEmailByFilename	getHasHeaderByPosition	setC
getAnalystEmailByPosition	getKindatByFilename	setE
getCategoryByFilename	getKindatByPosition	setH
getCategoryByPosition	getLine	setH
getExpIdByFilename	getMetadataSummaryByFilename	setK

__init__

```
__init__ (
        self,
        madDB=None,
        initFile=None,
```

```
        )
```

__init__ initializes MadrigalMetaFile by reading from fileTa

Inputs: Existing MadrigalDB object, by default = None.

String representing the full path to the metadata file. Default is None, in which cas
MadrigalDB.getMetadataDir()/fileTab.txt.

Returns: void

Affects: Initializes all the class member variables.

Exceptions: MadrigalError thrown by MadrigalMetadata if file not opened or pars

Exceptions	
	madrigal.admin.MadrigalError('Error in count of first row None)

__str__

```
__str__ ( self )
```

return possibly modified file as a string in same format as writeMetadata

deleteRowByFilename

```
deleteRowByFilename ( self,  filename )
```

deleteRowByFilename deletes a row with a given filename

Inputs: filename - name of file to search for.

Returns: None.

Affects: Removes item from self.__fileList if filename found

Exceptions: Thrown if filename not found.

Exceptions	
	madrigal.admin.MadrigalError('Could not delete file ' + fi self.__filename, None)

getAccessByPosition

```
getAccessByPosition ( self,  position=0 )
```

getAccessByPosition returns the access (0=public, 1=priv given position.

Inputs: position of file in list (first position is zero). Defaults to first.

Returns: the access level (0=public, 1=private) of the file at given position as an i position >= number of experiments.

Affects: None

Exceptions: Thrown if access column cannot be parsed into an integer

Exceptions

madrigal.admin.MadrigalError('Error in fileTab.txt parsing position), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [2]))

getAnalystByFilename

```
getAnalystByFilename ( self,  filename )
```

getAnalystByFilename returns file analyst name if there, N

Inputs: filename - name of file to search for.

Returns: file analyst name if there, None otherwise. Since not all fileTab.txt files r returns None if the column does not exist.

Affects: None

Exceptions: None

getAnalystByPosition

```
getAnalystByPosition ( self,  position=0 )
```

getAnalystByPosition returns file analyst name if there, N

Inputs: position of file in list (first position is zero). Defaults to first.

Returns: file analyst name if there, None otherwise. Since not all fileTab.txt files r returns None if the column does not exist. Returns None if position >= number of

Affects: None

Exceptions: None

getAnalystEmailByFilename

```
getAnalystEmailByFilename ( self,  filename )
```

getAnalystEmailByFilename returns file analyst email if th otherwise.

Inputs: filename - name of file to search for.

Returns: file analyst email if there, None otherwise. Since not all fileTab.txt files r returns None if the column does not exist.

Affects: None

Exceptions: None

getAnalystEmailByPosition

```
getAnalystEmailByPosition ( self,  position=0 )
```

getAnalystEmailByPosition returns file analyst email if the

Inputs: position of file in list (first position is zero). Defaults to first.

Returns: file analyst email if there, None otherwise. Since not all fileTab.txt files r returns None if the column does not exist. Returns None if position >= number of

Affects: None

Exceptions: None

getCategoryByFilename

```
getCategoryByFilename ( self,  filename )
```

getCategoryByFilename returns the category (eg., 1=defau 3=history, 4=real-time) of the file with the given filename.

Inputs: filename - name of file to search for.

Returns: the category (eg., 1=default, 2=variant, 3=history, 4=real-time) of the firs filename. Since filename may not be unique (although it usually is), the first matc matches found, returns None.

Affects: None

Exceptions: Thrown if category column cannot be parsed into an integer

Exceptions

madrigal.admin.MadrigalError('Error in fileTab.txt parsing position), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [2]))

getCategoryByPosition

```
getCategoryByPosition ( self,  position=0 )
```

getCategoryByPosition returns the category (eg., 1=defau 3=history, 4=real-time) of the file at given position.

Inputs: position of file in list (first position is zero). Defaults to first.

Returns: the category (eg., 1=default, 2=variant, 3=history, 4=real-time) of the file integer. Returns None if position >= number of experiments.

Affects: None

Exceptions: Thrown if category column cannot be parsed into an integer

Exceptions

madrigal.admin.MadrigalError('Error in fileTab.txt parsing position), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [2]))

getExpIdByFilename

```
getExpIdByFilename ( self,  filename )
```

getExpIdByFilename returns the first experiment id (integ filename.

Inputs: filename - name of file to search for.

Returns: the experiment id (integer) of the first row found with the given filename unique (although it usually is), the first match found is used. If no matches found,

Affects: None

Exceptions: Thrown if exp id cannot be parsed into an integer

Exceptions

madrigal.admin.MadrigalError('Error in fileTab.txt parsing position), traceback.format_exception(sys.exc_info() [0],

	sys.exc_info() [2]))

getExpIdByPosition

```
getExpIdByPosition ( self,  position=0 )
```

getExpIdByPosition returns the experiment id (integer) of position.

Inputs: position of file in list (first position is zero). Defaults to first.

Returns: the experiment id (integer) of the file at given position as an integer. Retu number of experiments.

Affects: None

Exceptions: Thrown if kinst exp id cannot be parsed into an integer

Exceptions	
	madrigal.admin.MadrigalError('Error in fileTab.txt parsing position), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [2]))

getFileCount

```
getFileCount ( self )
```

getFileCount returns the number of files (rows) in the met

Inputs: None

Returns: the number of files (rows) in the metadata file.

Affects: None

Exceptions: None

getFilenameByPosition

```
getFilenameByPosition ( self,  position=0 )
```

getFilenameByPosition returns the filename of the file at g

Inputs: position of file in list (first position is zero). Defaults to first.

Returns: the filename of the file at given position as a string. Returns None if posi experiments.

Affects: None

Exceptions: None

getHasCatalogByFilename

```
getHasCatalogByFilename ( self,  filename )
```

getHasCatalogByFilename returns true if the file with the catalog records, False otherwise.

Inputs: filename - name of file to search for.

Returns: true if the file with the given name has any catalog records, False otherw found

Affects: None

Exceptions: None.

getHasCatalogByPosition

```
getHasCatalogByPosition ( self,  position=0 )
```

getHasCatalogByPosition returns True if the file at given catalog records, False otherwise.

Inputs: position of file in list (first position is zero). Defaults to first.

Returns: True if the file at given position has any catalog records, False otherwise number of experiments.

Affects: None

Exceptions: None

getHasHeaderByFilename

```
getHasHeaderByFilename ( self,  filename )
```

getHasHeaderByFilename returns true if the file with the header records, False otherwise.

Inputs: filename - name of file to search for.

Returns: true if the file with the given name has any header records, False otherwi found

Affects: None

Exceptions: Thrown if filename not found.

getHasHeaderByPosition

```
getHasHeaderByPosition ( self,  position=0 )
```

getHasHeaderByPosition returns True if the file at given p header records, False otherwise.

Inputs: position of file in list (first position is zero). Defaults to first.

Returns: True if the file at given position has any header records, False otherwise number of experiments.

Affects: None

Exceptions: None

getKindatByFilename

```
getKindatByFilename ( self,  filename )
```

getKindatByFilename returns the first kindat (integer) wit

Inputs: filename - name of file to search for.

Returns: the kindat (integer) of the first row found with the given filename. Since (although it usually is), the first match found is used. If no matches found, returns

Affects: None

Exceptions: Thrown if kindat cannot be parsed into an integer

Exceptions	
	madrigal.admin.MadrigalError('Error in fileTab.txt parsing position), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [2]))

getKindatByPosition

```
getKindatByPosition ( self,  position=0 )
```

getKindatByPosition returns the kindat (kind of data code position.

Inputs: position of file in list (first position is zero). Defaults to first.

Returns: the kinst (kind of instrument code) of the file at given position as an integ
>= number of experiments.

Affects: None

Exceptions: Thrown if kinst column cannot be parsed into an integer

Exceptions	
	madrigal.admin.MadrigalError('Error in fileTab.txt parsing position), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [2]))

getLine

```
getLine ( self,  position )
```

getLine returns the line at a given position. Returns None of lines.

Inputs: position - position in file. First line = 0

getMetadataSummaryByFilename

```
getMetadataSummaryByFilename (
        self,
        filename,
        madExpObj=None,
        madInstObj=None,
        madKindatObj=None,
        )
```

getMetadataSummaryByFilename returns a string summa metadata given a filename.

Inputs: filename - name of file to search for.

The next three inputs are other metadata objects. If these objects already exist, per by passing them in rather than recreating them. If they do not exist, they will be cr

madExpObj - a MadrigalExperiment object to get experiment metadata from. If N MadrigalExperiment object is created.

madInstObj - a MadrigalInstrument object to get experiment metadata from. If No MadrigalInstrument object is created.

madKindatObj - a MadrigalKindat object to get experiment metadata from. If Nor MadrigalKindat object is created.

Returns: A string summarizing the metadata about the file. The format is:

```
Start Date and Time: 01/06/1997  14:07
End Date and Time:   01/10/1997  23:45
Instrument: Millstone Hill Incoherent Scatter Rada
Experiment name: World Day - Mesosphere/Lower-Ther
Kind of data: INSCAL (8.0) Basic Derived Parameter
```

Affects: None

Exceptions: Thrown if any parsing error in metadata.

getStatusByFilename

```
getStatusByFilename ( self,  filename )
```

getStatusByFilename returns the status description of the name.

Inputs: filename - name of file to search for.

Returns: the status description of the file with the given name. Returns none if nam

Affects: None

Exceptions: Thrown if filename not found.

getStatusByPosition

```
getStatusByPosition ( self,  position=0 )
```

getStatusByPosition returns the status description of the position.

Inputs: position of file in list (first position is zero). Defaults to first.

Returns: the status description of the file at given position as a string. Returns Non experiments.

Affects: None

Exceptions: None

setAccess

```
setAccess ( self,  accessType )
```

setAccess sets the access column to all 0's (public) or all

Inputs: accessType - either 0 to set to public access, or 1 to set to private access.

Returns: None.

Affects: Overwrite fileTab.txt file with access column set to all 0's (public) or all

Exceptions: Thrown if file cannot be written, if accessType is not 0 or 1

Exceptions

madrigal.admin.MadrigalError('MadrigalMetaFile.setAcce
accessType) + ', must be either 0 or 1', None)

setAccessByPosition

```
setAccessByPosition (
        self,
        position,
        access,
        )
```

setAccessByPosition sets the value of access for the file position.

Inputs:

position - position of file in list (first position is zero).

access - 0 of False for public, 1 or True for private

Returns: None.

Affects: None

Exceptions: If position beyond length.

Exceptions

ValueError, 'Illegal value for access in setAccessByPositio
ValueError, 'setAccessByPosition called for position %i be
len(self.__fileList))

setAnalystByPosition

```
setAnalystByPosition (
        self,
        position,
        analyst,
        )
```

setAnalystByPosition sets the file analyst name for the gi

Inputs:

position - position of file in list (first position is zero).

analyst - name of file analyst

Returns: None.

Affects: None

Exceptions: If position beyond length.

Exceptions

ValueError, 'setAnalystByPosition called for position %i b
%(position, len(self.__fileList))
madrigal.admin.MadrigalError('Error in setAnalystByPosit
str(position), analyst), [traceback.format_exc()])

setAnalystEmailByPosition

```
setAnalystEmailByPosition (
        self,
        position,
        analystEmail,
        )
```

setAnalystEmailByPosition sets the file analyst email for t

Inputs:

position - position of file in list (first position is zero).

analystEmail - email of file analyst

Returns: None.

Affects: None

Exceptions: If position beyond length.

Exceptions

ValueError, 'setAnalystEmailByPosition called for position
%(position, len(self.__fileList))
madrigal.admin.MadrigalError('Error in setAnalystEmailB
%(str(position, analystEmail), [traceback.format_exc()])

setCategoryByPosition

```
setCategoryByPosition (
        self,
        position,
        category,
        )
```

setCategoryByPosition sets the value of category for the position.

Inputs:

position - position of file in list (first position is zero).

category - 1=default, 2=variant, 3=history, 4=real-time

Returns: None.

Affects: None

Exceptions: If position beyond length.

Exceptions	
	ValueError, 'Illegal value for category in setCategoryByPo) ValueError, 'setCategoryByPosition called for position %i %(position, len(self.__fileList))

setExpIdByPosition

```
setExpIdByPosition (
        self,
        position,
        expId,
        )
```

setExpIdByPosition sets the experiment id of the file at gi

Inputs:

position - position of file in list (first position is zero).

expId - the new experiment id to use

Returns: None.

Affects: sets the experiment id of the file at given position

Exceptions: MadrigalError if any item in row cannot be cast to correct format, or

Exceptions

madrigal.admin.MadrigalError('Error in setExpIdByPositio
position), str(expId)), traceback.format_exception(sys.ex
sys.exc_info() [1], sys.exc_info() [2]))

setHasCatalogByPosition

```
setHasCatalogByPosition (
        self,
        position,
        hasCatalog,
        )
```

setHasCatalogByPosition sets the value of hasCatalog fo position.

Inputs:

position - position of file in list (first position is zero).

hasCatalog - 1 or True for yes, 0 or False for no

Returns: None.

Affects: None

Exceptions: If position beyond length.

Exceptions

ValueError, 'Illegal value for hasCatalog in setHasCatalog
hasCatalog))
ValueError, 'setHasCatalogByPosition called for position
%(position, len(self.__fileList))

setHasHeaderByPosition

```
setHasHeaderByPosition (
        self,
        position,
        hasHeader,
        )
```

setHasHeaderByPosition sets the value of hasHeader for position.

Inputs:

position - position of file in list (first position is zero).

hasHeader - 1 or True for yes, 0 or False for no

Returns: None.

Affects: None

Exceptions: If position beyond length.

Exceptions

ValueError, 'Illegal value for hasHeader in setHasHeaderB
hasHeader))
ValueError, 'setHasHeaderByPosition called for position %
%(position, len(self.__fileList))

setKindatByPosition

```
setKindatByPosition (
        self,
        position,
        kindat,
        )
```

setKindatByPosition sets the value of kindat for the file at

Inputs:

position - position of file in list (first position is zero).

kindat - integer kindat value

Returns: None.

Affects: None

Exceptions: If position beyond length.

Exceptions

ValueError, 'setKindatByPosition called for position %i be
len(self.__fileList))

setStatusByPosition

```
setStatusByPosition (
        self,
        position,
        status,
        )
```

setStatusByPosition sets the value of status string for the position.

Inputs:

position - position of file in list (first position is zero).

status - string describing status

Returns: None.

Affects: None

Exceptions: If position beyond length.

Exceptions

ValueError, 'Illegal value for status in setStatusByPosition
ValueError, 'setStatusByPosition called for position %i bey
len(self.__fileList))
ValueError, 'status string in fileTab.txt cannot contain a co

writeMetadata

```
writeMetadata ( self,  newFullPath=None )
```

writeMetadata writes a new version of the fileTab.txt file.

Inputs: newFullPath: a new path to write the fileTab.txt file to, if not the same as t
opened. Defaults to None, which overwrites metadata file that was read from.

Returns: None.

Affects: Writes updated version of metadata file.

Exceptions: If unable to write file

Exceptions

madrigal.admin.MadrigalError("Unable to write metadata
traceback.format_exception(sys.exc_info() [0], sys.exc_in
]))

Table of Contents

Table of Contents

Class: MadrigalMetadata **metadata.py**

MadrigalMetadata is a private class that parses a Madrigal metadata file.

This private class is used by all classes that need to parse a Madrigal metadata file. If the class is called with the name of the metadata file only, the metadata file is assumed to be at $MAD_ROOT/metadata. If a full path name is given that includes a directory separator, then that is used instead. The getList method returns a list with one item for each line in the file. That item for each line is simply a list of strings found in the line. The following is an example metadata file and the list the method getList would return.

Metadata file example:

```
Tom, Dick,,Harry
,,Joe,
Sally, Jane,Joe, Dick
```

The list returned by getList example:

```
[['Tom', 'Dick', '', 'Harry'],
['', '', 'Joe', ''],
['Sally', 'Jane', 'Joe', 'Dick']]
```

Non-standard Python modules used: None

MadrigalError exception thrown if:

1. Unable to open metadata file.
2. All lines in metadata file do not have same number of items

Change history:

Written by Bill Rideout Nov. 8, 2001

Methods

__getFullPath
__init__
__parseLine
getList
toString

__getFullPath

```
__getFullPath ( self,  filename )
```

getFullPath returns the full path name of the metafile.

Inputs: filename passed in as argument

Returns: The full path name of the metafile. If the filename argument already is a full path, filename is returned unchanged. If the filename is simply the name of a metadata file, then $MAD_ROOT/metadata is appended

Affects: Nothing

Exceptions: None

__init__

```
__init__ (
        self,
        metadataFileName,
        madDB=None,
        allowedLenList=None,
        )
```

__init__ initializes the MadrigalCategoryList by reading from __privateList.

Inputs: String metadataFileName - if not a full path, then MadrigalDB.getMetadataDir is included.

Existing MadrigalDB object, by default = None.

allowedLenList - a list of allowed lengths (integers) for data lines. If None (the default), then the rule is that all lengths must be the same.

Returns: void

Affects: Initializes private member variable __categoryList.

Exceptions: None.

Exceptions	
	madrigal.admin.MadrigalError("Unable to open metadata file " + self.__fileName, traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

__parseLine

```
__parseLine ( self,  line )
```

__parseLine adds a list of items in line to __fileList.

Inputs: Line of file to parse (String).

Returns: void

Affects: Adds a list of items in line to self.__fileList.

Exceptions: None.

Exceptions	
	madrigal.admin.MadrigalError("Wrong number of items found in metadata file " + self.__fileName + " at line " + str(len(self.__fileList)), [traceback.format_exc()])

getList

```
getList ( self )
```

getList returns the list of lists of items in each line in the metafile.

Inputs: None

Returns: The list of lists of items in each line in the metafile. That is, each item in the returned list is itself a list, representing a single line in the metafile. That single line's list is the list of items in that line.

Affects: Nothing

Exceptions: None

toString

```
toString ( self )
```

toString returns a simple string representation of a MadrigalMetadata object.

Inputs: None

Returns: String describing a simple representation of a __MadrigalMetadat object.

Affects: Nothing

Exceptions: None

Table of Contents

Table of Contents

Class: MadrigalSite **metadata.py**

MadrigalSite is an object that provides access to Madrigal site info from the metadata.

This object provides access to all Madrigal site information in the metadata file siteTab.txt.

Usage example:

```
import madrigal.metadata
import madrigal.admin

try:

    siteObj = madrigal.metadata.MadrigalSite()

    print siteObj.getSiteName(1)

except madrigal.admin.MadrigalError, e:

    print e.getExceptionStr()
```

Non-standard Python modules used: None

MadrigalError exception thrown if:

1. MadrigalMetadata fails to open or parse metadata file
2. Columns expected to be ints or floats cannot be converted

Change history:

Written by Bill Rideout Nov. 9, 2001

Methods

__init__
getSiteDocRoot

__init__

```
__init__ (
        self,
        madDB=None,
        initFile=None,
        )
```

__init__ initializes MadrigalSite by reading from siteTab.txt (or initFile).

Inputs: Existing MadrigalDB object, by default = None.

String representing the full path to the metadata file. Default is None, in which case file read is MadrigalDB.getMetadataDir()/siteTab.txt.

Returns: void

Affects: Initializes all the class member variables.

Exceptions: MadrigalError thrown by MadrigalMetadata if file not opened or parsed successfully.

getSiteDocRoot

```
getSiteDocRoot ( self,  siteID )
```

getSiteDocRoot returns the relative document root (e.g. madrigal) that matches siteID argument, or None if not found.

Inputs: siteID integer to get document root path.

Returns: the document root path that matches siteID argument, or None if not found.

Affects: None

Exceptions: MadrigalError if any item in row cannot be cast to correct format

Exceptions

madrigal.admin.MadrigalError('Error parsing metadata row in siteTab.txt: ' + str(site), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

getSiteEmail

```
getSiteEmail ( self,  siteID )
```

getSiteEmail returns the site email address that matches siteID argument, or None if not found.

Inputs: siteID integer to get Site email address.

Returns: the site email address that matches siteID argument, or None if not found. To list multiple email addresses in this field separate them with semicolons (since commas are delimiters). getSiteEmail will automatically replace semicolons with commas, as required by multiple email addresses.

Affects: None

Exceptions: MadrigalError if any item in row cannot be cast to correct format

Exceptions

madrigal.admin.MadrigalError('Error parsing metadata row in siteTab.txt: ' + str(site), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

getSiteList

```
getSiteList ( self )
```

getSiteList returns a list of all site ids and names.

Inputs: None.

Returns: a list of all site ids and names. Each item in the list is a tuple of the form (Site id (integer), site name (string)). Example item: (1,'Millstone')

Affects: None

Exceptions: MadrigalError if any item in row cannot be cast to correct format

Exceptions

madrigal.admin.MadrigalError('Error in siteTab.txt parsing metadata row: ' + str(inst), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

getSiteName

```
getSiteName ( self,  siteID )
```

getSiteName returns the site name that matches siteID argument, or None if not found.

Inputs: siteID integer to get SiteName.

Returns: the site name that matches siteID argument, or None if not found.

Affects: None

Exceptions: MadrigalError if any item in row cannot be cast to correct format

Exceptions

madrigal.admin.MadrigalError('Error parsing metadata row in siteTab.txt: ' + str(site), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

getSiteRelativeCGI

```
getSiteRelativeCGI ( self,  siteID )
```

getSiteRelativeCGI returns the relative cgi path (e.g.cgi-bin/madrigal) that matches siteID argument, or None if not found.

Inputs: siteID integer to get relative cgi path.

Returns: the relative cgi path that matches siteID argument, or None if not found.

Affects: None

Exceptions: MadrigalError if any item in row cannot be cast to correct format

Exceptions	
	madrigal.admin.MadrigalError('Error parsing metadata row in siteTab.txt: ' + str(site), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

getSiteServer

```
getSiteServer ( self,  siteID )
```

getSiteServer returns the site server (e.g., www.haystack.mit.edu) that matches siteID argument, or None if not found.

Inputs: siteID integer to get site server.

Returns: the site server that matches siteID argument, or None if not found.

Affects: None

Exceptions: MadrigalError if any item in row cannot be cast to correct format

Exceptions	
	madrigal.admin.MadrigalError('Error parsing metadata row in siteTab.txt: ' + str(site), traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

Table of Contents

This document was automatically generated on Fri May 30 14:39:57 2008 by HappyDoc version r1_5

Table of Contents

Module: openmadrigal	openmadrigal.py

The openmadrigal module provides access to all OpenMadrigal installations via http and to OpenMadrigal Subversion.

$Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $

Classes	
	OpenMadrigal

OpenMadrigal is an object that provides access to all Open Madrigal installations via http and to OpenMadrigal Subversion.

Table of Contents

This document was automatically generated on Thu Oct 20 16:51:50 2011 by HappyDoc version r1_5

Table of Contents

Class: OpenMadrigal **openmadrigal.py**

OpenMadrigal is an object that provides access to all Open Madrigal installations via http and to OpenMadrigal Subversion.

Usage example:

```
import madrigal.openmadrigal

try:

    test =  madrigal.openmadrigal.OpenMadrigal()

    # get the metadata file fileTab.txt from Madrigal site with id = 1

    test.getFileTab(1)

except madrigal.admin.MadrigalError, e:

    print e.getExceptionStr()
```

Non-standard Python modules used: None

Change history:

Written by Bill Rideout Aug. 14, 2002

Methods

__getMetadata
__init__
getAllRevisionNumbers
getCvsVersion
getDataMetadata
getExpMetadata
getFileMetadata
getInstMetadata
getInstTypeMetadata
getLatestCvsVersion
getLatestSubversionVersion
getMadCatMetadata
getMetadataFromOpenMadrigal
getParcodsMetadata
getSiteMetadata
getSubversionVersion
getTypeMetadata

getSiteServer returns the site server (e.g.,www.haystack.mit.edu) that matches siteIDargument, or None if n

__getMetadata

```
__getMetadata (
        self,
        siteId,
        metadataType,
        )
```

__getMetadata is a private helper function called to get metadata files via the web.

Inputs: siteId - integer identifying Madrigal site. metadataType - constant defined by getMetadata cgi script

Returns: The desired metadata file as a string, or None if not successful

Affects: None.

Exceptions: None.

__init__

```
__init__ ( self,  madDB=None )
```

__init__ initializes OpenMadrigal by setting or creating a MadrigalDB object.

Inputs: Existing MadrigalDB object, by default = None.

Returns: void

Affects: Initializes all the class member variables.

Exceptions: None.

getAllRevisionNumbers

```
getAllRevisionNumbers ( self,  fullPath )
```

getAllRevisionNumbers a list of all revision numbers for a given file in Subversion in order from latest to earliest.

Inputs: fullPath - full path to the Madrigal file in Subversion relative to trunk (example: `madroot/metadata/siteTab.txt`)

Returns: a list of all revision numbers for a given file in Subversion in order from latest to earliest. Empty list if file not found

Affects: Nothing

Exceptions: None

getCvsVersion

```
getCvsVersion (
        self,
        fullPath,
        revision,
        )
```

getCvsVersion returns the Subversion version of the file given by fullPath and revision.

CVS is no longer the Madrigal repository. Equivalent to getSubversionVersion.

Inputs:

fullPath - full path to the Madrigal file in Subversion (example: `madroot/metadata/siteTab.txt`)

revision - revision string (example `1445`)

Returns: the Subversion version of the file given by fullPath and revision, or None if not found

Affects: Nothing

Exceptions: None

getDataMetadata

```
getDataMetadata ( self,  siteId )
```

getDataMetadata returns the dataTab.txt file from siteId as a string.

This file is deprecated with Madrigal 2.5 and may not exist.

Inputs: None

Returns: the dataTab.txt file from siteId as a string, or None if not found

Affects: Nothing

Exceptions: None

getExpMetadata

```
getExpMetadata ( self,  siteId )
```

getExpMetadata returns the expTab.txt file from siteId as a string.

Inputs: None

Returns: the expTab.txt file from siteId as a string, or None if not found

Affects: Nothing

Exceptions: None

getFileMetadata

```
getFileMetadata ( self,  siteId )
```

getFileMetadata returns the fileTab.txt file from siteId as a string.

Inputs: None

Returns: the fileTab.txt file from siteId as a string, or None if not found

Affects: Nothing

Exceptions: None

getInstMetadata

```
getInstMetadata ( self,  siteId )
```

getInstMetadata returns the instTab.txt file from siteId as a string.

Inputs: None

Returns: the instTab.txt file from siteId as a string, or None if not found

Affects: Nothing

Exceptions: None

getInstTypeMetadata

```
getInstTypeMetadata ( self,  siteId )
```

getInstTypeMetadata returns the instType.txt file from siteId as a string.

Inputs: None

Returns: the instType.txt file from siteId as a string, or None if not found

Affects: Nothing

Exceptions: None

getLatestCvsVersion

```
getLatestCvsVersion ( self,  fullPath )
```

getLatestCvsVersion returns the latest Subversion version of the file given by fullPath as a string.

CVS is no longer the Madrigal repository. Equivalent to getLatestSubversionVersion.

Inputs: fullPath - full path to the Madrigal file in Subversion relative to trunk (example: `madroot/metadata/siteTab.txt`)

Returns: the latest Subversion version of the file given by fullPath as a string, or None if not found

Affects: Nothing

Exceptions: None

getLatestSubversionVersion

```
getLatestSubversionVersion ( self,  fullPath )
```

getLatestSubversionVersion returns the latest Subversion version of the file given by fullPath as a string.

Inputs: fullPath - full path to the Madrigal file in Subversion relative to trunk (example: `madroot/metadata/siteTab.txt`)

Returns: the latest Subversion version of the file given by fullPath as a string, or None if not found

Affects: Nothing

Exceptions: None

Simply calls equivalent method getLatestCvsVersion

getMadCatMetadata

```
getMadCatMetadata ( self,  siteId )
```

getMadCatMetadata returns the madCatTab.txt file from siteId as a string.

Inputs: None

Returns: the madCatTab.txt file from siteId as a string, or None if not found

Affects: Nothing

Exceptions: None

getMetadataFromOpenMadrigal

```
getMetadataFromOpenMadrigal ( self,  filename )
```

getMetadataFromOpenMadrigal returns a metadata file from OpenMadrigal server as a string

Inputs:

filename - metadata file to download, relative to metadata

Returns: File contents as a string

Affects: Nothing

getParcodsMetadata

```
getParcodsMetadata ( self,  siteId )
```

getParcodsMetadata returns the parcods.tab file from siteId as a string.

Inputs: None

Returns: the parcods.tab file from siteId as a string, or None if not found

Affects: Nothing

Exceptions: None

getSiteMetadata

```
getSiteMetadata ( self,  siteId )
```

getSiteMetadata returns the siteTab.txt file from siteId as a string.

Inputs: None

Returns: the siteTab.txt file from siteId as a string, or None if not found

Affects: Nothing

Exceptions: None

getSubversionVersion

```
getSubversionVersion (
        self,
        fullPath,
        revision,
        )
```

getSubversionVersion returns the Subversion version of the file given by fullPath and revision.

Inputs:

fullPath - full path to the Madrigal file in Subversion (example: `madroot/metadata/siteTab.txt`)

revision - revision string (example `1445`)

Returns: the Subversion version of the file given by fullPath and revision, or None if not found

Affects: Nothing

Exceptions: None

Simply calls equivalent method getCvsVersion

getTypeMetadata

```
getTypeMetadata ( self,  siteId )
```

getTypeMetadata returns the typeTab.txt file from siteId as a string.

Inputs: None

Returns: the typeTab.txt file from siteId as a string, or None if not found

Affects: Nothing

Exceptions: None

Table of Contents

This document was automatically generated on Thu Oct 20 16:51:50 2011 by HappyDoc version r1_5

Table of Contents

Module: __bgReport **ui/__bgReport.py**

__bgReport is the module that runs in a separate process to create a background report.

__bgReport contains a number of constants that can be modified to change the rules for generating background reports. These constants are:

tempDir - the name of the temporary directory on the web server where the reports are saved. This directory is located directly beneath the html directory defined in madrigal.cfg as MAD + SERVERDOCABS.

numDays - the number of days a temporary report is kept on the web server before being deleted by this script when it runs. This script will delete all reports older than numDays each time it runs.

$Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $

Functions

__dropLock
__getFilterStrList
__getLock

__dropLock

```
__dropLock ( filename )
```

__dropLock is a private helper function that drops exclusive access to filename via a locking file.

Inputs: filename = the file that exclusive access is required to.

Returns: None

Affects: Removes file filename + .LCK as a lock mechanism

Exceptions: None.

__getFilterStrList

```
__getFilterStrList ( filterList )
```

__getFilterStrList is a private helper function that creates 6 list describing filters needed by _Madrec.getIsprintReport.

Inputs: filterList - list of filter strings as passed in by report.py (See report.py for description).

Returns: List of Six lists as follows:

1. List giving filter types, len = # filters, items are 1,'*','/','+', or -
2. List with mnemonics of filter parameter 1 (items = # filters)
3. List with mnemonics of filter parameter 2 (items = # filters) (may be None)
4. List with number of ranges per filter (items = # filters)
5. List with doubles of filter lower limits (items = sum of number of ranges above)
6. List with doubles of filter upper limits (items = sum of number of ranges above)

Exceptions: None.

__getLock

```
__getLock ( filename )
```

__getLock is a private helper function that provides exclusive access to filename via a locking file.

Inputs: filename = the file that exclusive access is required to.

Returns: None

Affects: Writes file filename + .LCK as a lock mechanism

Exceptions: MadrigalError thrown if unable to write lock file

Notes: Will sleep for 1 second at a time, for a maximum of 10 seconds if the file is not modified. After each second, it will check for the lock file to be removed or modified. If it was modified, it resets the count to 0 sec and starts counting again. After _MaxSleep counts it then assumes lock file is orphaned and returns. Orphaned file will be removed when dropLock is called.

Exceptions	
	madrigal.admin.MadrigalError("Unable to open " + filename + ".LCK as locking file ", None)

Table of Contents

This document was automatically generated on Tue Mar 17 10:13:25 2009 by HappyDoc version r1_5

Table of Contents

Module: __init__ **ui/__init__.py**

Directory of all modules related to madrigal user interface.

Table of Contents

This document was automatically generated on Fri Dec 30 09:02:26 2005 by HappyDoc version r1_5

Table of Contents

Module: isprintExe **ui/isprintExe.py**

isprintExe is a private module designed to get output strings from the maddata engine in isprint format.

This module is not meant to be used directly by the user, and is only meant to be called from other classes in the madrigal python library.

$Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $

Classes	
MadrigalIsprintExe	MadrigalIsprintExe is a private class designed to get output strings from the maddata engine.

Table of Contents

This document was automatically generated on Fri Jul 13 09:36:49 2007 by HappyDoc version r1_5

Table of Contents

Class: MadrigalIsprintExe

__getLock is a private helper function thatprovides exclusive access to filename via alocking file.403

MadrigalIsprintExe is a private class designed to get output string engine.

MadrigalIsprintExe is a private class designed to get output strings using the maddata engine a postprocessing.

Non-standard Python modules used: None

Methods

__getLine
__getNeededParms
__init__
getModifiedIsprintString

__getLine

```
__getLine (
        self,
        filterList,
        lineNum,
        lineList,
        dataList,
        neededParms,
        acceptList,
        )
```

__getLine is a private function that adds a line of data to the data.

Inputs:

filterList - the list of filter strings to be applied

lineNum - number of present row

lineList - a list of line strings, one for each row

dataList - a list of lists of data elements (may be floats, or special strings such as

neededParms - a list parameters used in filters. Each item in list is a list containi representing column number

acceptList - a list containing accepted line so far

Returns: None.

Affects: Appends the appropriate line from lineList to acceptList if accepted.

Exceptions: None.

__getNeededParms

```
__getNeededParms (
        self,
        parmList,
        filterList,
        )
```

__getNeededParms is a private function that returns a lis filterList.

Inputs:

parmList - a list of mnemonic string parameters being displayed

filterList - the list of filter strings to be applied

Returns: a list which is a subset of parmList of all parameters found in filterList. two members, the string mnemonic and the position in the parmList where that p

Affects: None.

Exceptions: None.

__init__

```
__init__ ( self,  madDB=None )
```

__init__ initializes MadrigalIsprintExe by reading from M

Inputs: Existing MadrigalDB object, by default = None.

Returns: void

Affects: Initializes self.__metaDir.

Exceptions: None.

getModifiedIsprintString

```
getModifiedIsprintString (
        self,
        filename,
        parmList,
        filterList=None,
        recordListingStyle=0,
        )
```

getModifiedIsprintString extends the maddata engine by

getModifiedIsprintString is a function that expands the maddata engine by allow
logical expression involving any Madrigal parameters.

Inputs:

filename - full path to madrigal file.

parmList - a list of mnemonic or integer parameters.

filterList - a list of strings filtering the output. They must only contain parameter
the parmList. See example below for examples. Defaults to None.

recordListingStyle - If 0 (the default), show record summary and all data that me
summary of any record that contains any 2d data that meets all the filters. If 2, s
filters without record headers.

Returns: String containing report formatted in isprint fashion, or empty string if

Affects: None

Exceptions: If any item in filterList is not a valid logical expression.

Usage example:

```
import madrigal.ui.isprintExe

test = madrigal.ui.isprintExe.MadrigalIsprintExe()

filepath = os.environ.get('MADROOT') + '/experiments

parmList = ['range',  'ti', 'TN',  'kinst']

filterList = ['range &gt; 900 and ti &gt; 2000', 'ti

result = test.getModifiedIsprintString(filename, par

print result
```

Table of Contents

This document was automatically generated on Fri Dec 30 09:02:26 2005 by HappyDoc version r1_5

Table of Contents

Module: **ui/madrigalPlot.py**
madrigalPlot

madrigalPlot is the module that produces plots of Madrigal data.

Presently based on madplotlib: http://matplotlib.sourceforge.net/

$Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $

Functions

convertToAbsoluteTimeStr

```
convertToAbsoluteTimeStr (
        xticks,
        noTime=False,
        noDate=False,
        timezone=0,
        )
```

convertToAbsoluteTimeStr converts a list of strings containing seconds since 1/1/1950 to datetime string.

Input: xticks - a list of strings containing seconds since 1/1/1950

Returns: a list of strings formated as YYYY-MM-DD HH-MM-SS. If noTime, format as YYYY-MM-DD

defineHomeEnvVariable

```
defineHomeEnvVariable ()
```

defineHomeEnvVariable makes sure HOME env variable is defined, as required by matplotlib.

If not defined, sets HOME to MADROOT/metadata/userdata

getIsprintString

```
getIsprintString (
        filename,
        parms,
        filterStr,
        )
```

getIsprintString returns the output of isprint given inputs

Inputs: filename - Madrigal filename parms - comma-delimited parameters filterStr - arguments to pass to isprint cmd

get_vo_cmap

```
get_vo_cmap ()
```

get_vo_cmap is a function to return a colormap optimized to show sign changes in the middle of the range.

Classes

madHistogram	madHistogram is the class that produces Histogram plots of Madrigal
madIsrRecordSummary	madIsrRecordSummary is the class that produces a summary plot of a single ISR record.
madLineTimePlot	madLineTimePlot is the class the produces line plots of one or more parameters versus time.
madPcolorPlot	madPcolorPlot is the class that produces pcolor plots of x versus y with z intensity.
madPcolorScan	madPcolorScan is the class that produces pcolor scans.
madPcolorWedgeKmlScan	madPcolorWedgeKmlScan is the class that produces pcolor scan kml files where data is drawn as wedge shapes.
madPcolorWedgeScan	madPcolorWedgeScan is the class that produces pcolor scans where data is drawn as wedge shapes.
madScatterPlot	madScatterPlot is the class the produces two dimensional scatter plots of x versus y.
madXYScatterPlot	madXYScatterPlot is the class the produces XY scatter plots.
madXYwithErrorPlot	madXYwithErrorPlot is the class the produces two dimensional scatter plots of x versus y more its error.
scanPlotter	scanPlotter is the class that produces a series of scan plots for a single Madrigal file

Table of Contents

Class:

madHistogram

madHistogram is the class that produces Histogram plots of Madrigal Data.

Usage example:

```
obj = madHistogram(isprintText,
                   'Nel (log(m^-3)) - Millstone Hill - Oct. 30, 20
                   'Hours since midnight UT Oct. 30,2003'
                   'Altitude',
                   './isprint.png'
                   size = 'large')
```

Inputs:

isprintText - a string giving isprint output without headers. Any missing data should be written as `m` string that cannot be converted into a float. Only one column.

TitleStr - plot title(string) - should describe the plot being made

xLabelStr - Label for x axis

fullFilename - full path of file containing histogram plot to be saved. Extension must be .jpeg or .pr thrown

size - size of plot to be saved. Must be `small`,'wide', or `large`. Defaults to small

maxNumPoints - maximum number of points to be considered. Defaults to None, so all points in st

numBins - number of bins to be used to store the data. More bins means closer resolution in Histog

orientation - whether histogram shows horizontal or vertical. Must be written as `horizontal` or exception thrown--defaults to `vertical`

isNorm - whether histogram should be normalized or not. Default is 0, or False

isBottom - If true, sets the lowest passed value as zero

sdevs - number of standard deviations to take into account. When 0 is passed to it, this option is ign included in the distribution; otherwise, the range of values accepted is sdevs*(standard deviation of

Outputs:

A .png file is written to the fullFilename path given, resulting in a Histogram drawn by matplotlib

Change History:

Written by Brandon Scott Fines Aug. 1,2006 Written by Bill Rideout Aug. 1,2006

Methods

__filter_missing

```
__filter_missing ( self,  x )
```

__init__

```
__init__ (
        self,
        isprintText,
        titleStr,
        xLabelStr,
        fullFilename,
        size='small',
        maxNumPoints=None,
        numBins=30,
        Orientation='vertical',
        isNorm=0,
        isBottom=0,
        sdevs=0,
        )
```

__init__ writes a madHistogram string to file

Exceptions

ValueError, 'input text is not parseable'
ValueError, 'size must be "small","wide" or "large", not size))

__truncateIsprintStr

```
__truncateIsprintStr (
        self,
        isprintText,
        maxLines,
        )
```

__truncateIsprintStr truncates isprintText to have maxLines at most.

removeOutliers

```
removeOutliers (
        self,
        ndevs,
        values=[],
        )
```

Table of Contents

This document was automatically generated on Fri May 30 14:39:57 2008 by HappyDoc version r1_5

Table of Contents

Class: madIsrRecordSummary — **ui/madrigalPlot.py**

madIsrRecordSummary is the class that produces a summary plot of a single ISR record.

Methods

__init__

__init__

```
__init__ (
        self,
        TiData,
        TiError,
        TeData,
        TeError,
        VoData,
        VoError,
        NelData,
        NelError,
        isPopl,
        description,
        fullFilename,
        useRange=False,
        altResl=None,
        )
```

__init__ writes a madIsrRecordSummary to a file.

Inputs:

TiData - an Nx2 numeric array of Ti (col 0) and altitude in km (col 1)

TiError - an N length array of error in Ti

TeData - an Nx2 numeric array of Te (col 0) and altitude in km (col 1)

TeError - an N length array of error in Te

VoData - an Nx2 numeric array of Vo (col 0) and altitude in km (col 1)

VoError - an N length array of error in Vo

NelData - an Nx2 numeric array of (Popl or Nel in m^-3) (col 0) and altitude in km (col 1)

NelError - an 2xN length array of lower, upper error in Popl or Nel

isPopl - true if Nel contains Popl data, false if Nel data

description - text to put in fourth panel of plot - use carriage returns to separate lines

fullFilename - full filename to save output png file.

useRange - if False (the default), y axis = Altitude. If True, y axis = Range.

altResl - altitude resolution in km. If not None, show altitude resolution on each graph.

Returns: void

Affects: None

Table of Contents

This document was automatically generated on Fri May 30 14:39:57 2008 by HappyDoc version r1_5

Table of Contents

Class:	**ui/madrigalPlot.py**
madLineTimePlot	madLineTimePlot is the class the produces line plots of one or more parameters versus time.

Methods

__filter_missing

__init__ writes a madIsrRecordSummary to a file.

```
__filter_missing ( self,  x )
```

__init__

```
__init__ (
        self,
        isprintText,
        yParmList,
        titleStr,
        xLabelStr,
        yLabelStr,
        fullFilename,
        size='small',
        useAbsoluteTime=False,
        startTime=None,
        endTime=None,
        maxNumPoints=None,
        noTime=False,
        yMinimum=None,
        yMaximum=None,
        noDate=False,
        timezone=0,
        )
```

__init__ writes a madLineTimePlot plot to a file.

Inputs:

isprintText - a string giving isprint output without headers. First parameter must be UTH or UT1, depending on whether time scale should be relative to the experiment beginning (UTH) or absolute (UT1). The the following must be the parameters to be plotted. Any missing data should be written as "missing" or other string that cannot be converted to a float.

yParmList - a list of y parameters (strings). Length must == num columns in isprintText - 1

titleStr - plot title (string) - should describe parameter being plotted

xLabelStr - x label string

yLabelStr - ylabel string

fullFilename - full path of file containing pcolor plot to be saved. Extension must be jpeg or png, or exception thrown.

size - size of plot to save. Must be "small", "wide", or "large". Defaults to small.

useAbsoluteTime - if true, print time as YYYY-MM-DD HH:MM:SS. If false (default), print time as hour since beginning

of experiment (UTH). If useAbsoluteTime is true, first parameter in isprintText must be UT1, if false, it must be UTH.

startTime - start plot at given time. If useAbsoluteTime == True, then startTime must be in units of seconds since 1/1/1950. If useAbsoluteTime == False, then startTime must be in units of UTH (hours since midnight UT of first day of experiment). Default is None, which means start at lowest time found.

endTime - end plot at given time. If useAbsoluteTime == True, then endTime must be in units of seconds since 1/1/1950. If useAbsoluteTime == False, then endTime must be in units of UTH (hours since midnight UT of first day of experiment). Default is None, which means end at largest time found.

maxNumPoints - maximum number of points to plot. If not None, truncate isprintText if needed to have at most maxNumPoints lines.

noTime - if True and useAbsoluteTime True, then dates without times printed on x axis.

noDate - if True and useAbsoluteTime True, then times without dates printed on x axis.

yMinimum - set y minimum. If default=None, use data minimum

yMaximum - set y maximum. If default=None, use data maximum

timezone - The offset of the local (non-DST) timezone, in seconds west of UTC (negative in most of Western Europe, positive in the US, zero in the UK).

Returns: void

Affects: None

Exceptions

ValueError, 'No valid y data found'
ValueError, 'input text is not parseable'
ValueError, 'size must be "small", "wide", or "large", not %s' %(str(size))

__truncateIsprint

```
__truncateIsprint (
        self,
        isprintText,
        maxLines,
        )
```

__truncateIsprint truncates isprintText to have maxLines at most.

displayToScreen

```
displayToScreen ( self )
```

getFigureHandle

```
getFigureHandle ( self )
```

writeToFile

```
writeToFile ( self,  fullFilename )
```

Table of Contents

Table of Contents

Class: **ui/madriga**

madPcolorPlot

madPcolorPlot is the class that produces pcolor plots of x versus y w intensity.

Assumes the x axis is time.

Usage example:

```
obj = madPcolorPlot(isprintText,
                    'Nel (log(m^-3)) - Millstone Hill - Oct. 30, 2003 - Alt
                    'Hours since midnight UT Oct. 30, 2003',
                    'Altitude (km)',
                    './isprint.png',
                    size = 'large',
                    minColormap = 9,
                    maxColormap = 12,
                    smoothAltitude = False)
```

Non-standard Python modules used: matplotlib

Change history:

Written by Bill Rideout Mar. 31, 2005

Methods

__filter_missing
__getYIndex
__init__

__filter_missing

```
__filter_missing ( self,  x )
```

__getYIndex

```
__getYIndex ( self,  yvalue )
```

__getYIndex returns the correct index into the y dimen for a given y value.

Input: yvalue - value of y parameter

Returns: the correct index into the y dimension

Algorithm: if self.truncateAlt == False, simple use the dictionary self.yListD
Else loop through self.yListRanges and return the first greater than the reques
value

__init__

```
__init__ (
        self,
        isprintText,
        titleStr,
        xLabelStr,
        yLabelStr,
        fullFilename,
        size='small',
        minColormap=None,
        maxColormap=None,
        smoothAltitude=True,
        insertDataGap=5,
        useAbsoluteTime=False,
        startTime=None,
        endTime=None,
        sortTimeFlag=False,
        maxNumTimes=None,
        maxNumAlt=None,
        truncateIsprint=False,
        colorMap=matplotlib.cm.jet,
        yMinimum=None,
        yMaximum=None,
        altYTitle=None,
        altYLabels=None,
        noTime=False,
        noDate=False,
        timezone=0,
```

```
        background='w',
        )
```

__init__ writes a madPColorPlot to a file.

Inputs:

isprintText - a string giving isprint output without headers. First parameter m
UTH or UT1, depending on whether time scale should be relative to the expe
beginning (UTH) or absolute (UT1). The second must be gdalt, and third par
to be plotted. Any missing data should be written as "missing" or other string
cannot be converted to a float.

titleStr - plot title (string) - should describe parameter being plotted

xLabelStr - x label string

yLabelStr - ylabel string

fullFilename - full path of file containing pcolor plot to be saved. Extension m
jpeg or png, or exception thrown.

size - size of plot to save. Must be "small", "wide", or "large". Defaults to sm

minColormap - minimum parameter value (defaults to lowest parameter valu

maxColormap - maximum parameter value (defaults to highest parameter val
However, if both minColormap and maxColormap == None, autoscaling app

smoothAltitude - if True, extrapolate between existing data between altitudes
in missing data; if False, leave missing

insertDataGap - this parameter sets the threshold for inserting a data gap. The
intervals being plotted are ordered, and the time gap larger than 90% of the re
determined. Any time interval more than insertDataGap times bigger is then
considered missing data. Defaults to five. If None, no gaps are ever inserted.
data with close to uniform time intervals, no gaps will be inserted.

useAbsoluteTime - if true, print time as YYYY-MM-DD HH:MM:SS. If fals
(default), print time as hour since beginning of experiment (UTH). If
useAbsoluteTime is true, first parameter in isprintText must be UT1, if false,
must be UTH.

startTime - start plot at given time. If useAbsoluteTime == True, then startTi
must be in units of seconds since 1/1/1950. If useAbsoluteTime == False, the
startTime must be in units of UTH (hours since midnight UT of first day of
experiment). Default is None, which means start at lowest time found.

endTime - end plot at given time. If useAbsoluteTime == True, then endTime
be in units of seconds since 1/1/1950. If useAbsoluteTime == False, then end
must be in units of UTH (hours since midnight UT of first day of experiment
Default is None, which means end at largest time found.

sortTimeFlag - if true, check that time is correctly sorted. If false (the default
assume time already sorted

maxNumTimes - if not None, decimate the number of times in the isprint stri
maxNumTimes. If None (the default), plot all times.

maxNumAlt - if not None, reduce the number of altitudes to maxNumAlt. If
(the default), plot all altitudes.

truncateIsprint - if True, and both maxNumTimes and maxNumAlt not = Nor
then truncate the number of isprint lines to be maxNumTimes * maxNumAlt

colorMap - sets colormap. It not given, defaults to matplotlib.cm.jet

yMinimum - minumum y value. If None (default), set by data y minimum.

yMaximum - maximum y value. If None (default), set by data y maximum.

altYTitle - title of right (second) y axis. If None (the default), no Y axis on th
side.

altYLabels - a list of Y labels (strings) for the right axis. If None (the default)
labels on the right axis.

noDate - if True and useAbsoluteTime True, then times without dates printed
axis.

timezone - The offset of the local (non-DST) timezone, in seconds west of U
(negative in most of Western Europe, positive in the US, zero in the UK).

Returns: void

Affects: None

Exceptions

ValueError, 'No valid z data found'
ValueError, 'input text is not parseable'
ValueError, 'size must be "small", "wide", or "large", not %
%(str(size))

__truncateIsprint

```
__truncateIsprint (
        self,
        isprintText,
```

```
        maxLines,
        )
```

__truncateIsprint truncates isprintText to have maxLines at most.

decimateTimes

```
decimateTimes (
        self,
        array_data,
        maxNumTimes,
        insertDataGap,
        )
```

decimateTimes decimates array_data to have at most maxNumTimes times.

Input: array_data - two-dimensional array to be decimated by del times and missing data.

maxNumTimes: int representing the maximum number of times to keep in array_data

insertDataGap - this parameter sets the threshold for inserting a data gap. The intervals being plotted are ordered, and the time gap larger than 90% of the re determined. Note that this parameter is used here to stop the truncation of isp lines that will eventually be considered edge lines.

Returns: new array built from decimated array_data

displayToScreen

```
displayToScreen ( self )
```

to implement this takes a reworking away from pylab to use the underlying matplotlib code

getAverage

```
getAverage ( self,  X )
```

returns the average of items in a float array. Does not including missing data. data missing, returns self.__missing

getFigureHandle

```
getFigureHandle ( self )
```

sortArrayInTime

```
sortArrayInTime ( self,  array_data )
```

sortArrayInTime sorts a two-dimensional array so that first element in each row (time) is in ascending order.

Input: array_data - two-dimensional array to be sorted by rearranging rows so the first element in each row (time) is in ascending order

Returns: new_array

Table of Contents

Table of Contents

Class: madPcolorScan **ui/madrigalPlot.py**

madPcolorScan is the class that produces pcolor scans.

Usage example:

```
obj = madPcolorScan(isprintText,
                    'Nel (log(m^-3)) - 26 June 2006 13:49:43-14:07:36',
                    'Longitude',
                    'Latitude',
                    './isprint.png',
                    size = 'large',
                    minColormap = 9,
                    maxColormap = 12)
```

Non-standard Python modules used: matplotlib

Change history:

Written by Bill Rideout Jul. 20, 2006

Methods

__filter_input

Input: array_data - two-dimensional array to be decimated by deletingtimes and missing data.

```
__filter_input ( self,  x )
```

__filter_input is called in map to convert missing strings to self.__missing, and to round x and y vales to the nearest xGridSize or yGridSize

__init__

```
__init__ (
        self,
        isprintText,
        xGridSize,
        yGridSize,
        titleStr,
        xLabelStr,
        yLabelStr,
        fullFilename,
        size='small',
        xMinimum=None,
        xMaximum=None,
        yMinimum=None,
        yMaximum=None,
        minColormap=None,
        maxColormap=None,
        colorMap=matplotlib.cm.jet,
        maxNumLines=None,
        )
```

__init__ writes a madPcolorScan to a file.

Inputs:

isprintText - a string giving isprint output without headers. First parameter must be the X axis value, and the second must be the Y axis value. The third column is the value (intensity). Any missing data should be written as "missing" or other string that cannot be converted to a float.

xGridSize - grid size for x data (for example 0.1 for 0.1 degree longitude grid)

yGridSize - grid size for x data (for example 0.1 for 0.1 degree latitude grid)

titleStr - plot title (string) - should describe parameter being plotted

xLabelStr - x label string

yLabelStr - ylabel string

fullFilename - full path of file containing pcolor plot to be saved. Extension must be jpeg or png, or exception thrown.

size - size of plot to save. Must be "small", "wide", or "large". Defaults to small.

xMinimum = minumum x value. If None (default), uses lowest x value found.

xMaximum = maximum x value. If None (default), uses highest x value found.

yMinimum = minumum y value. If None (default), uses lowest y value found.

yMaximum = maximum y value. If None (default), uses highest y value found.

minColormap - minimum parameter value (defaults to lowest parameter value)

maxColormap - maximum parameter value (defaults to highest parameter value). However, if both minColormap and maxColormap == None, autoscaling applied.

colorMap - sets colormap. It not given, defaults to matplotlib.cm.jet

maxNumLine - max number of lines in isprintText before truncating. If None, no truncation

Returns: void

Affects: None

Exceptions

ValueError, 'No valid z data found'
ValueError, 'input text is not parseable'
ValueError, 'size must be "small", "wide", or "large", not %s' %(str(size))

__round

```
__round (
        self,
        value,
        increment,
        )
```

__round returns a value to the nearest increment

__truncateIsprint

```
__truncateIsprint (
```

```
        self,
        isprintText,
        maxLines,
        )
```

__truncateIsprint truncates isprintText to have maxLines at most.

displayToScreen

```
displayToScreen ( self )
```

to implement this takes a reworking away from pylab to use the underlying matplotlib code

getAverage

```
getAverage ( self,  X )
```

returns the average of items in a float array. Does not including missing data. If all data missing, returns self.__missing

getFigureHandle

```
getFigureHandle ( self )
```

sortArrayInX

```
sortArrayInX ( self,  array_data )
```

sortArrayInX sorts a two-dimensional array so that the first element in each row (x) is in ascending order.

Input: array_data - two-dimensional array to be sorted by rearranging rows so that the first element in each row (x) is in ascending order

Returns: new_array

Table of Contents

Table of Contents

Class: madScatterPlot **ui/madrigalPlot.py**

sortArrayInX sorts a two-dimensional array so that the first element in each row (x) is in ascending order.

madScatterPlot is the class the produces two dimensional scatter plots of x versus y.

Methods

__filter_missing
__init__
__truncateIsprint

__filter_missing

```
__filter_missing ( self,  x )
```

__init__

```
__init__ (
        self,
        isprintText,
        titleStr,
        xLabelStr,
        yLabelStr,
        fullFilename,
        size='small',
        useAbsoluteTime=False,
        startTime=None,
        endTime=None,
        maxNumPoints=None,
        noTime=False,
        noDate=False,
        yMinimum=None,
        yMaximum=None,
        timezone=0,
        )
```

__init__ writes a madScatter plot to a file.

Inputs:

isprintText - a string giving isprint output without headers. First parameter must be UTH or UT1, depending on whether time scale should be relative to the experiment beginning (UTH) or absolute (UT1). The second must be the parameter to be plotted. Any missing data should be written as "missing" or other string that cannot be converted to a float.

titleStr - plot title (string) - should describe parameter being plotted

xLabelStr - x label string

yLabelStr - ylabel string

fullFilename - full path of file containing pcolor plot to be saved. Extension must be jpeg or png, or exception thrown.

size - size of plot to save. Must be "small", "wide", or "large". Defaults to small.

useAbsoluteTime - if true, print time as YYYY-MM-DD HH:MM:SS. If false (default), print time as hour since beginning of experiment (UTH). If useAbsoluteTime is true, first parameter in isprintText must be UT1, if false, it must be UTH.

startTime - start plot at given time. If useAbsoluteTime == True, then startTime must be in units of seconds since 1/1/1950. If useAbsoluteTime == False, then startTime must be in units of UTH (hours since midnight UT of first day of experiment). Default is None, which means start at lowest time found.

endTime - end plot at given time. If useAbsoluteTime == True, then endTime must be in units of seconds since 1/1/1950. If useAbsoluteTime == False, then endTime must be in units of UTH (hours since midnight UT of first day of experiment). Default is None, which means end at largest time found.

maxNumPoints - maximum number of points to plot. If not None, truncate isprintText if needed to have at most maxNumPoints lines.

noTime - if True and useAbsoluteTime True, then dates without times printed on x axis.

noDate - if True and useAbsoluteTime True, then times without dates printed on x axis.

yMinimum - set y minimum. If default=None, use data minimum

yMaximum - set y maximum. If default=None, use data maximum

timezone - The offset of the local (non-DST) timezone, in seconds west of UTC (negative in most of Western Europe, positive in the US, zero in the UK).

Returns: void

Affects: None

Exceptions

ValueError, 'No valid y data found'
ValueError, 'input text is not parseable'
ValueError, 'size must be "small", "wide", or "large", not %s' %(str(size))

__truncateIsprint

```
__truncateIsprint (
        self,
        isprintText,
        maxLines,
        )
```

__truncateIsprint truncates isprintText to have maxLines at most.

Table of Contents

This document was automatically generated on Thu Oct 20 16:51:50 2011 by HappyDoc version r1_5

Table of Contents

Class: **ui/madrigalPlot.py**

madXYScatterPlot

madXYScatterPlot is the class the produces XY scatter plots.

Change History:

Written by Brandon Scott Fines Aug 2, 2006 Written by Bill Rideout Aug 2, 2006

Methods

__init__

__init__

```
__init__ (
        self,
        inputText,
        titleStr,
        xLabelStr,
        yLabelStr,
        fullFilename,
        size='small',
        lowBound=None,
        highBound=None,
        maxNumPoints=None,
        yLegend=None,
        )
```

Inputs:

inputText - string of values to be plotted. The formatting is as follows:

xval yval xval yval xval yval

titleStr - title of the Plot

xLabelStr - label for the x axis

yLabelStr - label for the y axis

fullFilename - full file path to save the resulting picture to

size - size of plot to be saved. Must be `small`,'wide', or `large`. defaults to `small`.

lowBound - lower bound on the x-axis value. If no bound is specified, the lowest value found in inputText will be used.

highBound - upper bound on the x-axis value. If no bound is specified, the highest value found in inputText will be used.

maxNumPoints - maximum number of points to be plotted.

Outputs: None

Affects: Creates a scatter plot using matplotlib and writes that to the file designated by the variable `fullFilename`

Exceptions: ValueError if lowBound or highBound cannot be converted to a float value

Non-standard python modules used:

matplotlib

Exceptions	
	ValueError, 'lowBound not a number' ValueError, 'no valid y data found' ValueError, 'size must be "small","wide", or "large", not %s' %(str(size))

Table of Contents

This document was automatically generated on Thu Oct 20 16:51:50 2011 by HappyDoc version r1_5

Table of Contents

Class:	**ui/madrigalPlot.py**

scanPlotter

scanPlotter is the class that produces a series of scan plots for a single Madrigal file

Non-standard Python modules used: matplotlib

Change history:

Written by Bill Rideout Jul. 26, 2006

Methods

__init__

```
__init__ (
        self,
        madFile,
        madDBObj=None,
        )
```

__init__ initializes a scanPlotter object.

Inputs:

madFile - the Madrigal file to be analyzed. Must contain SCNTYP and CYCN parameters.

madDBObj - a madrigal.metadata.MadrigalDB object. If None (default), one created

Returns: void

Affects: sets self.madFile, self.madDBObj

Exceptions

IOError, 'unable to read %s' %(str(madFile))

_getInstLocation

```
_getInstLocation ( self,  isprintStr )
```

_getInstLocation returns a tuple of instrument location (gdlatr, gdlonr, galtr)

Input isprint string has parameters ut1,scntyp,cycn,<parm>,gdalt,gdlat,glon,azm,elm,gcdist, gdlatr,gdlonr,galtr,range,az1,az2,el1,el2

_getLimits

```
_getLimits (
        self,
        scanList,
        xMinimum,
        xMaximum,
        yMinimum,
        yMaximum,
        )
```

_getLimits returns a dictionary with keys that may include (`az`, `el_lat`, `el_lon`, `el_gcdist`), though not all will neccessarily be there. Values are overall tuple of (xMinimum, xMaximum, yMinimum, yMaximum). These values are set by the input arguments unless it is None, in which case the scanList limits set the values.

Inputs:

scanList: a list of tuples, which each tuple representing a single scan, and has values of: (isprintString, startUT, startAz, startEl, endUT, endAz, endEl, type, minGdlat, maxGdlat,minGlon, maxGdlon,minGdalt, maxGdalt, minGcdist, maxGcdist). Isprint string has values (gdlatr, gdlonr, galtr, range, az1, az2, el1, el2,plotParm),. Type is Gdlat or Glon or Gcdist for el scans, None for az scans.

xMinimum, xMaximum, yMinimum, yMaximum - as passed into scanPlotter

createScanInfoList

```
createScanInfoList ( self,  isprintStr )
```

createScanInfoList creates a list of tuples, which each tuple representing a single scan, and values of:

(isprintString, startUT, startAz, startEl, endUT, endAz, endEl, type, minGdlat, maxGdlat,minGlon, maxGdlon,minGdalt, maxGdalt, minGcdist, maxGcdist). Isprint string has values (x,y,plotParm), where for az scan x=lon, y=lat, and for el scan y=alt, x=lat if starting az within 15 degrees of north or south, x=lon if starting az within 15 degrees of east or west, of x=gcdist if other az. Type is Gdlat or Glon or Gcdist for el scans, None for az scans.

Input isprint string has parameters ut1,scntyp,cycn,<parm>,gdalt,gdlat,glon,azm,elm,gcdist

Azimuth scans are split whenever direction or elevation changes. For elevation scans, there will be zero or one north-south scans, zero or one east-west scans, and zero or more off azimuth scans. An off azimuth scan is not within 15 degrees of north, south, east or west. Off azimuth scans are not combined with scans 180 degrees in the other direction, because the x axis is ground distance, which does not reverse sign.

createWedgeScanInfoList

```
createWedgeScanInfoList ( self,  isprintStr )
```

createWedgeScanInfoList creates a list of tuples, which each tuple representing a single scan, and values of:

(isprintString, startUT, startAz, startEl, endUT, endAz, endEl, type, minGdlat, maxGdlat,minGlon, maxGdlon,minGdalt, maxGdalt, minGcdist, maxGcdist).

Input isprint string has parameters ut1,scntyp,cycn,<parm>,gdalt,gdlat,glon,azm,elm,gcdist, gdlatr,gdlonr,galtr,range,az1,az2,el1,el2

Azimuth scans are split whenever direction or elevation changes. For elevation scans, there will be zero or more north-south scans, zero or more east-west scans, and zero or more off azimuth scans. A south to north elevation scan or a west to east elevation scan is called clockwise, and a switch from clockwise to counterclockwise will create a new elevation scan. An off azimuth scan is not within 15 degrees of north, south, east or west. Off azimuth scans are not combined with scans 180 degrees in the other direction, because the x axis is ground distance, which does not reverse sign.

getDateStrFromUT

```
getDateStrFromUT ( self,  ut )
```

getDateStrFromUT returns a date string formated as YYYY-MM-DD HH:MM:SS from a ut time (seconds since 1/1/1950)

getTimeStrFromUT

```
getTimeStrFromUT ( self,  ut )
```

getTimeStrFromUT returns a time string formated as HH:MM:SS from a ut time (seconds since 1/1/1950)

plotAllScans

Inputs:

```
plotAllScans (
        self,
        scanType,
        fullFilenameTemplate,
        plotParm,
        size='small',
        xMinimum=None,
        xMaximum=None,
        yMinimum=None,
        yMaximum=None,
        xGridSize=None,
        yGridSize=None,
        minColormap=None,
        maxColormap=None,
        colorMap=matplotlib.cm.jet,
        maxNumLines=None,
        filterStr='',
        )
```

plotAllScans creates a series of az or el scans.

Inputs:

scanType - must be `az` or `el`

fullFilename - full path of file containing pcolor plot to be saved. Each image created will have _#.png appended, with # starting at 0

plotParm - mnemonic of parameter to be plotted.

size - size of plot to save. Must be "small", "wide", or "large". Defaults to small.

xMinimum = minumum x value. If None (default), uses lowest x value found for each scan.

xMaximum = maximum x value. If None (default), uses highest x value found for each scan.

yMinimum = minumum y value. If None (default), uses lowest y value found for each scan.

yMaximum = maximum y value. If None (default), uses highest y value found for each scan.

minColormap - minimum parameter value (defaults to lowest parameter value)

maxColormap - maximum parameter value (defaults to highest parameter value). However, if both minColormap and maxColormap == None, autoscaling applied.

colorMap - sets colormap. It not given, defaults to matplotlib.cm.jet

maxNumLine - max number of lines in isprintText before truncating. If None, no truncation

filterStr - a filter string to pass to isprint. Defaults to no filtering

Returns - a list of tuples, where each tuple has two items: 1. time string, 2. metadata string in form for scanTab.txt. List length = number of plots created

Exceptions

ValueError, 'scantype must be az or el, not %s' %(str(scanType))

plotAllWedgeScans

```
plotAllWedgeScans (
        self,
        scanType,
        fullFilenameTemplate,
        plotParm,
        size='small',
        xMinimum=None,
        xMaximum=None,
        yMinimum=None,
        yMaximum=None,
        minColormap=None,
        maxColormap=None,
        colorMap=matplotlib.cm.jet,
        maxNumLines=None,
        filterStr='',
        addTitle='',
        includeKml=False,
        radarName=None,
        radarDesc=None,
        )
```

plotAllWedgeScans creates a series of az or el scans. Similar to plotAllScans, except produces fancier wedge plots with uniform scalling.

Inputs:

scanType - must be `az` or `el`

fullFilename - full path of file containing pcolor plot to be saved. Each image created will have _#.png appended, with # starting at 0

plotParm - mnemonic of parameter to be plotted.

size - size of plot to save. Must be "small", "wide", or "large". Defaults to small.

xMinimum = minumum x value. If None (default), uses lowest x value found for each scan.

xMaximum = maximum x value. If None (default), uses highest x value found for each scan.

yMinimum = minumum y value. If None (default), uses lowest y value found for each scan.

yMaximum = maximum y value. If None (default), uses highest y value found for each scan.

minColormap - minimum parameter value (defaults to lowest parameter value)

maxColormap - maximum parameter value (defaults to highest parameter value). However, if both minColormap and maxColormap == None, autoscaling applied.

colorMap - sets colormap. It not given, defaults to matplotlib.cm.jet

maxNumLine - max number of lines in isprintText before truncating. If None, no truncation

filterStr - a filter string to pass to isprint. Defaults to no filtering

addTitle - a string with additional title. If empty string (the default), no additional title string.

includeKml - if True, create a corresponding kml file for every png file. Same names as png files, except extension = .kml. If False, do not.

radarName - name to label radar placemark if kml generated.

radarDesc - description text for radar placemark if kml generated

Returns - a list of tuples, where each tuple has two items: 1. time string, 2. metadata string in form for scanTab.txt. List length = number of plots created

Exceptions	
	ValueError, 'scantype must be az or el, not %s' %(str(scanType))

Table of Contents

Table of Contents

Module: report — **ui/report.py**

report is the module that creates reports on Madrigal data.

$Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $

Classes

MadrigalReport	MadrigalReport is the class that produces reports on Madrigal data.

Table of Contents

This document was automatically generated on Fri Jul 13 09:36:49 2007 by HappyDoc version r1_5

Table of Contents

Class: MadrigalReport — **ui/report.py**

MadrigalReport is the class that produces reports on Madrigal data.

Non-standard Python modules used: None

Change history:

Written by Bill Rideout May. 23, 2002

Modified by Bill Rideout Feb. 6, 2003 to use the C Maddata module

Modified by Bill Rideout Apr. 9, 2003 to add looker reports

Methods

__addPid
__dropLock
__getLock
__getNumQueriesRunning
__init__
genBackgroundReport
looker
looker2

__addPid

```
__addPid ( self,  pid )
```

__addPid is a private helper function that adds a new pid to queryPid.txt.

Inputs: the pid to add.

Returns: None

Affects: Locks the global file metadata/userdata/queryPid.txt while being used; adds new pid to that file.

Exceptions: None

__dropLock

```
__dropLock ( self,  filename )
```

__dropLock is a private helper function that drops exclusive access to filename via a locking file.

Inputs: filename = the file that exclusive access is required to.

Returns: None

Affects: Removes file filename + .LCK as a lock mechanism

Exceptions: None.

__getLock

```
__getLock ( self,  filename )
```

__getLock is a private helper function that provides exclusive access to filename via a locking file.

Inputs: filename = the file that exclusive access is required to.

Returns: None

Affects: Writes file filename + .LCK as a lock mechanism

Exceptions: MadrigalError thrown if unable to write lock file

Notes: Will sleep for 1 second at a time, for a maximum of _MaxSleep seconds (presently 10) if the file is not modified. After each second, it will check for the lock file to be removed or modified. If it was modified, it resets the count to 0 sec and starts

counting again. After _MaxSleep counts it then assumes lock file is orphaned and returns. Orphaned file will be removed when dropLock is called.

Exceptions

madrigal.admin.MadrigalError("Unable to open " + filename + ".LCK as locking file ", None)

__getNumQueriesRunning

```
__getNumQueriesRunning ( self )
```

__getNumQueriesRunning is a private helper function that returns the number of global queries running.

Inputs: None.

Returns: the number of global queries running

Affects: Locks the global file metadata/userdata/queryPid.txt while being used; removes any dead process ids from that file.

Exceptions: None

Exceptions

madrigal.admin.MadrigalError('Unable to read from file ' + str(queryFileName), None)
madrigal.admin.MadrigalError('Unable to write to file ' + str(queryFileName), None)

__init__

```
__init__ ( self,  madDB=None )
```

__init__ initializes MadrigalWeb by reading from MadridalDB..

Inputs: Existing MadrigalDB object, by default = None.

Returns: void

Affects: Initializes self.__metaDir.

Exceptions: None.

genBackgroundReport

```
genBackgroundReport (
        self,
        email,
        filenameList,
        parmList,
        filterList,
        headerStr='',
        summaryLevelIndicator=0,
        )
```

genBackgroundReport starts a separate process that emails a report.

Inputs:

email - the email address to send the complete report to

filenameList - a list of full file paths to collect data from. (May be generated by MadrigalDB.getFileList). The order of the filenames will be the order they are listed in the report.

parmList - list of Mnemonics (may be in the form of integer strings) to be displayed.

filterList - list of filters to select data. Each filter is a string of the form "mnem1 [,[+, -, /, *],mnem2],[lower limit 1],[upper limit 1][other limits]

Examples:

```
&quot;gdalt,-,sdwht,0,10000&quot;

&quot;azm,170,180,-180,-170&quot;
```

headerStr - a String to insert at the beginning of the report. (For example, can include iformation on the parameters used to select the data.) Defaults to ''

summaryLevelIndicator: 0 - data level (the default), include data, and file info. 1 - same as 0 (data level), but with no file names and parameter labels between data. 2 - record level, include only record headers and file info. 3 - cycle level (not yet implemented, for now acts just like experiment level) 4 - experiment level, include only file info.

Returns: Process id (integer) of background process created. If too many processes are already running, returns -1 and does not generate a report.

Affects: Generates a separate process to email a report.

looker

```
looker (
        self,
        parmList,
        start_lat,
        stop_lat,
        step_lat,
        start_lon,
        stop_lon,
        step_lon,
        start_alt,
        stop_alt,
        step_alt,
        year,
        month,
        day,
        hour,
        min,
        sec,
        printHeaderFlag=1,
        oneDParmList=[],
        oneDParmValues=[],
        )
```

looker prints a Madrecord for a range of lat, lon, and alt.

Inputs:

parmList - a list of requested parameter mnemonics. Do not include gdlat, glon, or gdalt - included by default

start_lat - start latitude in degrees

stop_lat - end latitude in degrees

step_lat - number of degrees in lat to step

start_lon - start longitude in degrees

stop_lon - end longitude in degrees

step_lon - number of longitude in lat to step

start_alt - start altitude in km

stop_alt - stop altitude in km

step_alt - number of km in alt to step

year

month

day

hour

min

sec

printHeaderFlag - if 0, don't print header. Default will print header.

oneDParmList - a python list of one-D parameters as mnemonics. Defaults to [].

oneDParmValues - a python list of doubles representing values of the one-D parameters set in oneDParmList. Length must = len(oneDParmList). Defaults to [].

Returns: None.

Affects: Prints a single Madrecord with lat, lon and alt as 2D parameters, along with all requested parameters. If stop_lon < start_lon, goes through 0 long.

Exceptions: If problems with arguments

Exceptions

ValueError, 'Illegal float %s in oneDParmValues' %(str(oneDParmValues [i]))
ValueError, 'Illegal parm %s in oneDParmList' %(str(oneDParmList [i]))
ValueError, 'Len of oneDParmList=%i, must equal len of oneDParmValues=%i' %(len(oneDParmList), (oneDParmValues))

looker2

```
looker2 (
        self,
        parmList,
        lats,
        longs,
        alts,
        year,
        month,
        day,
        hour,
        min,
```

```
        sec,
        printHeaderFlag=1,
        oneDParmList=[],
        oneDParmValues=[],
        twoDParmList=[],
        twoDParmValues=[],
        )
```

looker2 prints a Madrecord for a collection of lat, lon, and alt values.

It is similar to looker, but it does not use a grid of lat, lon, and alts. Instead, each input point is independent. Also, it allows users to set both 1D and 2D parameters.

Inputs:

parmList - a list of requested parameter mnemonics. Do not include gdlat, glon, or gdalt - included by default

lats - a list of latitudes to use

longs - a list of longitudes to use

alts - a list of altitudes (km) to use

year

month

day

hour

min

sec

printHeaderFlag - if 0, don't print header. Default will print header.

oneDParmList - a python list of one-D parameters as mnemonics. Defaults to [].

oneDParmValues - a python list of doubles representing values of the one-D parameters set in oneDParmList. Length must = len(oneDParmList). Defaults to [].

twoDParmList - a python list of two-D parameters as mnemonics. Defaults to [].

twoDParmValues - a python list of lists of len = len(twoDParmList). Each individual list is a list of doubles representing values of the two-D parameter set in twoDParmList, with a length = number of points (or equal to len(lats)). Defaults to [].

Returns: None.

Affects: Prints a single Madrecord with lat, lon and alt as 2D parameters, along with all requested parameters.

Exceptions: If problems with arguments

Exceptions	
	ValueError, 'Illegal float %s in oneDParmValues' %(str(oneDParmValues [i])) ValueError, 'Illegal parm %s in oneDParmList' %(str(oneDParmList [i])) ValueError, 'Len of oneDParmList=%i, must equal len of oneDParmValues=%i' %(len(oneDParmList), (oneDParmValues))

Table of Contents

This document was automatically generated on Thu Oct 20 16:51:50 2011 by HappyDoc version r1_5

Table of Contents

Module:	ui/userData.py

userData

userData is responsible for interfacing to all persisted user data on the madrigal web site.

This module is meant to hide the storage mechnaism of user data on the madrigal web site, so that the present format (xml files) can be changed by only changing this module. The data stored at the moment consists of user login information, and directories (private and public) of stored filters, where a filter is all information that determines the output of an isprint display.

This modules requires that the non-standard python module PyXML be installed (this module may become standard in a future release of python). See the python XML SIG for this module.

$Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $

Classes	

MadrigalDirectory	MadrigalDirectory is a public object that provides access to information in a single user directory.
MadrigalFilter	MadrigalFilter is an object used to filter Madrigal data.
MadrigalUserData	MadrigalUserData is an object that provides access to all user data.
getDirInfoParser	getDirInfoParser is a private Sax Parser designed to rapidly parse a <username>.xml file for directory/filter names.
getFilterParser	getFilterParser is a private Sax Parser designed to rapidly parse a <username>.xml file for one filter.

Table of Contents

This document was automatically generated on Fri Jul 13 09:36:49 2007 by HappyDoc version r1_5

Table of Contents

Class: MadrigalDirectory **ui/userDa**

MadrigalDirectory is a public object that provides access to information in a single user directory.

The MadrigalDirectory object is designed to allow easy access to the information in one directo of user data. At the moment this data is the directory name, the directory type, and a list of filternames in that directory. Created by MadrigalUserData.getAllDirInfo()

Usage example:

```
import madrigal.ui.userData

test = madrigal.ui.userData.MadrigalUserData()

userDirInfo = test.getAllDirInfo()

# print all the directory information for user brideout

for madDir in userDirInfo['brideout']:

    print 'Directory name is ' + madDir.dirName

    print 'This directory type (public or private) is ' + madDir.dir

    print 'The filter names in this directory are:'

    for filtername in madDir.filterList:

        print '    ' + filtername
```

Non-standard Python modules used: None

Exceptions thrown: None

Change history:

Written by Bill Rideout Dec. 11, 2001

Methods

__init__
toString

__init__

```
__init__ (
        self,
        dirName,
        dirType,
        )
```

__init__ initializes MadrigalDirectory by initializing th class members from the inputs.

Inputs: dirName - case sensitive name of directory (string), dirType = pub or private (string).

Returns: void

Affects: Initializes all the class member variables. All member variables a public.

Exceptions: None.

toString

```
toString ( self )
```

Table of Contents

This document was automatically generated on Fri Dec 30 09:02:26 2005 by HappyDoc version r1_5

Table of Contents

Class: MadrigalFilter **ui/userData.py**

MadrigalFilter is an object used to filter Madrigal data.

This object provides access to all parameters used to filter Madrigal data for display. All its data members are public.

Non-standard Python modules used: None

Exceptions thrown: None

Change history:

Written by Bill Rideout Dec. 10, 2001

Methods

__init__
toString

__init__

```
__init__ (
        self,
        name=None,
        startyear=None,
        startmonth=None,
        startday=None,
        endyear=None,
        endmonth=None,
        endday=None,
        starthour=None,
        startmin=None,
        startsec=None,
        endhour=None,
        endmin=None,
        endsec=None,
        minalt=None,
        maxalt=None,
        minaz=None,
        maxaz=None,
        minaz2=None,
        maxaz2=None,
        minel=None,
        maxel=None,
        minel2=None,
        maxel2=None,
        minpl=None,
        maxpl=None,
        mnemStr1=None,
        lower1=None,
        upper1=None,
        mnemStr2=None,
        lower2=None,
        upper2=None,
        flkinst=None,
        flkdat=None,
        parmlist=None,
        description=None,
```

```
        header=None,
        badval=None,
        mxchar=None,
        assumed=None,
        knownBad=None,
        )
```

__init__ initializes MadrigalFilter by reading in arguments if any. All default to None.

Inputs: name filter name - string

startyear starting year - integer

startmonth starting month - integer

startday starting day - integer

endyear ending year - integer

endmonth ending month - integer

endday ending day - integer

starthour starting hour - integer

startmin starting minute - integer

startsec starting second - integer

endhour ending hour - integer

endmin ending minute - integer

endsec ending second - integer

minalt minimum altitude in km - float

maxalt maximum altitude in km - float

minaz minimum azimuth in degrees - float

maxaz maximum azimuth in degrees - float

minaz2 minimum azimuth for second range in degrees - float

maxaz2 maximum azimuth for second range in degrees - float

minel minimum elevation in degrees - float

maxel maximum elevation in degrees - float

minel2 minimum elevation for second range in degrees - float

maxel2 maximum elevation for second range in degrees - float

minpl minimum pulse length in microseconds - float

maxpl maximum pulse length in microseconds - float

mnemStr1 a general mnemonic or two mnemonics separated by +-*/ - str

lower1 lower limit (if any) for mnemStr1 - float

upper1 upper limit (if any) for mnemStr1 - float

mnemStr2 a general mnemonic or two mnemonics separated by +-*/ - str

lower2 lower limit (if any) for mnemStr2 - float

upper2 upper limit (if any) for mnemStr2 - float

flkinst the kind of instrument filter (used only if more than one in file) - integer

flkdat the kind of data filter (used only if more than one in file) - integer

parmlist a python string holding the mnemonics representing the parameters to display (space delimited)

description a python string holders the users description

header a python string = `checked`

badval a python string representing a missing value in isprint output

mxchar represents the maximum characters per column to print - integer

assumed a python string representing an assumed value in isprint output

knownBad a python string representing a known bad value in isprint output

Returns: void

Affects: Initializes any class member variables with data passed in.

Exceptions: none

toString

```
toString ( self )
```

Table of Contents

This document was automatically generated on Tue Mar 17 10:13:25 2009 by HappyDoc version r1_5

Table of Contents

Class:
MadrigalUserData

MadrigalUserData is an object that provides access to all user data

The object MadrigalUserData is an object that provides read and write access to all persisted us
Madrigal web site. For the moment this data consists of directories of filters used in isprint, but
may be added later. At the moment this data is stored in xml files, but the public part of this cla
if another storage implementation (e.g., a database) is used

Usage example:

```
import madrigal.ui.userData

test = madrigal.ui.userData.MadrigalUserData()

print test.getUsersList()
```

Non-standard Python modules used:

xml from PyXML SIG

MadrigalError exception thrown if:

1. Problem reading/writing to xml files containing user data

Change history:

Written by Bill Rideout Dec. 11, 2001

Methods

__createFilterElement	addUser	getReg
__dropLock	changePassword	getReg
__elementExists	getAllDirInfo	getUse
__getLock	getFilter	register

__createFilterElement

```
__createFilterElement (
        self,
        userDoc,
        filter,
        )
```

__createFilterElement is a private helper function that tak python object and converts to an xml node.

Inputs: userDoc - the xml document to be added to, filter - a MadrigalFilter objec node in an xml file.

Returns: the xml node created

Affects: None

Exceptions: None

Notes: Each attribute added to the node takes four lines of code, basically 1) creat element, 3 append text element to new element, and 4) append new element to its MadrigalFilter has at the moment 23 attributes, this function is long but simply re One dom implementaion hint: always finish creating a node first before appendin appended to a parent, it cannot have other nodes appended to it. So always appen

__dropLock

```
__dropLock ( self,  filename )
```

__dropLock is a private helper function that drops exclus filename via a locking file.

Inputs: filename = the file that exclusive access is required to.

Returns: None

Affects: Removes file filename + .LCK as a lock mechanism

Exceptions: None.

__elementExists

```
__elementExists (
        self,
        elem,
        elemTag,
        doc,
        )
```

__elementExists is a private helper function that returns element elemTag with text elem, 0 otherwise.

Inputs: elem - a string to match text nodes against, elemTag - a string giving the e and doc is the xml document.

Returns: returns 1 if the xml doc has a element elemTag with text elem, 0 otherwi

Affects: None

Exceptions: None

Usage: If an xml file contains <person age=43> John Doe </person>, and you cal `Doe`, `person`, doc), 1 will be returned

__getLock

```
__getLock ( self,  filename )
```

__getLock is a private helper function that provides exclu filename via a locking file.

Inputs: filename = the file that exclusive access is required to.

Returns: None

Affects: Writes file filename + .LCK as a lock mechanism

Exceptions: MadrigalError thrown if unable to write lock file

Notes: Will sleep for 1 second at a time, for a maximum of _MaxSleep seconds (modified. After each second, it will check for the lock file to be removed or modi resets the count to 0 sec and starts counting again. After _MaxSleep counts it ther orphaned and returns. Orphaned file will be removed when dropLock is called.

Exceptions	
	madrigal.admin.MadrigalError("Unable to open " + filena ", None)

__getText

```
__getText ( self,  nodelist )
```

__dropLock is a private helper function that drops exclusive access tofilename via a locking file.

__getText is a private helper function that returns a string text nodes of a node list.

Inputs: A list of xml nodes.

Returns: a string made up of all the text nodes of a node list.

Affects: None

Exceptions: None

Usage: If an xml file contains <person age=43> John Doe </person>, and the elem `person` element, then self.__getText(elem.childNodes) will return `John Doe`, text node of person, and __getText strips leading and trailing whitespace.

__getXmlString

```
__getXmlString ( self,  obj )
```

__getXmlString is a private helper function that extends s return an empty string on a null object.

Inputs: obj - the object to be converted to a string.

Returns: if the object == None, returns '', otherwise returns str(obj). Normally str(

Affects: None

Exceptions: None.

__init__

```
__init__ ( self,  madDB=None )
```

__init__ initializes MadrigalUserData by reading from Mad

Inputs: Existing MadrigalDB object, by default = None.

Returns: void

Affects: Initializes self.__metaDir.

Exceptions: None.

__isValidFileName

```
__isValidFileName ( self,  name )
```

__isValidFileName is a private helper function that validat name does not contain excluded characters.

Inputs: name - the string to be validated.

Returns: 1 if all characters are allowed, 0 otherwise. Valid if the following charac string after leading and trailing whitespace is removed: [' ', /, <, >, "']

Affects: None

Exceptions: None.

__loadUserData

```
__loadUserData ( self )
```

__loadUserData is a private helper function that reads in from an xml file.

Inputs: None.

Returns: void

Affects: Populates self.__userList.

Exceptions: MadrigalError thrown if problem parsing users.xml.

Depends on: Existance of file in metadata dir self.__userXMLDir + / + self.__us userdata/user.xml)

This file must be of the form of the following format:

<?xml version='1.0'?>

<users>

<user>

<name>brideout</name>

<password>briWy6v1L.z1E</password>

</user>

possibly more users...

</users>

The password is stored encrypted by crypt.crypt(password, self.__cryptStr). Impl
is only a short file, but for higher speed (and more complex code) could be imple
only.

Exceptions

madrigal.admin.MadrigalError("Unable to open " + filena

addDirectory

```
addDirectory (
        self,
        username,
        dirname,
        dirtype,
        )
```

addDirectory returns 1 if directory added successfully to error string otherwise.

Inputs: username string, directory name string, dirtype string (public or private).

Returns: 1 if directory added successfully, error string otherwise. dirtype must be
Directory names are case sensitive, so Bill and bill can both be directories

Affects: Adds new directory to <username>.xml file

Exceptions: MadrigalError thrown if unable to write to <username>.xml file

Notes: uses getLock and dropLock to insure exclusive access to <username>.xml

Exceptions

madrigal.admin.MadrigalError, "Unable to open " + filena

addFilter

```
addFilter (
        self,
        username,
        dirname,
        filter,
        )
```

addFilter returns 1 if a new filter is added successfully to <username>.xml, 0 otherwise.

Inputs: username string, directory name string, MadrigalFilter object to be added.

Returns: 1 if filter added successfully, error string otherwise.

Affects: Adds new filter to <username>.xml file

Exceptions: MadrigalError thrown if unable to write to <username).xml file

Notes: uses getLock and dropLock to insure exclusive access to <username>.xml

Exceptions

madrigal.admin.MadrigalError, "Unable to open " + filena

addUser

```
addUser (
        self,
        username,
        password,
        )
```

addUser returns 1 if user added successfully, error string

Inputs: username string, password string (password is not yet encrypted).

Returns: 1 if user added successfully, error string otherwise.

Affects: Adds new user to self.__userList, writes new empty <username>.xml file converted and stored as lower case, so its case insensitive.

Exceptions: MadrigalError thrown if unable to write to user xml file

Notes: uses getLock and dropLock to insure exclusive access to user file

changePassword

```
changePassword (
        self,
        username,
        password,
        )
```

changePassword returns 1 if user password changed suc string otherwise.

Inputs: username string, password string (password is not yet encrypted).

Returns: 1 if password changed successfully, error string otherwise.

Affects: Modifies password in self.__userList, writes new user.xml file

Exceptions: MadrigalError thrown if unable to write user xml file

getAllDirInfo

```
getAllDirInfo ( self )
```

getAllDirInfo returns a dictionary with key=username, val MadrigalDirectory objects owned by user.

Inputs: none.

Returns: a dictionary with key=username, value=list of MadrigalDirectory object for the description of the information in a MadrigalDirectory class.

Affects: builds __dirsDict. Other public functions that use __dirsDict will call get __dirsDict if its = None

Exceptions: MadrigalError thrown if unable to read a user file

Usage example:

```
import madrigal.ui.userData

test = madrigal.ui.userData.MadrigalUserData()

userDirInfo = test.getAllDirInfo()

# print all the directory information for user brideo

for madDir in userDirInfo['brideout']:

    print 'Directory name is ' + madDir.dirName

    print 'This directory type (public or private) is

    print 'The filter names in this directory are:'

    for filtername in madDir.filterList:

        print '    ' + filtername
```

Notes: Presently implemented by the sax handler class getDirInfoParser (included

Exceptions

madrigal.admin.MadrigalError, "Unable to open " + userF

getFilter

```
getFilter (
        self,
        username,
        dirname,
        filtername,
        )
```

getFilter returns a MadrigalFilter object specified by the u and filtername.

Inputs: username string (case insensitive), dirname string (case sensitive), filterna

Returns: a MadrigalFilter object specified by the username, dirname, and filternar None

Affects: None

Exceptions: MadrigalError thrown if unable to read a user file

Notes: Presently implemented by the sax handler class getFilterParser (included i than the use of dom.

Exceptions

	madrigal.admin.MadrigalError, "Unable to open " + userF

getFilterNameList

```
getFilterNameList (
        self,
        username,
        dirName,
        )
```

getFilterNameList returns a list of strings of the names of directory of a given user.

Inputs: username (string), directory name (string).

Returns: a list of strings of the names of all filters in a given directory of a given u

Affects: Builds __dirsDict by calling getAllDirInfo if not yet populated

Exceptions: None

getPrivateDirNameList

```
getPrivateDirNameList ( self,  username )
```

getPrivateDirNameList returns a list of strings of the nam directories for a given user.

Inputs: username (string).

Returns: a list of strings of the names of all private directories owned by that user

Affects: Builds __dirsDict by calling getAllDirInfo if not yet populated

Exceptions: None

getPublicDirNameList

```
getPublicDirNameList ( self )
```

getPublicDirNameList returns a list of strings of the name directories in form `user:dirName`.

Inputs: none.

Returns: a list of strings of the names of all public directories

Affects: Builds __dirsDict by calling getAllDirInfo if not yet populated

Exceptions: None

getRegisteredDict

```
getRegisteredDict ( self )
```

will return a dictionary with key + email, and value = tuple of all registered exper

This data will be stored in a text file: MADROOT/metadata/userdata/ regExp.txt. delimited columns: email experimentString.

getRegisteredExperiments

```
getRegisteredExperiments ( self,  email )
```

getRegisteredExperiments will return a list of experiments as strings. For exampl getRegisteredExperiments(`brideout at haystack.mit.edu`) might retur ['experiments/2010/mlh/18jan10', `experiments/2010/mlh/ 19jan10`, 'experiments/2010/mlh/20jan10']. If the user has no registered experiments, it wil

Inputs: user email

getRegisteredInstDict

```
getRegisteredInstDict ( self )
```

will return a dictionary with key = email, and value = tuple of all registered instru email.

This data will be stored in a text file: MADROOT/metadata/userdata/ regInst.txt. delimited columns: email instrument code (int).

getRegisteredInstUsers

`getRegisteredInstUsers ( self,  kinst )`

getRegisteredInstUsers will return a list of users registered for a given instrument
example, getRegisteredInstUsers(30) might return ['brideout at haystack.mit.edu '
jro.igp.gob.pe']. If the instrument has no registered users, it will return an empty l

getRegisteredInstruments

`getRegisteredInstruments ( self,  email )`

getRegisteredInstruments will return a list of instruments as ints. For example,
getRegisteredInstruments(`brideout at haystack.mit.edu`) might return
no registered instruments, it will return an empty list.

Inputs: user email

getRegisteredUsers

`getRegisteredUsers ( self,  experimentString )`

getRegisteredUsers will return a list of users register for a given experiment as str
getRegisteredUsers(`experiments/2010/mlh/18jan10`) might return ['bric
'miguel.urco at jro.igp.gob.pe']. If the experiment has no registered users, it will r

getUsersList

`getUsersList ( self )`

getUsersList returns a list of user names/encrypted passw exist.

Inputs: none.

Returns: a list of user names/passwords. Each item in the returned list is itself a li
username, and 2) encrypted password.

Usage example:

```
import madrigal.ui.userData

test = madrigal.ui.userData.MadrigalUserData()

userlist = test.getUsersList()

for user in userlist:

    print 'User name is ' + user[0] + ' and encrypted
```

Affects: None

getPublicDirNameList returns a list of strings of the names of all publicdirectories in form user:dirname.

Exceptions: none

registerExperiment

```
registerExperiment (
        self,
        email,
        experimentString,
        )
```

registerExperiment method will register an experiment for a given email. For exa
registerExperiment(`brideout at haystack.mit.edu`, `experiments/`
would register an experiment. If the experiment is already registed, it will not rais
nothing.

Affects: This new data will be stored in a text file: MADROOT/metadata/userdat
be two space delimited columns: email experimentString.

registerInstrument

```
registerInstrument (
        self,
        email,
        kinst,
        )
```

registerInstrument method will register an instrument for a given email. For exam
registerInstrument(`brideout at haystack.mit.edu`, 30) would register
instrument is already registed, it will not raise an error; it will simply do nothing.

Affects: This new data will be stored in a text file: MADROOT/metadata/userdat
be two space delimited columns: email kinst.

removeDirectory

```
removeDirectory (
        self,
        username,
        dirname,
        )
```

removeDirectory returns 1 if a directory is removed succe directory in <username>.xml, error string otherwise.

Inputs: username string, directory name to be removed.

Returns: 1 if filter removed successfully, error string otherwise. Will not remove
filters.

Affects: Removes existing dirctory from <username>.xml file

Exceptions: MadrigalError thrown if unable to write to <username).xml file

Notes: uses getLock and dropLock to insure exclusive access to <username>.xml

Exceptions

madrigal.admin.MadrigalError, "Unable to open " + filena

removeFilter

```
removeFilter (
        self,
        username,
        dirname,
        filtername,
        )
```

removeFilter returns 1 if a filter is removed successfully f <username>.xml, error string otherwise.

Inputs: username string, directory name string, filtername of filter to be removed.

Returns: 1 if filter removed successfully, error string otherwise.

Affects: Removes existing filter from <username>.xml file

Exceptions: MadrigalError thrown if unable to write to <username).xml file

Notes: uses getLock and dropLock to insure exclusive access to <username>.xml

Exceptions

madrigal.admin.MadrigalError, "Unable to open " + filena

unregisterExperiment

```
unregisterExperiment (
        self,
        email,
        experimentString,
        )
```

unregisterExperiment method will unregister an experiment for a given email. Fo unregisterExperiment(`brideout at haystack.mit.edu`, `experiments` would unregister an experiment. If the experiment is not registed, it will not raise nothing.

Affects: This data will be removed from a text file: MADROOT/metadata/userda will be two space delimited columns: email experimentString.

unregisterInstrument

```
unregisterInstrument (
        self,
```

```
        email,
        kinst,
        )
```

unregisterInstrument method will unregister an instrument for a given email. For
unregisterInstrument(`brideout at haystack.mit.edu`, 30) would unreg
instrument is not registed, it will not raise an error; it will simply do nothing.

Affects: This data will be removed from a text file: MADROOT/metadata/userda
will be two space delimited columns: email kinst.

userExists

```
userExists ( self,  username )
```

userExists returns 1 if username (case insensitive) exists

Inputs: username string.

Returns: 1 if username (case insensitive) exists, 0 otherwise.

Affects: None

Exceptions: none

verifyUser

```
verifyUser (
        self,
        username,
        password,
        )
```

verifyUser returns 1 if username, password okay, 0 other

Inputs: username string, password string.

Returns: 1 if username, password okay, 0 otherwise.

Affects: None

Exceptions: none

Table of Contents

This document was automatically generated on Thu Oct 20 16:51:50 2011 by HappyDoc version r1_5

Table of Contents

removeFilter returns 1 if a filter is removed successfully from a directory in<username>.xml, erroring othe

Class: getDirInfoParser **ui/userData.py**

getDirInfoParser is a private Sax Parser designed to rapidly parse a <username>.xml file for directory/filter names.

This Sax parser is designed to return a list of MadrigalDirectory classes found in a given file. This list is presently used to support the isprint selection web page, which needs to know about all user directories and filters.

For usage information for any Python Sax parser, see Python Sax tutorial

This parser assumes the username.xml is of the form of the following format:

<?xml version='1.0'?>

<userInfo>

<directory>

<dirname>jmh_public</dirname>

<dirtype>public</dirtype>

<filter>
    <filtername>jmh_pub1</filtername>

    more filter elements ...

</filter>

possibly more filters...

</directory>

possibly more directories...

</userInfor>

Non-standard Python modules used:

PyXML

Exceptions thrown:

Exception possibly thrown by sax module

Change history:

Written by Bill Rideout Dec. 11, 2001

Base Classes

xml.sax.handler.ContentHandler

Methods

__init__
characters
endElement
startElement

__init__

```
__init__ ( self )
```

Override of saxutils.DefaultHandler __init__ function to support parsing <username>.xml.

characters

```
characters ( self,  ch )
```

Override of saxutils.DefaultHandler characters function to support parsing <username>.xml.

endElement

```
endElement ( self,  name )
```

Override of saxutils.DefaultHandler endElement function to support parsing <username>.xml.

startElement

```
startElement (
        self,
        name,
        attrs,
        )
```

Override of saxutils.DefaultHandler startElement function to support parsing <username>.xml.

Table of Contents

This document was automatically generated on Thu Oct 20 16:51:50 2011 by HappyDoc version r1_5

Table of Contents

Class: ui/userData.py
getFilterParser

getFilterParser is a private Sax Parser designed to rapidly parse a <username>.xml file for one filter.

This Sax parser is designed to return a list of MadrigalDirectory classes found in a given file. This list is presently used to support the isprint selection web page, which needs to know about all user directories and filters.

For usage information for any Python Sax parser, see Python Sax tutorial

This parser assumes the <username>.xml is of the form of the following format:

<?xml version='1.0'?\>

<userInfo>

<directory>

<dirname>jmh_public</dirname>

<dirtype>public</dirtype>

<filter>
    <filtername>jmh_pub1</filtername>

    more filter elements ...

</filter>

possibly more filters...

</directory>

possibly more directories...

</userInfor>

Non-standard Python modules used:

xml.sax

Exceptions thrown:

Exception possibly thrown by sax module

Change history:

Written by Bill Rideout Dec. 11, 2001

Base Classes

xml.sax.handler.ContentHandler

Methods

__init__
characters
endElement
startElement

__init__

```
__init__ (
        self,
        username,
        dirname,
        filtername,
        )
```

Override of saxutils.DefaultHandler startElement function to support parsing <username>.xml.

characters

```
characters ( self,  ch )
```

Override of saxutils.DefaultHandler characters function to support parsing <username>.xml.

endElement

```
endElement ( self,  name )
```

Override of saxutils.DefaultHandler endElement function to support parsing <username>.xml.

Exceptions

madrigal.admin.MadrigalError("Unable to read filter: " + self.filtername + ' in directory: ' + self.dirname + ' in file: ' + self.username + '.xml', traceback.format_exception(sys.exc_info() [0], sys.exc_info() [1], sys.exc_info() [2]))

startElement

```
startElement (
        self,
        name,
        attrs,
        )
```

	Override of saxutils.DefaultHandler startElement function to support parsing <username>.xml.

Table of Contents

This document was automatically generated on Thu Oct 20 16:51:50 2011 by HappyDoc version r1_5

Table of Contents

Module: web **ui/web.py**

web is the module that interfaces to cgi madrigal web pages.

The web module contains general functions to produce html, along with producing html relating to specific user data or madrigal data.

Classes

MadrigalWeb	MadrigalWeb is the class that produces output for the web.
MadrigalWebFormat	MadrigalWebFormat defines the format of an web interface.

Table of Contents

This document was automatically generated on Fri Dec 30 09:02:26 2005 by HappyDoc version r1_5

Table of Contents

Class: MadrigalWeb

MadrigalWeb is the class that produces output for the web.

All text written to the web is produced in this class.

Non-standard Python modules used: None

Change history:

Written by Bill Rideout Dec. 17, 2001

Methods

__dropLock	getCookieDateOneYearAhead
__getLock	getFirstOwner
__init__	getFirstPrivDir
_getDaynoFilter	getFirstPubDir

__dropLock

```
__dropLock ( self,  filename )
```

__dropLock is a private helper function that drops exclusive locking file.

Inputs: filename = the file that exclusive access is required to.

Returns: None

Affects: Removes file filename + .LCK as a lock mechanism

Exceptions: None.

__getLock

```
__getLock ( self,  filename )
```

__getLock is a private helper function that provides exclusive locking file.

Inputs: filename = the file that exclusive access is required to.

Returns: None

Affects: Writes file filename + .LCK as a lock mechanism

Exceptions: MadrigalError thrown if unable to write lock file

Notes: Will sleep for 1 second at a time, for a maximum of _MaxSleep seconds (prese each second, it will check for the lock file to be removed or modified. If it was modifie counting again. After _MaxSleep counts it then assumes lock file is orphaned and retu dropLock is called.

Exceptions

madrigal.admin.MadrigalError("Unable to open " + filenam

__init__

```
__init__ ( self,  madDB=None )
```

__init__ initializes MadrigalWeb by reading from MadridalDB.

Inputs: Existing MadrigalDB object, by default = None.

Returns: void

Affects: Initializes self.__metaDir, self._logFile.

Exceptions: None.

_getDaynoFilter

```
_getDaynoFilter (
        self,
        seasonalStartDate,
        seasonalEndDate,
        )
```

_getDaynoFilter returns a filter str in the form dayno,<lower>

Inputs: seasonalStartDate - a string in form `MM/DD`. Assumes non-leap year. Empty str
date. seasonalStartDate - a string in form `MM/DD`. Assumes non-leap year. Empty strin
date.

createGlobalIsprintCmd

```
createGlobalIsprintCmd (
        self,
        language,
        madrigalUrl,
        parmList,
        output,
        user_fullname,
        user_email,
        user_affiliation,
        start_datetime,
        end_datetime,
        instCode,
        filterList,
        kindatList,
        expName,
        fileDesc,
        seasonalStartDate,
        seasonalEndDate,
        )
```

createGlobalIsprintCmd returns a string representing a globa a particular language.

Inputs:

language - which language to use. Allowed values are (`python`, `Matlab`, `IDL`) madr
where data is parmList - ordered list of parameters requested. output - output file name

user_affiliation start_datetime - a datetime object. Reject experiments before that datet
Reject experiments after that datetime instCode - instrument code (integer) filterList -
"mnem,lower,upper" where lower and/or upper may be empty kindatList - a list of kin
kindats expName - filter experiments by the experiment name. Matching is case insens
allowed. Empty string is no filtering by experiment name. fileDesc - filter files by the
insensitive and fnmatch characters * and ? are allowed. Empty string is no filtering by
string in form `MM/DD`. Dates before then in any year will be ignored. Assumes non-lea
by seasonal start date. seasonalEndDate - a string in form `MM/DD`. Dates after then in a
non-leap year. Empty string means no filtering by seasonal end date.

Exceptions

ValueError, 'language %s not supported' %(str(language))
ValueError, 'parmList cannot be empty'

filterHtmlFormat

```
filterHtmlFormat ( self,  madFilter )
```

filterHtmlFormat returns a filter formatted as an html table fo

Inputs: a madrigal filter

Returns: a filter formated as an html table for display on the web.

Affects: None.

Exceptions: None.

generateLogout

```
generateLogout (
        self,
        fileName,
        expName,
        )
```

generateLogout generates a java script which sends a user t logout automatically.

Inputs: fileName: the madrigal file to return to expName: the experiment name of the f

Returns: a java script which sends a user to the madLogin page to logout automatically

Affects: None.

Exceptions: None.

getCgiString

```
getCgiString ( self,  urlStr )
```

getCgiString returns a string made safe for urls by using urlli

Inputs: urlStr - a url string that may contain unsafe url characters

Returns: a string as returned by urllib.quote

Affects: None.

Exceptions: None.

getCgiStringFromList

```
getCgiStringFromList ( self,  list )
```

getCgiStringFromList returns a string based on a list with ite

Inputs: list - a python list

Returns: a string based on list with items separated by %20's

Affects: None.

Exceptions: None.

getCookieDateOneYearAgo

```
getCookieDateOneYearAgo ( self )
```

getCookieDateOneYearAgo returns a string formated as per c year in the past.

Inputs: none

Returns: a string formated as per cookie standard of time one year in the past. Used to deleted

Affects: None

Exceptions: None.

getCookieDateOneYearAhead

```
getCookieDateOneYearAhead ( self )
```

generateLogout generates a java script which sends a user to the madLogin page tologout automatically.

getCookieDateOneYearAhead returns a string formated as pe one year in future.

Inputs: none

Returns: a string formated as per cookie standard of time one year in future. Used to se persisted.

Affects: None

Exceptions: None.

getFirstOwner

```
getFirstOwner ( self,  madUserDataObj )
```

getFirstOwner returns the first owner name of a public direct new file is loaded.

Inputs: None

Returns: the first owner name of a public directory - used when a new file is loaded. E

Affects: None.

Exceptions: None.

getFirstPrivDir

```
getFirstPrivDir (
        self,
        madUserDataObj,
        username,
        )
```

getFirstPrivDir returns the first private directory name found new file is loaded.

Inputs: None

Returns: the first private directory name found - used by default when a new file is loa exist.

Affects: None.

Exceptions: None.

getFirstPubDir

```
getFirstPubDir ( self,  madUserDataObj )
```

getFirstPubDir returns the first public directory name found - file is loaded.

Inputs: None

Returns: the first public directory name found - used by default when a new file is load exist.

Affects: None.

Exceptions: None.

getFirstUserDir

```
getFirstUserDir (
        self,
        madUserDataObj,
        username,
        )
```

getFirstUserDir returns the first directory name found for a g when save filter is loaded.

Inputs: None

Returns: the first directory name found owned by a given user - used by default when s no public directories exist.

Affects: None.

Exceptions: None.

getIntListFromStrList

```
getIntListFromStrList ( self,  strList )
```

getIntListFromStrList returns a list containing integers given form

Inputs: strList - a list a strings that can be converted to integers

Returns: a list containing integers given a list of integers in string form

Affects: None

Exceptions: MadrigalError thrown if a string in the list cannot be converted to an integ

Exceptions

madrigal.admin.MadrigalError("Unable to convert following
traceback.format_exception(sys.exc_info() [0], sys.exc_inf

getOptionAllUserDirTags

```
getOptionAllUserDirTags (
        self,
        madUserDataObj,
        filterOwner,
        dirName='',
        )
```

getOptionAllUserDirTags outputs a series of html option tags all dirs owned by a user

Inputs: madUserDataObj - an object of type MadrigalUserData filterOwner - a string r owner

Returns: a string containing option tags of form <option value="<dirName> (public)"> is appended to value based on dir type. If dirName is passed in and found to match, the first row will be selected

Affects: None.

Exceptions: None.

getOptionFilterTags

```
getOptionFilterTags (
        self,
        madUserDataObj,
        filterOwner,
        filterDir,
        filterName='',
        )
```

getOptionFilterTags outputs a series of html option tags with filters

Inputs: madUserDataObj - an object of type MadrigalUserData filterOwner - a string r owner filterDir - a string representing the name of the private directory filterName - a s selected - if blank or not found, select first

Returns: a string containing option tags of form <option value="<filterName>"><filter found to match, then that filter will be selected. Otherwise, the first row will be selecte

Affects: None.

Exceptions: None.

getOptionKindatListTags

```
getOptionKindatListTags (
        self,
        optionList,
        selectedNumber=None,
        )
```

getOptionKindatListTags outputs a series of html option tags value="<kindat num>"><kindat description>..

Inputs: optionList - a list of kindat numbers to be options selectedNumber - the numbe None, none selected

Returns: a string containing option tags of form <option value="<kindatt num>"><kin

Affects: None.

Exceptions: None.

getOptionKinstListTags

```
getOptionKinstListTags (
        self,
        optionList,
        selectedNumber=None,
        )
```

getOptionKinstListTags outputs a series of html option tags value="<kinst num>"><kinst description>..

Inputs: optionList - a list of kinst numbers to be options selectedNumber - the number None, none selected

Returns: a string containing option tags of form <option value="<kinst num>"><kinst

Affects: None.

Exceptions: None.

getOptionListTags

```
getOptionListTags (
        self,
        optionList,
        selectedNumber=None,
        )
```

getOptionListTags outputs a series of html option tags with r list

Inputs: optionList - a list of items to be options selectedNumber - the number of the ite
none selected

Returns: a string containing option tags of form <option value="<item>"><item>...

Affects: None.

Exceptions: None.

getOptionMonthTags

```
getOptionMonthTags ( self,  selectedMonth )
```

getOptionMonthTags outputs a series of html option tags wit of the year

Inputs: selectedMonth - an integer (1-12) representing the month to be selected

Returns: a string containing option tags of form <option value="1">Jan<option value=

Affects: None.

Exceptions: None.

getOptionNumericTags

```
getOptionNumericTags (
        self,
        startNum,
        endNum,
        selectedNum,
        )
```

getOptionTags outputs a series of html option tags with num to endNum

Inputs: startNum - integer value of first option tag endNum - integer value of last optio
which tag is selected

Returns: a string containing option tags of form <option value="1">1<option value="2

Affects: None.

Exceptions: None.

getOptionPrivDirTags

```
getOptionPrivDirTags (
        self,
        madUserDataObj,
        filterOwner,
        filterDir='',
```

```
        )
```

getOptionDirTags outputs a series of html option tags with re directories

Inputs: madUserDataObj - an object of type MadrigalUserData filterOwner - a string r owner filterDir - a string representing the name of the private directory

Returns: a string containing option tags of form <option value="<dirName>"><dirNan passed in and found to match, then that dir will be selected. Otherwise, the first row wi

Affects: None.

Exceptions: None.

getOptionPubDirTags

```
getOptionPubDirTags (
        self,
        madUserDataObj,
        filterOwner='',
        filterDir='',
        )
```

getOptionDirTags outputs a series of html option tags with re directories

Inputs: madUserDataObj - an object of type MadrigalUserData filterOwner - a string r owner filterDir - a string representing the name of the directory

Returns: a string containing option tags of form <option value="<userName:dirName> filterOwner and filterDir are passed in and found to match, then that dir will be selecte selected

Affects: None.

Exceptions: None.

getRulesOfTheRoad

```
getRulesOfTheRoad ( self )
```

getRulesOfTheRoad returns a string giving the rules in html f data.

Inputs: None

Returns: a string giving the rules in html formal for using madrigal data

Affects: None.

Exceptions: None.

getSpaceString

```
getSpaceString ( self,  cgiString )
```

getSpaceString returns a string based on urllib.unquote

Inputs: cgiString - a string that may contain characters substituted by urllib.quote

Returns: a string possibly modified by urllib.unqoute

Affects: None.

Exceptions: None.

isTrusted

```
isTrusted ( self )
```

isTrusted returns 1 if browser ip matches any in the trustedI

Inputs: None

Returns: 1 if browser ip matches any in the trustedIPs.txt file; 0 otherwise. Also return
trustedIPs.txt cannot be opened.

Affects: None.

Exceptions: None.

logDataAccess

```
logDataAccess (
        self,
        fullFilenameList,
        user_fullname=None,
        user_email=None,
        user_affiliation=None,
        )
```

logDataAccess logs queries that access low-level data.

Records user name, email, affiliation, datetime, and full path the file(s) accessed.

Inputs:

fullFilenameList either a list of full filenames, or a string with one filename

user_fullname - if None, try to read from cookie. Also, any commas replaced by space

user_email - if None, try to read from cookie. Also, any commas replaced by spaces.

user_affiliation - if None, try to read from cookie. Also, any commas replaced by spac

Outputs: None

Affects: Write line to log file with 5 or more comma-delimited columns. Example:

Bill Rideout,brideout@haystack.mit.edu,MIT Haystack,2002-12-25 00:00:00,
/opt/madrigal/experiments/2005/mlh/01sep05/mlh050901g.001,/opt/madrigal/experim

Uses __getLock and __dropLock to ensure single users access to log file

outputHead

```
outputHead ( self,  title )
```

outputHead outputs a standard html <head> with the given ti

Inputs: title - string : title of cgi page

Returns: a standard html <head> with the given title. This function also defines the "lb
changing that format here changes it in all madrigal python cgi pages.

Affects: None.

Exceptions: None.

outputIsprintReport

```
outputIsprintReport (
        self,
        madFilter,
        madForm,
        filename,
        )
```

outputIsprintReport prints a isprint-formated maddata object

Works by calling madrigal._Madrec.getIsprintReport

Inputs: 1) madFilter = the MadrigalFilter that determines all the settings.

1. madForm - a cgi.FieldStorage object representing the submitted form
2. filename = the full path to the madrigal file to use

Returns: None

Affects: prints a isprint-formated maddata object given a madrigal filter.

Exceptions: None.

Exceptions

madrigal.admin.MadrigalError("Illegal mnemonic operator
madrigal.admin.MadrigalError("Illegal mnemonic string: "
madrigal.admin.MadrigalError("Illegal mnemonic string: "

Table of Contents

This document was automatically generated on Thu Oct 20 16:51:50 2011 by HappyDoc version r1_5

Table of Contents

Class: MadrigalWebFormat **ui/web.py**

MadrigalWebFormat defines the format of an web interface.

Information about how a web page is formatted is stored in this class. In particular, the possible derived parameters to display for a given format (such as Short or Comprehensive) are set in this class. Edit this class to create new formats or modify existing ones.

Non-standard Python modules used: None

No exceptions thrown

Change history:

Written by Bill Rideout Oct. 29, 2001

Methods

getFormat

getFormat

```
getFormat ( self,  formatName )
```

Table of Contents

This document was automatically generated on Fri Dec 30 09:02:26 2005 by HappyDoc version r1_5

Inputs: 1) madFilter = the MadrigalFilter that determines all the settings.

Madrigal C API

The madrec library (libmadrec.a) contains a comprehensive set of C-language procedures for working with Cedar Database files (the madrec module), and recently has been upgraded to work at a higher data level with both measured and derived data (the maddata module).

The madrec library (libmadrec.a) contains a comprehensive set of C-language procedures for working with Cedar Database files. Five version of CEDAR file are supported - Madrigal, Blocked Binary, Cbf, Unblocked binary and Ascii. In addition, an entire file may be read into memory for rapid random record access. When opening a file for reading, the madrec package is able to determine the file version automatically. Data, header and catalog records are all supported. The Cedar format itself is big-endian, but Madrec is endian-neutral and works on little endian as well as big endian computers.

The maddata module allows an application writer to ignore the difference between measured and derived parameters - from the maddata level, any file can be assumed to contain its measured data and all possible derived data. The maddata module is also designed to be easily extended to derive new parameters.

The following are suggested lines to add to your Makefile when using the Madrigal C API:

```
# Library directory  LIBDIR = $(MADROOT)/lib    # Include directory  INCLUDEDIRS = -I. -I$(MA
```

madrec and maddata procedures

The madrec library (libmadrec.a) contains a comprehensive set of C-language procedures for working with Cedar Database files (the madrec module), and at a higher data level with both measured and derived data (the maddata module). As of release 2.2, all Madrigal software is built on the madrec C library.

Overview of file-level madrec API

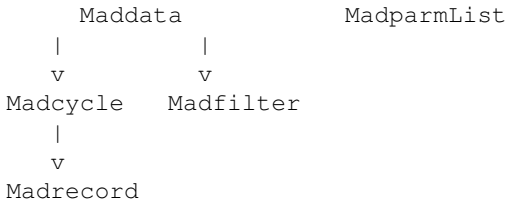
As detailed in the synopsis of madrecOpen, five version of CEDAR file are supported - Madrigal, Blocked Binary, Cbf, Unblocked binary and Ascii. In addition, and entire file may be read into memory for rapid random record access. When opening a file for reading, the madrec package is able to determine the file version automatically. Data, header and catalog records are all supported. The Cedar format itself is big-endian, but Madrec is endian-neutral and works on little endian as well as big endian computers.

All methods in madc that return an array of any sort (char, int, or double) allocate the returned array from the heap. The user of this library is responsible for freeing these arrays when done. Some methods, such as cedarGetGeodetic, also allocate arrays for pointers passed in as arguments. This is the case when the method is designed to return more than one array, as is the case with cedarGetGeodetic. See the documentation of the individual methods for details.

Overview of data level maddata API

The maddata level exposes Madrigal in such a way that the user does not need to worry about whether data is measured in a file or derived. The user needs only deal with the abstact maddata data structure.The struct maddata is intented to provide easy access to madrigal data from a single cedar file that has been combined with derived parameters and has been filtered. The maddata module can also be used without files by passing in data directly needed to calculate other parameters.

The data is organized as follows:



All data in this structure corresponds to Madrigal parameters, and so is referenced by its unique mnemonic, not by Cedar parameter codes. All data is stored as doubles, with special values as defined in cedar.h.

While this module is written in C and not C++; its methods and design are as close as I could get to object-oriented. Every data structure should be instantiated via a create* method and released via destroy*. All other methods take the respective data structure pointer as the first argument. See usage in simpleMaddata.c.

The Madrecord structure defined in this file differs from the Madrec structure defined in madrec.h in that the Madrecord struct does not care about the Cedar file format, or indeed in what way the data is stored. It's basic unit of data is a double, not the 16 bit Int as in the Cedar format.

The derivation engine behind this interface is defined in the private modules madDeriveEngine and madDeriveMethods. Extending maddata simply involves adding new methods (and possibly parameters), as fully explained in madDeriveMethods.h.

See the files simpleMaddata.c and simpleNonfileMaddata.c for example usage, or the online examples.

The madrec module

madrecCreate	madrecDestroy	madrecOpen	madrecClose	mad
madrecPutNextRec	madrecRewind	madrecGetPreviousRec	madrecGetRecByRecno	mad
madrecGenKeys	madrecDeleteKeys	madrecPrintKeys	madrecCheckFile	mad
madrecGetFileType	madrecSetError	madrecGetError	madrecGetMissing	mad
madrecGetParmsList	madrecGetParmLoc	madrecGetParmMin	madrecGetParmMax	mad
madrecHasHeader	madrecGetSortedRecnoList	compareCedarIndices	cedarGetMadroot	ced
cedarGetKrec	isDataRecord	cedarGetKinst	cedarGetKindat	ced
cedarGetIbdt	cedarGetIbhm	cedarGetIbcs	cedarGetIeyr	ced
cedarGetIehm	cedarGetIecs	cedarGetLprol	cedarGetJpar	ced

cedarGetNrow	cedarGetKpar	cedarGetWord	cedarGetStartTime	ced
cedarGetStartJday	cedarGetEndJday	cedarGetStartIndex	cedarGetEndIndex	ced
cedarGet2dParcodes	cedarHas1DParcode	cedarHas2DParcode	cedarGet1dParm	ced
cedarGet2dParmValue	cedarGetFlatParm	hasData	cedarGetParmCodeArray	ced
cedarGetGeodetic	cedarGet1dInt	cedarGet2dInt	cedarGet2dIntValue	ced
cedarCreateCatalogRecord	cedarAppendCatalogRecord	cedarCreateHeaderRecord	cedarAppendHeaderRecord	ced
cedarSetKinst	cedarSetKindat	cedarSetStartTime	cedarSetEndTime	ced
cedarSetNorm1dParm	cedarSet2dParm	cedarSetNorm2dParm	cedarSet1dInt	ced
cedarPrintRecord	cedarGetInformation	cedarPrintProlog	cedarReadParCodes	ced
cedarGetParCode	cedarGetParCodeIndex	madGetParMnemIndex	isMadparmError	getS
cedarGetParCodeType	madGetParMnemType	madGetCategoryIndex	cedarGetParDescription	mad
cedarGetParInt16Description	madGetParInt16Description	cedarGetParScaleFactor	madGetParScaleFactor	ced
madGetNormScaleFactor	cedarGetParUnits	madGetParUnits	cedarGetParMnemonic	ced
cedarGetParFormat	madGetParFormat	cedarGetParWidth	madGetParWidth	ced
madHasHtmlDesc	cedarCheckRecord	cedarHexPrintRecord	cedarDecimalPrintRecord	ced
cedarGetError	cedarTabInt	cedarUpdateParmsList	cedarGetStationPos	ced
searchFilesByDate	loadExpFileTable	goodDataExists	sprod	vad
vsub	csconv	vctcnv	point	lool
convrt	rpcart	gdv	los2geodetic	sola
solardist	shadowheight	sunrise_set	jday	jdat
idmyck	dmadptr	getKey	dinvmadptr	mad

```
/**************************************************************************
*
* madrecCreate     creates a madrec object
*
*   arguments:
*       None
*
*   returns:
*       pointer to the new madrec object
*
*/

Madrec *
madrecCreate ()



/**************************************************************************
*
* madrecDestroy    destroys a madrec object
*
*   arguments:
*       madrecp - pointer to the madrec object
*
*   returns
*       0
*
*/

int
```

```
madrecDestroy (Madrec *madrecp)



/*****************************************************************
*
* madrecOpen     opens a madrec data file
*
*   arguments:
*       madrecp - pointer to the madrec object
*       iotype  - file type as described below
*       filnam  - file name
*
*   The following file types (iotype) are supported:
*
*       Open Cedar file for sequential reading:
*             1 - Determine file type automatically
*            10 - Madrigal file
*            11 - Blocked Binary file
*            12 - Cbf file
*            13 - Unblocked Binary file
*            14 - Ascii file
*
*       Create Cedar file for update; discard previous contents if any:
*             2 - Madrigal file
*            20 - Madrigal file
*            21 - Blocked Binary file
*            22 - Cbf file
*            23 - Unblocked Binary file
*            24 - Ascii file
*
*       Create Cedar file in memory for sequential and random read and write.
*            30 - Determine file type automatically
*            40 - Madrigal file
*            41 - Blocked Binary file
*            42 - Cbf file
*            43 - Unblocked Binary file
*            44 - Ascii file
*
*       Fast create Cedar file in memory for sequential and random read and write.
*          Does not calculate min and and max parameter values
*            50 - Determine file type automatically
*            60 - Madrigal file
*            61 - Blocked Binary file
*            62 - Cbf file
*            63 - Unblocked Binary file
*            64 - Ascii file
*
*   returns:
*       0 - File opened successfully
*       1 - Invalid file type (iotype)
*       2 - unable to open data file
*       3 - data file already open
*       4 - input file name too long
*       5 - error writing file to memory
*
*/

int
madrecOpen (Madrec *madrecp, int iotype, char *filnam)
```

```
/*****************************************************************
*
* madrecClose    closes a madrec data file
*
*   arguments:
*       madrecp - pointer to the madrec object
*
*   returns:
*       0 - File closed successfully
*       1 - error closing file
*       2 - file not open
*       3 - error flushing file
*
*/

int
madrecClose (Madrec *madrecp)



/*****************************************************************
*
* madrecGetNextRec   reads a cedar record and fills a Madrec structure
*                    with the information in the record.
*
*   arguments:
*       madrecp - pointer to the madrec object
*
*   returns:
*       0 - Record read successfully
*       1 - Illegal file type (bad iotype in madrec)
*      -n - Error in CedarIO package
*
*/

int
madrecGetNextRec (Madrec *madrecp)



/*****************************************************************
*
* madrecPutNextRec   writes a cedar record.
*
*   arguments:
*       madrecp - pointer to the madrec object
*
*   returns:
*       0 -
*       1 - Illegal file type (bad iotype in madrec)
*      -n - Error in CedarIO package
*
*/

int
madrecPutNextRec (Madrec *madrecp)
```

```
/*****************************************************************
*
* madrecRewind   rewinds a cedar record.
*
*   arguments:
*       madrecp - pointer to the madrec object
*
*   returns:
*       0 -
*       1 - Illegal file type (bad iotype in madrec)
*      -n - Error in CedarIO package
*
*/

int
madrecRewind (Madrec *madrecp)



/*****************************************************************
*
* madrecGetPreviousRec   reads an madrigal record and fills a Mad
*                        structure with the information in the record.
*
*   arguments:
*       madrecp - pointer to the madrec object
*
*   returns:
*       0 -
*
*/

int
madrecGetPreviousRec(Madrec *madrecp)



/*****************************************************************
*
* madrecGetRecByRecno   reads a madrigal record and fills a Mad
*                          structure with the information in the record.
*                          The record number is specified by the second
*                          argument. The first record is recno=0.
*
*   arguments:
*       madrecp - pointer to the madrec object
*       recno   - the record number to get. The first record is recno=0.
*
*   returns:
*       0 - Record read successfully
*      -1 - Specified record not in file
*
*/

int
madrecGetRecByRecno(Madrec *madrecp, int recno)



/*****************************************************************
*
```

```
* madrecGetRecordByKey   reads an madrigal record and fills a Mad
*                        structure with the information in the record.
*                        The record is the first data record for which key is
*                        greater than or equal to the start key of the
*                        record, and less than the start time of the
*                        following record. Thus, if the specified key
*                        corresponds to a time within a record, the
*                        first such record is returned. Header or catalog
*                        records are never returned.
*
*
*   arguments:
*       madrecp - pointer to the madrec object
*       key - time in seconds since 1/1/1950
*
*   returns:
*       0 - if record found
*      -1 - if record not found
*
*/

int
madrecGetRecByKey(Madrec *madrecp, double key)



/*****************************************************************
*
* madrecGenKeys   Generates madrec key array. All information needed to
*                 access records randomly by key or record number is
*                 saved.
*
*   arguments:
*       madrecp - pointer to the madrec object
*
*   returns:
*       0 -
*
*/


int
madrecGenKeys (Madrec *madrecp)



/*****************************************************************
*
* madrecDeleteKeys   Deletes madrec key array.
*
*   arguments:
*       madrecp - pointer to the madrec object
*
*   returns:
*       0 -
*
*/


int
madrecDeleteKeys (Madrec *madrecp)
```

```
/*****************************************************************
*
* madrecPrintKeys   Prints madrec file key table
*
*   arguments:
*       madrecp - pointer to the madrec object
*
*   returns:
*       0 -
*
*/


int
madrecPrintKeys (Madrec *madrecp)



/*****************************************************************
*
* madrecCheckFile    checks the structure of a madrec data file
*
  Block (Physical record) structure:

  6720 16bit words (Int16)

  word[0]       = Total number of significant words in the block
                  record (all blocks are 13440 bytes long)

  word[1]       = Pointer to the first word of the first logical
                  record contained in the block.

                  (set to zero if the block doesn't contain
                   any complete logical records i.e. it just
                   contains the last part of a logical record.)

  word[2]       = Pointer to the first word of the last logical
                  record contained in block.

  word[word[0]-1] = Checksum.

  Logical Records:

  word[0 - 15]  = Same as NCAR binary logical records.
  word[16]      = Pointer to word 1 of previous logical record.
                   -could be contained in previous block.
                   -Set to zero in the first logical record of the file.
*/

int
madrecCheckFile(Madrec *madrecp)



/*****************************************************************
*
* madrecCopy   Copies logical record from one madrec object to another
*
```

```
*   arguments:
*       madrec1p - pointer to the source madrec object
*       madrec2p - pointer to the destination madrec object
*
*   returns:
*       0 - Record copied successfully
*       1 - Empty source record
*
*/

int
madrecCopy (Madrec *madrec1p, Madrec *madrec2p)



/*****************************************************************
*
* madrecGetFileType   Gets file type
*
*   arguments:
*       madrecp - pointer to madrec object
*
*/
int
madrecGetFileType (Madrec *madrecp)



/*****************************************************************
*
* madrecSetError   sets madrec error
*
*   arguments:
*       madrecp - pointer to the madrec object
*
*   returns:
*       0 -
*
*/


int
madrecSetError (Madrec *madrecp, const char *error)



/*****************************************************************
*
* madrecGetError   gets last madrec error
*
*   arguments:
*       madrecp - pointer to the madrec object
*
*   returns:
*       error string
*
*/

char *
madrecGetError (Madrec *madrecp)
```

```
/*****************************************************************
*
* madrecGetMissing   gets missing data value
*
*   arguments:
*       madrecp - pointer to the madrec object
*
*   returns:
*       double precision missing value
*
*/

double
madrecGetMissing (Madrec *madrecp)



/*****************************************************************
*
* madrecGetNumParms   gets number of different parameters in file
*
*   arguments:
*       madrecp - pointer to the madrec object
*
*   returns:
*       number of distint parameters
*
*/

int
madrecGetNumParms (Madrec *madrecp)



/*****************************************************************
*
* madrecGetParmsList   gets list (int array) of different parameters
*                      codes in file
*
*   arguments:
*       madrecp - pointer to the madrec object
*
*   returns:
*       codes of distinct parameter
*
*  This returned array is a pointer to an internal structure in Madrec;
*  it does not need to be freed by the user.
*
*/

int *
madrecGetParmsList (Madrec *madrecp)



/*****************************************************************
*
* madrecGetParmLoc   gets location of parameter
*
```

```
*   arguments:
*       madrecp - pointer to the madrec object
*
*   returns:
*       parameter location:
*           1 - 1D array
*           2 - 2D array
*           3 - Derived
*
*/

int *
madrecGetParmLoc (Madrec *madrecp)



/**********************************************************************
*
* madrecGetParmMin   gets minimum value of parameter
*
*   arguments:
*       madrecp - pointer to the madrec object
*
*   returns - Minimum value of parameter in entire file
*
*/

double *
madrecGetParmMin (Madrec *madrecp)



/**********************************************************************
*
* madrecGetParmMax   gets maximum value of parameter
*
*   arguments:
*       madrecp - pointer to the madrec object
*
*   returns - Maximum value of parameter in entire file
*
*/

double *
madrecGetParmMax (Madrec *madrecp)



/**********************************************************************
*
* madrecHasCatalog   returns 1 if file has catalog record, 0 otherwise
*
*   arguments:
*       madrecp - pointer to the madrec object
*
*   returns - 1 if file has catalog record, 0 otherwise
*
*/

int madrecHasCatalog(Madrec *madrecp)
```

```
/*****************************************************************
*
* madrecHasHeader   returns 1 if file has header record, 0 otherwise
*
*   arguments:
*       madrecp - pointer to the madrec object
*
*   returns - 1 if file has header record, 0 otherwise
*
*/

int madrecHasHeader(Madrec *madrecp)



/*****************************************************************
*
* madrecGetSortedRecnoList   gets int array of recno's sorted by start time key
*
*   arguments:
*       madrecp - pointer to the madrec object
*
*   returns - int array of recno's sorted by start time key, length = nrecords
*
*/

int * madrecGetSortedRecnoList (Madrec *madrecp)



/*****************************************************************
*
* compareCedarIndices   a private method to compare one CedarIndex to another
*
*   arguments:
*       void * cedarIndex1 - void pointer to the first CedarIndex
*       void * cedarIndex2 - void pointer to the second CedarIndex
*
*   returns - if cedarIndex1 before cedarIndex2, return -1
*             if cedarIndex1 same as cedarIndex2, return 0
*             if cedarIndex1 after cedarIndex2, return 1
*
*/
int compareCedarIndices(const void * index1, const void * index2)



/*****************************************************************
*
* cedarGetMadroot   copies Madroot path into user-supplied character
*                   buffer.
*
*   Simply calls getenv, if not found, uses #define __MAD_ROOT__
*
*   Returns  void
*/
void cedarGetMadroot(char * buf)
```

```
/*****************************************************************
*
* cedarGetLtot   gets length of record
*
*/

int cedarGetLtot (Int16 *cedarp)



/*****************************************************************
*
* cedarGetKrec   gets Kind of record
*
*/

int cedarGetKrec (Int16 *cedarp)



/*****************************************************************
*
* isDataRecord   returns 1 if data record, 0 if catalog or header
*
*/

int isDataRecord(Int16 *cedarp)



/*****************************************************************
*
* cedarGetKinst   gets instrument code for these data
*
*/

int cedarGetKinst (Int16 *cedarp)



/*****************************************************************
*
* cedarGetKindat   gets kind-of-data code
*
*/

int cedarGetKindat (Int16 *cedarp)



/*****************************************************************
*
* cedarGetIbyr   gets beginning year
*
*/

int cedarGetIbyr (Int16 *cedarp)
```

```
/*****************************************************************
*
* cedarGetIbdt   gets beginning date (100*month+day)
*
*/

int cedarGetIbdt (Int16 *cedarp)



/*****************************************************************
*
* cedarGetIbhm   gets beginning hour and minute (100*hour + minute)
*
*/

int cedarGetIbhm (Int16 *cedarp)



/*****************************************************************
*
* cedarGetIbcs   gets beginning centisecond
*
*/

int cedarGetIbcs (Int16 *cedarp)



/*****************************************************************
*
* cedarGetIeyr   gets ending year
*
*/

int cedarGetIeyr (Int16 *cedarp)



/*****************************************************************
*
* cedarGetIedt   gets ending date (100*month+day)
*
*/

int cedarGetIedt (Int16 *cedarp)



/*****************************************************************
*
* cedarGetIehm   gets ending hour and minute (100*hour + minute)
*
*/

int cedarGetIehm (Int16 *cedarp)



/*****************************************************************
```

```
*
* cedarGetIecs   gets ending centisecond
*
*/

int cedarGetIecs (Int16 *cedarp)



/*****************************************************************
*
* cedarGetLprol   gets prolog length
*
*/

int cedarGetLprol (Int16 *cedarp)



/*****************************************************************
*
* cedarGetJpar   gets number of single-valued parameters
*
*/

int cedarGetJpar (Int16 *cedarp)



/*****************************************************************
*
* cedarGetMpar  gets number of multiple-valued parameters
*
*/

int cedarGetMpar (Int16 *cedarp)



/*****************************************************************
*
* cedarGetNrow   gets number of entries for each multiple-valued parameter
*
*/

int cedarGetNrow (Int16 *cedarp)



/*****************************************************************
*
* cedarGetKpar  gets number of derived parameters
*
*  Deprecated - use Maddata module for all derived data
*
*/

int cedarGetKpar (Int16 *cedarp)
```

```
/*****************************************************************
*
* cedarGetWord   gets specified word from cedar record
*
*/

int cedarGetWord (Int16 *cedarp, int word)



/*****************************************************************
*
* cedarGetStartTime   gets start time of record
*
*/

int cedarGetStartTime (Int16 *cedarp, int *year, int *month, int *day,
                   int *hour, int *minute, int *second, int *centisecond)



/*****************************************************************
*
* cedarGetEndTime   gets end time of record
*
*/

int cedarGetEndTime (Int16 *cedarp, int *year, int *month, int *day,
                   int *hour, int *minute, int *second, int *centisecond)



/*****************************************************************
*
* cedarGetStartJday   gets start Julian Day plus fractioanl day
*
*/

double cedarGetStartJday (Int16 *cedarp)



/*****************************************************************
*
* cedarGetEndJday   gets end Julian Day plus fractioanl day
*
*/

double cedarGetEndJday (Int16 *cedarp)



/*****************************************************************
*
* cedarGetStartIndex   gets start index time of record
*
*/

double cedarGetStartIndex (Int16 *cedarp)
```

```
/*****************************************************************
*
* cedarGetEndIndex   gets end index time of record
*
*/

double cedarGetEndIndex (Int16 *cedarp)



/*****************************************************************
*
* cedarGet1dParcodes  gets 1D parameter codes from a madrigal record
*
*  This methods allocates dynamic memory for the array of ints
*  returned.  The caller of this method is responsible for
*  calling free to release this memory when finished with it.
*
*  If no 1D parameter codes, returns NULL pointer.
*
*/

int * cedarGet1dParcodes(Int16 *cedarp)



/*****************************************************************
*
* cedarGet2dParcodes  gets 2D parameter codes from a madrigal record
*
*  This methods allocates dynamic memory for the array of ints
*  returned.  The caller of this method is responsible for
*  calling free to release this memory when finished with it.
*
*  If no 2D parameter codes, returns NULL pointer.
*/

int * cedarGet2dParcodes(Int16 *cedarp)



/*****************************************************************
*
* cedarHas1DParcode  returns 1 if cedarp has particular 1D parcode, 0 otherwise.
*
*/
int cedarHas1DParcode(Int16 *cedarp, int parcode)



/*****************************************************************
*
* cedarHas2DParcode  returns 1 if cedarp has particular 2D parcode, 0 otherwise.
*
*/
int cedarHas2DParcode(Int16 *cedarp, int parcode)



/*****************************************************************
```

```
*
* cedarGet1dParm   gets a scaled 1D parameter from a madrigal record
*
*  If 1D parm does not exist, returns double "missing"
*  If 1D parm = -32767 (missing), returns double "missing"
*  If 1D parm is an error code, and = -32766 (assumed), returns double "assumed"
*  If 1D parm is an error code, and = +32767 (known bad), returns double "knownbad"
*
*  Otherwise, scales value and includes additional increment values
*  if they exist
*
*  If cedarp is a header or catalog record; warning is printed to std err
*  and returns "missing"
*
*/

double cedarGet1dParm(Int16 *cedarp, int parcode)



/*****************************************************************
*
* cedarGet2dParm   gets a scaled 2D parameter from a madrigal record
*
*  If 2D parm does not exist, returns array of double "missing"
*  If 2D parm = -32767 (missing), returns double "missing"
*  If 2D parm is an error code, and = -32766 (assumed), returns double "assumed"
*  If 2D parm is an error code, and = +32767 (known bad), returns double "knownbad"
*
*  Otherwise, scales value and includes additional increment values
*  if they exist
*
*  This methods allocates dynamic memory for the array of doubles
*  returned.  The caller of this method is responsible for
*  calling free to release this memory when finished with it.
*
*  If nrow = 0, returns NULL pointer.
*
*  If parcode not found, returns array of missing
*
*  If cedarp is a header or catalog record; warning is printed to std err
*  and returns NULL
*/

double * cedarGet2dParm(Int16 *cedarp, int parcode)



/*****************************************************************
*
* cedarGet2dParmValue   gets a single scaled 2D parameter from a madrigal record
*
*  If 2D parm does not exist, returns double "missing"
*  If 2D parm = -32767 (missing), returns double "missing"
*  If 2D parm is an error code, and = -32766 (assumed), returns double "assumed"
*  If 2D parm is an error code, and = +32767 (known bad), returns double "knownbad"
*  If row is greater than number of 2d rows, returns double "missing"
*
*  Otherwise, scales value and includes additional increment values
*  if they exist
*
```

```
*  This method differs from cedarGet2dParm in that it only returns a
*  single double from a single row, so no malloc/free is required.
*
*
*  If cedarp is a header or catalog record; warning is printed to std err
*  and returns missing
*/

double cedarGet2dParmValue(Int16 *cedarp, int parcode, int row)



/*****************************************************************
*
* cedarGetFlatParm   creates a flattened 2D parameter
*
*  If 1D parmeter exists, copies array of double of length nrow with
*  every value set to the 1D value.  If not, uses cedarGet2dParm.  Note
*  cedarGet2dParm returns all "missing" is parm not found.
*
*  This methods allocates dynamic memory for the array of doubles
*  returned.  The caller of this method is responsible for
*  calling free to release this memory when finished with it.
*
*  If nrow = 0, returns NULL pointer.
*
*  If parcode not found, returns array of missing
*/

double * cedarGetFlatParm(Int16 *cedarp, int parcode)



/*****************************************************************
*
* hasData   determines whether any non-missing data in a double array
*
*  Returns 0 if all data in 2d array of length nrow is missing, 1
*  otherwise.
*/

int hasData(int nrow, double * parp)



/*****************************************************************
*
* cedarGetParmCodeArray   gets parameter codes of all parameters
*                         in specp->pparms which are actually available
*                         from the current record.
*
*  User is responsible for calling free to release the returned
*  array of ints when finished with them.
*
*  Deprecated - use Maddata module instead
*/

int * cedarGetParmCodeArray(Int16 *cedarp, Ffspec *specp, int *nlines) {
```

```
/*****************************************************************
*
* cedarGetParmArray   flattens a subset of a CEDAR file
*
*  If record is rejected, nlinesp will be set to 0; returned
*     double array will be set to random values.
*
*  User is responsible for calling free to release the returned
*  array of doubles when finished with them.
*
*  Deprecated - use Maddata module
*/

double * cedarGetParmArray(Int16 *cedarp, Ffspec *specp, int *nlinesp)



/*****************************************************************
*
* cedarGetGeodetic   gets geodetic coordinates from radar coordinates
*
*  cedarGetGeodetic modifies the three arrays of doubles to return
*  lat, long, and alt.  Length of each array is nrows.  Geodetic
*  coordinates will be calculated in any of the following ways:
*
*     1) az, el, and range - az from azm, az1, or az2, and
*        el from elm, el, or el2
*     2) (altb, alte, or gdalt), gdlat and glon
*     3) (altb, altav or altb, alte, or gdalt) alone - lat and long assumed
*        to be that of instrument
*
*     If all three methods fail, all three arrays populated with missing
*
*     All parameters can be either 1d or 2d.
*
*  This methods allocates dynamic memory for the array of doubles
*  modified.  The caller of this method is responsible for
*  calling free to release the memory from these 3 arrays when
*  finished with them.
*
*  If nrow = 0, returns -1.
*/

int cedarGetGeodetic(Int16 *cedarp, double **gdlatpp, double **glonpp, double **gdaltpp)



/*****************************************************************
*
* cedarGet1dInt   gets a 1D parameter (unscaled) from a madrigal record
*
*/

Int16 cedarGet1dInt(Int16 *cedarp, int parcode)



/*****************************************************************
*
* cedarGet2dInt   gets a 2D parameter (unscaled) from a madrigal record
*
```

```
*  This method allocates dynamic memory for the array of ints
*  returned.  The caller of this method is responsible for
*  calling free to release this memory when finished with it.
*
*  If nrow = 0, returns NULL pointer.
*
*  If parcode not found, returns array of missingData
*/

Int16 * cedarGet2dInt(Int16 *cedarp, int parcode)



/*****************************************************************
*
* cedarGet2dIntValue   gets a single 2D parameter (unscaled) from a madrigal record
*
*  This method differs from cedarGet2dInt in that it only returns
*  a single unscaled Int16 from a particular row.
*
*  If nrow = 0,  or row > number of 2d rows, returns missingData.
*
*  If parcode not found, returns missingData
*/

Int16 cedarGet2dIntValue(Int16 *cedarp, int parcode, int row)



/*****************************************************************
*
* cedarCreateRecord  creates a new Cedar record
*
*   User is responsible for freeing dynamically allocated array
*   when finished with it.
*
*   If two many rows passed in to fit in Cedar format, then prints error
*   to stderr and returns NULL pointer
*/

Int16 *cedarCreateRecord(int lprol, int jpar, int mpar, int nrow,
                         int krec, int kinst, int kindat,
                         int year1, int month1, int day1,
                         int hour1, int minute1, int second1, int centisecond1,
                         int year2, int month2, int day2,
                         int hour2, int minute2, int second2, int centisecond2)



/*****************************************************************
*
* cedarCreateCatalogRecord  creates a new Cedar Catalog record
*
*   This method creates a catalog record with or without the actual text.
*   Users can also append text to this record by calling cedarAppendCatalogRecord
*
*   Inputs:
*
*       kinst - instrument code from instTab.txt
*       modexp - code describing the mode of the experiment
*       year1, month1, day1, hour1, minute1, second1, centisecond1 - starting time
```

```
*           of experiment
*       year2, month2, day2, hour2, minute2, second2, centisecond2 - starting time
*           of experiment
*       text - text to append.  See Cedar database format for suggested layout.
*              Must be multiple of 80 characters in length - no line feeds.  May
*              be empty, if user is planning to use cedarAppendCatalogRecord.
*
*   Returns - pointer to Int16 holding newly allocated catalog record. User is
*   responsible for freeing dynamically allocated array
*   when finished with it.
*/

Int16 *cedarCreateCatalogRecord(int kinst, int modexp,
                                int year1, int month1, int day1,
                                int hour1, int minute1, int second1, int centisecond1,
                                int year2, int month2, int day2,
                                int hour2, int minute2, int second2, int centisecond2,
                                char * text)



/**************************************************************************
*
* cedarAppendCatalogRecord  appends text to an existing Catalog Record
*
*   Users should first create a catalog record by calling cedarCreateCatalogRecord
*
*   Inputs:
*
*       Int16 *cedarp - pointer to existing catalog record
*       char * text - text to append.  See Cedar database format for suggested layout.
*                     Must be multiple of 80 characters in length - no line feeds.
*
*   Returns: 0 if success, -1 if failure
*
*/
int cedarAppendCatalogRecord(Int16 **cedarpp, char * text)



/**************************************************************************
*
* cedarCreateHeaderRecord  creates a new Cedar Header record
*
*   This method creates a header record with or without the actual text.
*   Users can also append text to this record by calling cedarAppendHeaderRecord
*
*   Inputs:
*
*       kinst - instrument code from instTab.txt
*       kindat - code describing the kind of data
*       year1, month1, day1, hour1, minute1, second1, centisecond1 - starting time
*           of experiment
*       year2, month2, day2, hour2, minute2, second2, centisecond2 - starting time
*           of experiment
*       jpar - number of single-valued parameters in accompanying data records
*       mpar - number of multiple-valued parameters in accompanying data records
*       text - text to append.  See Cedar database format for suggested layout.
*              Must be multiple of 80 characters in length - no line feeds.  May
*              be empty, if user is planning to use cedarAppendHeaderRecord.
*
```

```
*   Returns - pointer to Int16 holding newly allocated header record. User is
*   responsible for freeing dynamically allocated array
*   when finished with it.
*/

Int16 *cedarCreateHeaderRecord(int kinst, int kindat,
                               int year1, int month1, int day1,
                               int hour1, int minute1, int second1, int centisecond1,
                               int year2, int month2, int day2,
                               int hour2, int minute2, int second2, int centisecond2,
                               int jpar, int mpar,
                               char * text)



/*****************************************************************
*
* cedarAppendHeaderRecord  appends text to an existing Header Record
*
*   Users should first create a header record by calling cedarCreateHeaderRecord
*
*   Inputs:
*
*       Int16 *cedarp - pointer to existing header record
*       char * text - text to append.  See Cedar database format for suggested layout.
*                     Must be multiple of 80 characters in length - no line feeds.
*
*   Returns: 0 if success, -1 if failure
*
*/
int cedarAppendHeaderRecord(Int16 **cedarpp, char * text)



/*****************************************************************
*
* cedarSetKrec   sets Kind of record
*
*/

int cedarSetKrec (Int16 *cedarp, int krec)



/*****************************************************************
*
* cedarSetKinst   sets instrument code for these data
*
*/

int cedarSetKinst (Int16 *cedarp, int kinst)



/*****************************************************************
*
* cedarSetKindat   sets kind-of-data code
*
*/

int cedarSetKindat (Int16 *cedarp, int kindat)
```

```
/*****************************************************************
*
* cedarSetStartTime   sets start time of record
*
*/

int cedarSetStartTime (Int16 *cedarp, int year, int month, int day,
                       int hour, int minute, int second, int centisecond)



/*****************************************************************
*
* cedarSetEndTime   sets end time of record
*
*/

int cedarSetEndTime (Int16 *cedarp, int year, int month, int day,
                     int hour, int minute, int second, int centisecond)



/*****************************************************************
*
* cedarSet1dParm   sets a 1D parameter in a Cedar record
*
*   Inputs:
*       Int16 *cedarp - pointer to existing Cedar record
*       int parcode   - Cedar parmater code
*       double parm   - doubles containing value to set.Special values
*                       may be set by setting values to #defines
*                       missing, assumed, or knownbad
*       int index     - index of which 2d parameter to set
*
*   Returns 1 if failure, 0 if success.  If value out of Int16 range, will set
*   value to missing and return failure.
*/

int cedarSet1dParm(Int16 *cedarp, int parcode, double parm, int index)



/*****************************************************************
*
* cedarSetNorm1dParm   sets a 1D parameter in a Cedar record with the units
*                      of the standard parameter even if additional increment parameter
*
*   Inputs:
*       Int16 *cedarp - pointer to existing Cedar record
*       int parcode   - Cedar parmater code
*       double parm   - doubles containing value to set.Special values
*                       may be set by setting values to #defines
*                       missing, assumed, or knownbad
*       int index     - index of which 2d parameter to set
*
*   Returns 1 if failure, 0 if success.  If value out of Int16 range, will set
*   value to missing and return failure.
*/
```

```
int cedarSetNorm1dParm(Int16 *cedarp, int parcode, double parm, int index)



/*****************************************************************
*
* cedarSet2dParm   sets all values for a 2D parameter in a cedar record
*
*   Inputs:
*       Int16 *cedarp - pointer to existing Cedar record
*       int parcode   - Cedar parmater code
*       double *parmp - array of doubles containing values to set.  Length
*                       must be nrow.  Special values may be set by setting
*                       values to #defines missing, assumed, or knownbad
*       int index     - index of which 2d parameter to set
*
*   Returns 1 if failure, 0 if success. If any value out of Int16 range, will set
*   value to missing, but all valid values will still be set.  Returns 1 if any
*   out of range data found.
*
*/

int cedarSet2dParm(Int16 *cedarp, int parcode, double *parmp, int index)



/*****************************************************************
*
* cedarSetNorm2dParm   sets all values for a 2D parameter in a cedar record
*                      with the units as standard parameter even if
*                      additional increment parameter
*
*   Inputs:
*       Int16 *cedarp - pointer to existing Cedar record
*       int parcode   - Cedar parmater code
*       double *parmp - array of doubles containing values to set.  Length
*                       must be nrow.  Special values may be set by setting
*                       values to #defines missing, assumed, or knownbad
*       int index     - index of which 2d parameter to set
*
*   Returns 1 if failure, 0 if success. If any value out of Int16 range, will set
*   value to missing, but all valid values will still be set.  Returns 1 if any
*   out of range data found.
*
*/

int cedarSetNorm2dParm(Int16 *cedarp, int parcode, double *parmp, int index)



/*****************************************************************
*
* cedarSet1dInt   puts a 1D parameter (unscaled) into a madrigal record
*
*/

int cedarSet1dInt(Int16 *cedarp, int parcode, Int16 int1d, int index)
```

```
/*****************************************************************
*
* cedarSet2dInt   puts a 2D parameter (unscaled) into a madrigal record
*
*/

int cedarSet2dInt(Int16 *cedarp, int parcode, Int16 *int2dp, int index)



/*****************************************************************
*
* cedarPrintRecord   prints cedar record
*
*/

int cedarPrintRecord(Int16 *cedarp)



/*****************************************************************
*
* cedarGetInformation   gets Ascii Information from Catalog or Header record
*
*   inputs:  Int16 * cedarp (pointer to cedar record)
*
*   outputs:  char * (pointer to dynamically allocated string holding
*             ASCII text in catalog or header record, or empty string
*             if no text available.  Will return empty string if called
*             with a data record instead of a header or catalog record).
*             The string will have 81 characters for each line of
*             information - the first 80 characters will be the 80
*             characters in the file with unprintable characters converted
*             to spaces, and the 81st character a newline. After the last
*             line a null character is added to make a valid c string.
*
*  The user is resposible for freeing the returned string when finished
*  with it.
*
*/

char * cedarGetInformation(Int16 *cedarp)



/*****************************************************************
*
* cedarPrintProlog   prints cedar record prolog
*
*/

int cedarPrintProlog(Int16 *cedarp)



/*****************************************************************
*
* cedarReadParCodes   reads the following metadata tables:
*
*  1. parcods.tab  (parameter information)
*  2. instTab.txt   (instrument location)
```

```
*  3. madCatTab.txt  (parameter category information)
*
*   For the moment hard-coded to the column layout of parcods.tab
*      0-7     Code
*      10-48   Description   Note: DESC_LEN   = 40
*      50-60   Int16Desc     Note: DESC16_LEN = 12
*      62-68   ScaleFactor
*      70-77   Units         Note: UNIT_LEN   =  9
*      81-100  Mnemonic      Note: MNEM_LEN   = 21
*      105-112 Format
*      114-115 Width
*      118-120 CatId
*      122-122 hasDesc  - does this mnemonic have an html description?
*      124-124 hasErrDesc - does this error mnemonic have an html description?
*
*   Returns 0 if successful, non-zero and error set if not successful
*/

int cedarReadParCodes ()



/*****************************************************************
*
* cedarGetNumParCodes   Gets number of Cedar parameter codes in parcods.tab
*/

int cedarGetNumParCodes ()



/*****************************************************************
*
* cedarGetParCode   Gets Cedar parameter code from table given its position
*                   in file parcods.tab
*
*
*/

int
cedarGetParCode (int position)



/*****************************************************************
*
* cedarGetParCodeIndex   Gets index of Cedar parameter code in table, given its code
*
*   For a pure Madrigal parameter with code 0, will return missing.  Use
*   madGetParMnemIndex instead.  For a negative parcode, will return negitive
*   of index found.  If not found, returns missingData.
*
*   No longer requires that parcods.tab be in order.
*/

int cedarGetParCodeIndex (int parcode)



/*****************************************************************
*
```

```
*  madGetParMnemIndex   Gets index of Madrigal parameter code in table, given its mnemonic
*
*   Returns the index of the specified mnemonic.  Matching is case-insensitive, and
*   ignores whitespace. If not found and begins with "D", will next search with "D"
*   removed, and return the negitive of the index found.  If still not found,
*   returns missingData.
*
*/
int madGetParMnemIndex (char * mnem)



/*****************************************************************
*
*  isMadparmError   returns 1 if this is an error parm, 0 if standard,
*                   -1 if neither
*
*
*/
int isMadparmError(const char * mnem)



/*****************************************************************
*
*  getStdMnem   converts a str to standard mnemonic form
*
*   Inputs: const char * mnem    - the string containing the mnemonic to be converted
*           char * stdMnem - a string to copy the standard form of the
*                            mnemonic to.  Allocated by the user.  At most
*                            MNEM_LEN - 1 characters will be copied.  Std form
*                            strips all whitespace and is upper case.
*
*/

void getStdMnem (const char * mnem, char * stdMnem)



/*****************************************************************
*
* cedarGetParCodeType   Gets type of Cedar parameter code from table, given its code.
*
*   For a pure Madrigal parameter with code 0, will return first found.  Use
*   madGetParMnemType instead.  If not in parcods.tab but in a standard range,
*   as defined by madCatTab.txt will return Cedar values.  If not in parcodes
*   and not in any standard range, will return "Unknown Parameter Type"
*
*   User is responsible for freeing dynamically allocated string
*   when finished with it.
*/

char * cedarGetParCodeType (int parcode)



/*****************************************************************
*
* madGetParMnemType   Gets type of Madrigal parameter from table, given its mnemonic.
*
*   If mnemonic not in parcods.tab, will return "Unknown Parameter Type"
```

```
*
*   User is responsible for freeing dynamically allocated string
*   when finished with it.
*/

char * madGetParMnemType (char * mnem)



/*****************************************************************
*
* madGetCategoryIndex   Gets the index of a given Category name.
*
*   If category string not found, returns missingData
*   Matching is case and whitespace sensitive
*
*/

int madGetCategoryIndex (char * category)



/*****************************************************************
*
* cedarGetParDescription   Gets Cedar parameter code description from
* table
*
*   User is responsible for freeing dynamically allocated string
*   when finished with it.
*
*
*/

char * cedarGetParDescription (int parcode)



/*****************************************************************
*
* madGetParDescription   Gets Madrigal parameter code description from
* table, given mnemonic
*
*   User is responsible for freeing dynamically allocated string
*   when finished with it.
*
*/

char * madGetParDescription (char * mnem)



/*****************************************************************
*
* cedarGetParInt16Description   Gets Cedar parameter code Int16
*                               description from table
*
*   User is responsible for freeing dynamically allocated string
*   when finished with it.
*/

char * cedarGetParInt16Description (int parcode)
```

```
/*****************************************************************
*
* madGetParInt16Description   Gets Madrigal parameter code Int16
*                               description from table, given mnemonic
*
*/

char * madGetParInt16Description (char * mnem)



/*****************************************************************
*
* cedarGetParScaleFactor   Gets Cedar parameter scale factor from table
*
*/

double cedarGetParScaleFactor (int parcode)



/*****************************************************************
*
* madGetParScaleFactor   Gets Madrigal parameter scale factor from table,
*                        given mnemonic
*
*/

double madGetParScaleFactor (char * mnem)



/*****************************************************************
*
* cedarGetNormScaleFactor   Gets Cedar parameter scale factor, where additional
*                           increment parameters use the same units as main
*                           parameter.  Differs from cedarGetParScaleFactor, which
*                           returns scale factors for additional increment parameters
*                           that may have different units than the main parameter.
*
*/

double cedarGetNormScaleFactor (int parcode)



/*****************************************************************
*
* madGetNormScaleFactor   Gets Madrigal parameter scale factor, where additional
*                           increment parameters use the same units as main
*                           parameter.  Differs from cedarGetParScaleFactor, which
*                           returns scale factors for additional increment parameters
*                           that may have different units than the main parameter.
*
*/

double madGetNormScaleFactor (char * mnem)
```

```
/*****************************************************************
*
* cedarGetParUnits   Gets Cedar parameter code units from table
*
*   User is responsible for freeing dynamically allocated string
*   when finished with it.
*/

char * cedarGetParUnits (int parcode)



/*****************************************************************
*
* madGetParUnits   Gets Madrigal parameter code units from table,
*                  given mnemonic
*
*   User is responsible for freeing dynamically allocated string
*   when finished with it.
*/

char * madGetParUnits (char * mnem)



/*****************************************************************
*
* cedarGetParMnemonic   Gets Cedar parameter code mnemonic from table
*
*   User is responsible for freeing dynamically allocated string
*   when finished with it.  If parcode 0 passed in, first Madrigal
*   parameter found will be returned.
*
*   If unknown parcode passed in, mnemonic is atoi(parcode)
*/

char * cedarGetParMnemonic (int parcode)



/*****************************************************************
*
* cedarGetParCodeFromMnemonic   Gets Cedar parameter code given mnemonic
*
*   If mnemonic is integer in form of string, returns that integer
*   If not found, returns missingData
*
*/

int cedarGetParCodeFromMnemonic (char * mnem)



/*****************************************************************
*
* cedarGetParFormat   Gets Cedar parameter code format from table.
*
*   If not found, returns NULL
*/
```

```
char * cedarGetParFormat (int parcode)



/*****************************************************************
*
* madGetParFormat   Gets Madrigal parameter format from table (given mnemonic)
*
*   If not found, returns NULL
*/

char * madGetParFormat (char * mnem)



/*****************************************************************
*
* cedarGetParWidth   Gets Cedar parameter field width from table
*
*    If unknown, returns default value of 11
*
*/

int cedarGetParWidth (int parcode)



/*****************************************************************
*
* madGetParWidth   Gets Madrigal parameter field width from table,
*                  given mnemonic
*
*    If unknown, returns default value of 11
*/

int madGetParWidth (char * mnem)



/*****************************************************************
*
* cedarHasHtmlDesc   Returns 1 if parameter has entry in Html description
*                    page, 0 if not.  Works also for error codes (<0)
*
*    If unknown, returns default value of 0
*
*/

int cedarHasHtmlDesc(int parcode)



/*****************************************************************
*
* madHasHtmlDesc   Returns 1 if mnemonic has entry in Html description
*                    page, 0 if not.  Works also for error mnemonics.
*
*    If unknown, returns default value of 0
*/
```

```
int madHasHtmlDesc (char * mnem)


/*****************************************************************
*
* cedarCheckRecord   checks cedar record for consistency
*
*/

int cedarCheckRecord (Int16 *cedarp)


/*****************************************************************
*
* cedarHexPrintRecord   prints hex version of record
*
*/

int cedarHexPrintRecord (Int16 *cedarp)


/*****************************************************************
*
* cedarDecimalPrintRecord   prints hex version of record
*
*/

int cedarDecimalPrintRecord (Int16 *cedarp)


/*****************************************************************
*
* cedarSetError   sets cedar error
*
*/

int cedarSetError (const char *error)


/*****************************************************************
*
* cedarGetError   gets last cedar error
*
*/

char * cedarGetError ()


/*****************************************************************
*
* cedarTabInt   linear interpolation routine
*
* tabint interpolates linearly to calculate y(x) from a table
* containing nt independent variable values xt and dependent
* variable values yt. the xt are assumed to be in non-decreasing
```

```
* order.
*
*/

double cedarTabInt (int nt, double *xt, double *yt, double x, double badval)



/*****************************************************************
*
* cedarUpdateParmsList   Updates list of parameters and their minimum
*                        and maximum values
*
*   The first eleven parameters are effectively derived:
*   [0] 10: year, [1] 11: month,  [2] 12: day
*   [3] 13: hour, [4] 14: minute, [5] 15: second,
*   [6] 16: centisecond [7] 34: uth, [8] 160: gdlat,
*   [9] 170: glon, [10] 110: gdalt
*
*   All parameters actually in the file will be listed starting with
*   the 12th.  If any of the above parameters are actually in the file
*   itself, they will appear again.
*
*   Does not include data from 2D rows if all error parameters are
*   missing of knownbad
*
*   Also updates earliestStartTime, latestEndTime, and lists of
*   all kinsts and kindats found in file.
*
*   If header or catalog record, returns 0 immediately without making
*   any changes.
*
*/

int cedarUpdateParmsList(Int16 *cedarp, int *numParmsp,
                         int *parmsListpp[], int *parmLocpp[],
                         double *parmMinpp[], double *parmMaxpp[], int *parmMissing[],
                         int *startJday0,
                         double * earliestStartTime, double * latestEndTime,
                         int * numKinst, int * kinstArr,
                         int * numKindat, int * kindatArr)



/*****************************************************************
*
*  cedarGetStationPos  Gets instrument coordinates for a given kinst
*
*      Uses data from metadata/instTab.txt
*
*/

void cedarGetStationPos(int kinst, double * lat, double * lon, double * alt)



/*****************************************************************
*
*  cedarGetStationName  Gets instrument name for a given kinst
*
*      Uses data from metadata/instTab.txt
```

```
*
*/

char * cedarGetStationName(int kinst)



/*****************************************************************
*
* searchFilesByDate    searches the metadata for all files between
*                      starttime and endtime.
*
*   arguments:
*       double starttime: seconds since 1/1/1950.
*       double endtime: seconds since 1/1/1950.
*       int * numFilesFound: pointer to int, set to number of files found
*       char ** fileList: pointer to char pointer to be allocated and
*                         populated with a comma-delimited list of full
*                         paths to files found
*       double ** fileStarttime: pointer to double array to be allocated and
*                                populated with start times of each file found
*                                (number of seconds since 1/1/1950)
*       double ** fileEndtime: pointer to double array to be allocated and
*                              populated with end times of each file found
*                              (number of seconds since 1/1/1950)
*
*       To be found, the file must start after starttime and end before endtime.
*       File must also be a default file.
*
*       User must free fileList, fileStarttime, fileEndtime if numFilesFound > 0
*
*   returns: 0 if success, non-zero and error set if not successful
*
*/
int searchFilesByDate(double starttime,
                      double endtime,
                      int * numFilesFound,
                      char ** fileList,
                      double ** fileStarttime,
                      double ** fileEndtime)



/*****************************************************************
*
* loadExpFileTable    Loads data from expTab.txt and fileTab.txt into
*                     global data. Private method - do not call directly.
*
*   arguments: None
*
*   returns: 0 if success, non-zero and error set if not successful
*
*   Affects: loads global data that deals with expTab and fileTab
*
*/
int loadExpFileTable()



/*****************************************************************
*
```

```
* goodDataExists    Returns 1 if record contains valid data at 2d parameter
*                   index index2D.  Valid data is when the absolute value
*                   of any error parameter is not missing or knownbad.
*                   If no error parameters, always returns 1.
*                   0 otherwise.
*
*   arguments:
*       record pointer to Madrigal record
*       index into 2D parameter values
*
*   returns:
*       1 if 2d index contains valid data,
*       0 if not
*
*/
int goodDataExists(Int16 * recordp, int index2D)



/*********************************************************************
*
* sprod calculates the scalar product of two vectors a and b,
* sprod = a .dot. b.
*/
double
sprod(double *a, double *b)



/*********************************************************************
*
* vadd calculates the sum of two vectors a and b, c = a + b.
*/
int
vadd(double *a, double *b, double *c)



/*********************************************************************
*
* vsub calculates the difference of two vectors a and b, c = a - b.
*/
int
vsub(double *a, double *b, double *c)



/*********************************************************************
*
* csconv converts between cartesian coordinates x,y,z and spherical
* coordinates r,theta,phi. if imode=1, (x,y,z) -> (r,theta,phi).
* if imode=2, (r,theta,phi) -> (x,y,z). theta and phi are in
* degrees.
*/
int
csconv(double *xp, double *yp, double *zp,
       double *rp, double *thetap, double *phip,
       int imode)
```

```
/*****************************************************************
*
* vctcnv converts between the cartesian and spherical coordinate
* representations of a vector field f. (fx,fy,fz) are the
* components of the field at (x,y,z). (fr,ft,fp) are the
* components of the field at (r,theta,phi) in the directions of
* increasing r, increasing theta and increasing phi. if imode=1,
* (fx,fy,fz,x,y,z) -> (fr,ft,fp,r,theta,phi). if imode=2,
* (fr,ft,fp,r,theta,phi) -> (fx,fy,fz,x,y,z). theta and phi are
* in degrees.
*/
int
vctcnv(double *fxp, double *fyp, double *fzp,
       double *xp, double *yp, double *zp,
       double *frp, double *ftp, double *fpp,
       double *rp, double *thetap, double *phip,
       int imode)



/*****************************************************************
*
* point calculates the position of a point defined by the radar
* line-of sight vector to that point.
*
* input parameters
*    sr    - distance of station from center of earth (km)
*    slat  - geocentric latitude of station (deg)
*    slon  - longitude of station (deg)
*    az    - radar azimuth (deg)
*    el    - radar elevation (deg)
*    range - radar range (km)
*
* output parameters
*    pr    - distance from center of earth of observation point (km)
*   glat  - observation point geocentric latitude (deg)
*    glon  - observation point longitude (deg)
*/
int
point(double *srp, double *slatp, double *slonp,
      double *azp, double *elp, double *rangep,
      double *prp, double *glatp, double *glonp)



/*****************************************************************
*
* look calculates the azimuth, elevation and range from a radar
* of a specified point.
*
* input parameters
*    sr    - distance of station from center of earth (km)
*    slat  - geocentric latitude of station (deg)
*    slon  - longitude of station (deg)
*    pr    - distance from center of earth of observation point (km)
*    glat  - observation point geocentric latitude (deg)
*    glon  - observation point longitude (deg)
*
* output parameters
*    az    - radar azimuth (deg)
*    el    - radar elevation (deg)
```

```
*    range - radar range (km)
*/
int
look(double *srp, double *slatp, double *slonp,
     double *prp, double *glatp, double *glonp,
     double *azp, double *elp, double *rangep)



/*********************************************************************
*
* convrt converts between geodetic and geocentric coordinates. the
* reference geoid is that adopted by the iau in 1964. a=6378.16,
* b=6356.7746, f=1/298.25. the equations for conversion from
* geocentric to geodetic are from astron. j., vol 66, 1961, p. 15.
*      i=1   geodetic to geocentric
*      i=2   geocentric to geodetic
*    gdlat   geodetic latitude (degrees)
*    gdalt   altitude above geoid (km)
*    gclat   geocentric latitude (degrees)
*      rkm   geocentric radial distance (km)
*/
int
convrt(int i, double *gdlatp, double *gdaltp,
       double *gclatp, double *rkmp)



/*********************************************************************
*
* rpcart computes the components (rfx,rfy,rfz) relative to an earth
* centered cartesian coordinate system of the radar line of sight
* vector from a radar with coordinates sr (distance from center
* of earth), slat (geocentric latitude) and slon (longitude). the
* observation point is specified by az (azimuth), el (elevation) and
* range (range). the cartesian coordinates of the observation
* point are returned in (pfx,pfy,pfz).
*    input - sr,slat,slon,az,el,range
*    output - rfx,rfy,rfz,pfx,pfy,pfz
*/
int
rpcart (double *srp,  double *slatp, double *slonp,
        double *azp,  double *elp,   double *rangep,
        double *rfxp, double *rfyp,  double *rfzp,
        double *pfxp, double *pfyp,  double *pfzp)



/*********************************************************************
*
* gdv converts a vector field f at geodetic latitude gdlat and
* geocentric latitude gclat from a geocentric based representation
* to a geodetic based representation. the geocentric components
* are fr (radial outward), ft (increasing geocentric colatitude,
* e.g. southward) and fp (increasing east longitude). the
* geodetic components are fx (northward, parallel to surface of
* earth), fy (eastward, parallel to surface of earth) and fz
* (downward, perpendicular to surface of earth). fr,ft,fp thus
* correspond to spherical coordinates r,theta,phi, with their
* origin at the center of the earth. x,y,z are the coordinates
```

```
* customarily used to describe the three components of the
* geomagnetic field. fp and fy are the same.
*/
int
gdv(double *gdlatp, double *gclatp,
    double *frp, double *ftp, double *fpp,
    double *fxp, double *fyp, double *fzp)



/*****************************************************************
*
* los2geodetic calculates the position of a point defined by an instrument
* line-of sight vector to that point. This is a convenience routine in
* which the instrument location is specified by its CEDAR instrument code
* and which returns the geodetic coordinates of the point.
*
*
* input parameters
*    kinst - instrument code in metadata
*    az    - radar azimuth (deg)
*    el    - radar elevation (deg)
*    range - radar range (km)
*
* output parameters
*    gdlat - observation point geodetic latitude (deg)
*    glon  - observation point longitude (deg)
*    gdalt - altitude above geoid (km)
*/

int los2geodetic(int kinst, double az, double el, double range,
             double *gdlatp, double *glonp, double *gdaltp)



/*****************************************************************
*
* solarzen_az calculates the solar zenith and az angles for a given time, gdlat,
*          and glon.
*
*
* input parameters
*    double ut    - Universal time in seconds since 1950
*    double gdlat - geodetic latitude in degrees
*    double glon  - geodetic longitude in degrees
*
* output parameters
*    szen - solar zenith angle (deg, 0=directly overhead)
*    saz  - solar azimuth angle (deg, N=0, E=90)
*
*
*    Solar zenith angle is calculated at 0 alt, although this changes
*    very little with altitute.  No atmospheric correction is applied.
*    This method uses solpos.c, written by National Renewable Energy
*    Laboratory.
*
*    This method is a modified version of  stest.c found at
*    http://rredc.nrel.gov/solar/codes_algs/solpos/
*/
void solarzen_az(double ut, double gdlat, double glon, double * szen, double * saz)
```

```
/**********************************************************************
*
* solardist calculates the distance in km from the center of the earth
*          to the center of the sun at time ut.
*
*
* input parameters
*    double ut    - Universal time in seconds since 1950
*
* returns double - distance in km from the center of the earth
*          to the center of the sun at time ut
*
*    This method is taken from "Practical Astronomy with Your
*    Calculator" 2nd edition, Peter Duffett-Smith, p. 80-87.
*/
double solardist(double ut)



/**********************************************************************
*
* shadowheight calculates the distance directly above any gdlat and glon
*              for a given UT in km at which the earth's shadow terminates.
*              Will be 0.0 on dayside of earth.
*
*
* input parameters
*    double ut    - Universal time in seconds since 1950
*    double gdlat - geodetic latitude in degrees
*    double glon  - geodetic longitude in degrees
*
* returns double - distance directly above any gdlat and glon
*              for a given UT in km at which the earth's shadow terminates.
*              Will be 0.0 on dayside of earth.
*
*    This method uses the results of solarzen and solardist to create a simple
*    cone on a sphere model of the earth's shadow.  Shadow height is defined as
*    the lowest elevation at which any part of the sun can be seen.  No atmospheric
*    bending of light is included. The radius of the earth is calculated at the
*    tan point of the sun's rays.
*
*    Algorithm:
*
*    Solar Zenith Angle = Z
*    Solar Azimuth      =Az (0 = North, 90 = East)
*
*    Get latitude of tangent point of sun's rays:
*
*      = gdlat + cos(Az)*(Z-90.0)
*
*    Get earthRadius at that point using convrt at gdlat = 0 (sea level)
*
*    ConeHalfAngle = C = atan((sunRadius - earthRadius)/soldist)
*
*
*                                (cos Z tan C + 1 - sin Z)
*    Shadowheight = earthRadius -------------------------
*                                   (sin Z - cos Z tan C)
*
```

```
*    Daytime (Shadowheight = 0) if numerator negitive or if
*    Z <= 91.0.
*/
double shadowheight(double ut, double gdlat, double glon)



/*****************************************************************
*
* sunrise_set calculates the time UT  of ionospheric sunrise
*             and sunset.
*
*
* input parameters
*    double ut    - Universal time in seconds since 1950
*    double gdlat - geodetic latitude in degrees
*    double glon  - longitude in degrees
*    double gdalt - geodetic altitude in km
*    double * sunrise - pointer to double allocated by user to be
*                       set to sunrise time UT
*    double * sunset - pointer to double allocated by user to be
*                       set to sunset time UT
*
* returns void
*
*    If either sunrise or sunset not found, set to missing.
*    All times in seconds since 1/1/1950
*
*  Algorithm:
*
*    Depends on shadowheight calculation, so limitations discussed
*    there apply (atmospheric bending of light ignored).
*
*    If glon <0:
*      solar midnight = UT - glon*24/360
*      solar noon     = UT + 12 - glon*24/360
*
*    If sun is not set at solar midnight (defined by shadowheight > gdalt),
*    or sun not up at solar noon, check if any difference between 0 and 24 UT.
*    If so, find only one of sunrise and sunset as described below, and set
*    the other to missing.  If not, return missing for both, because that point
*    is either in the sun or in the shadow all day.  The user must determine
*    which by comparing shadowheight and gdalt.  Otherwise, seek sunrise
*    between solar midnight and solar noon, slice remaining time in half each
*    guess.  Stop when time step less than 1 minute.
*
*            If sun is up at 0 UT that day:  Seek sunset between 0.0 and
*    solar midnight as above.
*            Else: Seek sunset between solar noon and 24.0 as above.
*
*    Else if glon > 0:
*        solar midnight = UT + 24 - glon*24/360
*        solar noon     = UT + 12 - glon*24/360
*
*    If sun is not set at solar midnight (defined by shadowheight > gdalt),
*    or sun not up at solar noon, check if any difference between 0 and 24 UT.
*    If so, find only one of sunrise and sunset as described below, and set
*    the other to missing.  If not, return missing for both, because that point
*    is either in the sun or in the shadow all day.  The user must determine
*    which by comparing shadowheight and gdalt.  Otherwise, seek sunset
*    between solar noon and solar midnight, slice remaining time in half each
```

```
*    guess.  Stop when time step less than 1 minute.
*
*            If sun is up at 0 UT that day:  Seek sunrise between solar
*    midnight and 24.0 as above.
*
*            Else: Seek sunrise between 0.0 and solar noon as above.
*
*   Note: day is divided 11 times to ensure greater than one minute resolution.
*/
void sunrise_set(double ut,
                 double gdlat,
                 double glon,
                 double gdalt,
                 double * sunrise,
                 double * sunset)



/*****************************************************************
*  jday  - returns Julian day number given day, month, and year
*
*    Julian day 0 is Nov. 24, -4713
*
*    Returns -1 if illegal year, month, day passed in
*/
int jday(int day, int month, int year)



/*****************************************************************
*  jdater - sets day, month and year given jdayno
*
*    Inverse of jday method
*/
    int jdater (int jdayno, int *day, int *month, int *year)



/*****************************************************************
*
*  idmyck - returns 0 if valid day, month, and year
*
*/
int idmyck(int day, int month, int year)



/*****************************************************************
*
* dmadptr - returns number of seconds since 1/1/1950 for iyr, imd, ihm, ics
*           as double
*
*    Can retain fractions of a second
*
*    Inputs: iyr - year
*            imd - month/day as integer mmdd
*            ihm - hour/min as integer hhmm
*            ics - centiseconds since last minute
*
*/
double dmadptr(int iyr, int imd, int ihm, int ics)
```

```
/*****************************************************************
*
* getKey - returns number of seconds since 1/1/1950 for year,
*          month, day, hour, minute, second
*
*/
double getKey(int year, int month, int day,
           int hour, int minute, int second)



/*****************************************************************
*
* dinvmadptr - sets iyr, imd, ihm, ics given dmadptr (number of seconds
*              as a double since 1/1/1950)
*
*    inverse of dmadptr.
*
*    Returns 0 if success, 1 if failure (out of range time)
*    Uses time.h, and constant to shift from 1/1/1970 to 1/1/1950
*/
int dinvmadptr(double dmadptr, int * iyr, int * imd, int * ihm, int * ics)



/*****************************************************************
*
* madGetDayno - gets day number (1-366) given year, month, and day
*
*
*    Returns -1 if illegal year, month, day passed in
*/
int madGetDayno(int year, int month, int day)
```

The maddata module

createMadparmList	copyMadparmList	destroyMadparmList	appendMadparm	hasParm
isErrorParm	getIndex	getMinParm	getMaxParm	analyzeFileParms
getDerivedParms	createMadfilterList	destroyMadfilterList	appendMadfilter	copyMadfilterList
getMadfilterListFromStr	createMaddata	createNonfileMaddata	destroyMaddata	appendMadrecPar
appendMadcycle	createMadcycle	destroyMadcycle	createMadrecord	destroyMadrecord
simpleMadrecordPrint	simpleMadfilterPrint	simpleMaddataPrint	classicIsprint	printIsprintHeader
getIsprintHeader	printIsprintLabel	getIsprintLabel	classicMadrecordPrint	getClassicMadrec
lookerMadrecordPrint	populate1DDataFromStr	populate2DDataFromStr		

```
/*****************************************************************
*
* createMadparmList   initializes a new MadparmList
*
*   arguments: None
```

```
*
*   returns - pointer to newly created MadparmList.  Use
*             destroyMadparmList when done
*/
MadparmList * createMadparmList()



/*****************************************************************
*
* copyMadparmList   copies an existing MadparmList
*
*   arguments: Pointer to existing MadparmList
*
*   returns - pointer to newly copied MadparmList, newly allocated
*             on the heap.  Use destroyMadparmList when done
*             Returns NULL if NULL passed in
*/
MadparmList * copyMadparmList(MadparmList * madparmList)



/*****************************************************************
*
* destroyMadparmList   releases an existing MadparmList
*
*   arguments: pointer to existing MadparmList
*
*   returns - void
*/
void destroyMadparmList(MadparmList * madParmList)



/*****************************************************************
*
* appendMadparm   adds a new parameter to the MadparmList
*
*   arguments: MadparmList * madparmList - pointer to existing MadparmList
*              const char * mnem - string containing name
*
*      mnem is copied into newly allocated memory, so user is free to
*      release mnem after this method.  mnem converted to standard form.
*
*   returns - 0 if success, -1 if failure (if mnem too long or unknown)
*/
int appendMadparm(MadparmList * madparmList, const char * mnem)



/*****************************************************************
*
*  hasParm   returns 1 if madparmList has parameter, 0 otherwise
*
*   arguments: MadparmList * madparmList - pointer to existing MadparmList
*              const char * mnem - string containing name
*
*      comparision is done after coverting mnem to standard form
*
*   returns - 0 if success, -1 if failure (if mnem too long)
*/
```

```
int hasParm(MadparmList * madparmList, const char * mnem)


/*****************************************************************
*
*  isErrorParm   returns 1 parameter at index is error, 0 if standard
*
*   arguments: MadparmList * madparmList - pointer to existing MadparmList
*              int index - index into madparmList
*
*   returns - 1 parameter at index is error, 0 if standard, -1 if
*             index out of bounds
*/
int isErrorParm(MadparmList * madparmList, int index)


/*****************************************************************
*
*  getIndex   returns index of parameter if found, -1 otherwise
*
*   arguments: MadparmList * madparmList - pointer to existing MadparmList
*              const char * mnem - string containing name
*
*      comparision is done after coverting mnem to standard form
*
*   returns - index of parameter if found, -1 otherwise
*/
int getIndex(MadparmList * madparmList, const char * mnem)


/*****************************************************************
*
*  getMinParm   returns minimum value of mnem, or missing if unknown
*
*   arguments: MadparmList * madparmList - pointer to existing MadparmList
*              char * mnem - string containing name of parameter
*
*   returns - minimum value of mnem, or missing if unknown or not in list
*/
double getMinParm(MadparmList * madparmList, char * mnem)


/*****************************************************************
*
*  getMaxParm   returns maximum value of mnem, or missing if unknown
*
*   arguments: MadparmList * madparmList - pointer to existing MadparmList
*              char * mnem - string containing name of parameter
*
*   returns - maximum value of mnem, or missing if unknown or not in list
*/
double getMaxParm(MadparmList * madparmList, char * mnem)


/*****************************************************************
*
```

```
* analyzeFileParms   get 4 lists of parameters from file:
*                     1) all 1D measured parameters
*                     1) all 2D measured parameters
*                     1) all 1D derivable parameters
*                     1) all 2D derivable parameters
*
*   arguments: char * filename - full path to file
*              MadparmList * list1DMeasParms - pointer to MadparmList to be
*                  populated with all 1D measured parameters found in file
*              MadparmList * list2DMeasParms - pointer to MadparmList to be
*                  populated with all 2D measured parameters found in file
*              MadparmList * list1DDervParms - pointer to MadparmList to be
*                  populated with all 1D parameters that could be derived
*              MadparmList * list2DDervParms - pointer to MadparmList to be
*                  populated with all 2D parameters that could be derived
*              FILE * errFile   - errFile to write an error messages to
*
*   returns - 0 if success, -1 otherwise
*
*   affects - populates the four input lists.  All four pointers should point
*   to NULL when passed in.  When done with these four lists, user should call
*   destroyMadparmList for each to free memory.
*
*   Note: Since a file may contain more than one type of record, these lists contain
*   parameters from any record that fits into each list.  For example, a certain 1D
*   parameter is measured in one type of record, but can be derived from another type
*   where its not measured, that parameter would appear in both list1DMeasParms and
*   list1DDervParms.
*
*   See also method getDerivableParms, which accepts two lists of measured 1D and
*   measured 2D parameters, and returns two lists of derivable 1D and
*   derivable 2D parameters.  Since this other method does not analyze a file, it does not
*   have the ambiguities of analyzeFileParms discussed above.
*/
int analyzeFileParms(char * filename,
                     MadparmList ** list1DMeasParms,
                     MadparmList ** list2DMeasParms,
                     MadparmList ** list1DDervParms,
                     MadparmList ** list2DDervParms,
                     FILE * errFile)



/*****************************************************************
*
* getDerivedParms   gets a list of derivable parameters given a list of
*                   measured parameters
*
*   arguments: MadparmList * listMeasParms - pointer to MadparmList
*                  containing measured parameters
*
*   returns - MadparmList * listDervParms - pointer to MadparmList
*                  containing all parameters that could be derived.
*                  User is responsible for calling destroyMadparmList
*                  when done with this list
*
*/
MadparmList * getDerivedParms(MadparmList * listMeasParms)
```

```
/*****************************************************************
*
* createMadfilterList   initializes a new MadfilterList
*
*   arguments: None
*
*   returns - pointer to newly created MadfilterList.  Use
*             destroyMadfilterList when done
*/
MadfilterList * createMadfilterList()



/*****************************************************************
*
* destroyMadfilterList   releases an existing MadfilterList
*
*   arguments: pointer to existing MadfilterList
*
*   returns - void
*/
void destroyMadfilterList(MadfilterList * madFiltList)



/*****************************************************************
*
* appendMadfilter   adds a new Madfilter to the MadfilterList
*
*   arguments: MadfilterList * madfilt_list - pointer to existing MadfilterList
*              Filter_type filtType - enum used to identify filter types
*              int numRange - number of ranges included - must be greater than 0
*              double * lower - array of lower limits of range - if "missing", no
*                               lower limit for that range
*              double * upper - array of upper limits of range - if "missing", no
*                               upper limit for that range
*              char * - madParm1 - Mnemonic of first parameter - cannot be 0 length
*              char * - madParm2 - Mnemonic of second parameter - can be 0 length if SINGLE_FIL
*
*      lower, upper, madParm1 and madParm2 are copied into newly allocated memory, so user is f
*      release them after this method.  madParm1 and madParm2 converted to standard mnemonic fo
*
*   returns - 0 if success, -1 if failure (if either mnemonic too long, if madfilt_list NULL,
*             or numRange <1)
*/
int appendMadfilter(MadfilterList * madFiltList,
                    Filter_type filtType,
                    int numRange,
                    double * lower,
                    double * upper,
                    char * madParm1,
                    char * madParm2)



/*****************************************************************
*
* copyMadfilterList   copies an existing MadfilterList
*
*   arguments: Pointer to existing MadfilterList
*
```

```
*   returns - pointer to newly copied MadfilterList, newly allocated
*             on the heap.  Use destroyMadfilterList when done
*             Returns NULL if NULL passed in
*/
MadfilterList * copyMadfilterList(MadfilterList * madfilterList)



/*****************************************************************
*
* getMadfilterListFromStr   creates a MadfilterList based on an isprint-like command string
*
*   arguments: str - an isprint-like command string
*
*      lower, upper, madParm1 and madParm2 are copied into newly allocated memory, so user is f
*      release them after this method.  madParm1 and madParm2 converted to standard mnemonic fo
*
*      The filter string is the same string that is used in the new isprint
*      command line.  Filters are separated by spaces.  The allowed filters
*      are:
*
*         date1=mm/dd/yyyy  (starting date to be examined. If time1 not given, defaults to 0 UT
*            Example: date1=01/20/1998
*
*         time1=hh:mm:ss (starting UT time to be examined. If date1 given, is applied to date1.
*                         If not, applies on the first day of the experiment.)
*            Example: time1=13:30:00
*
*         date2=mm/dd/yyyy (ending date to be examined.  If time2 not given, defaults to 0 UT.)
*            Example: date2=01/21/1998
*
*         time2=hh:mm:ss (ending UT time to be examined - If date2 not given, ignored.)
*            Example: time2=15:45:00
*
*         In the follow arguments ranges are used.  If any range value is not given, it may be
*         indicate no lower or upper limit (but the comma is always required). Ranges are inclu
*         of the end points:
*
*         z=lower alt limit1, upper alt limit1 [or lower alt limit2 , upper alt limit2 ...] (km
*            Example 1: z=100,500  (This would limit the geodetic altitude to 100 to 500 km.)
*            Example 2: z=100,200or300,400  (This would limit the geodetic altitude to 100 to 2
*                                             or 300 to 400 km.)
*            Example 3: z=,200or300,400   (Since the lower limit of the first range is missing,
*                                          would limit the geodetic altitude to anything below
*                                          or from 300 to 400 km.)
*
*         az=lower az limit1, upper az limit1 [or lower az limit2 , upper az limit2 ...] (from
*            Example 1: az=100,120  (This would limit the azimuth to 100 to 120 degrees.)
*            Example 2: z=-180,-90or90,180  (This would limit the azimuth to between -180 and -
*                                            to between 90 and 180 degrees.  Note this allows a
*                                            through 180 degrees.)
*
*         el=lower el limit1, upper el limit1 [or lower el limit2 , upper el limit2 ...] (from
*            Example 1: z=0,45  (This would limit the elevation from 0 to 45 degrees.)
*
*         plen=lower pl limit1, upper pl limit1 [or lower pl limit2 , upper pl limit2 ...] (pul
*            Example 1: z=,5e-4  (This would limit the pulse length to 5e-4 seconds or less.)
*
*
*         Free form filters using any mnemonic, or two mnemonics added, subtracted, multiplied,
*         Any number of filters may be added:
```

```
*
*         filter=[mnemonic] or [mnemonic1,[+*-/]mnemonic2] , lower limit1 , upper limit1 [or lo
*            Example 1: filter=ti,500,1000or2000,3000   (Limits the data to points where Ti is
*                                                       or between 2000 and 3000 degrees.  Note
*                                                       those of the Cedar standard.)
*            Example 2: filter=gdalt,-,sdwht,0,    (This filter implies "gdalt - sdwht" must be
*                                                   sdwht is shadow height - the distance above
*                                                   where the sun is first visible - this filte
*                                                   in direct sunlight will be displayed.)
*            Example 3: filter=ti,/,Dti,100,   (Limits the data to points where the ratio Ti/dT
*
*         So an full FLTSTR argument might be:
*
*            "date1=01/20/1998 time1=13:30:00 z=,200or300,400 filter=gdalt,-,sdwht,0, filter=ti
*
*   returns - MadfilterList if success, NULL if failure
*/
MadfilterList * getMadfilterListFromStr(char * str)



/*****************************************************************
*
* createMaddata   creates a new Maddata
*
*   arguments:
*
*     char * filename - full path to the file which was basis of data
*     char * infoStr  - Information string (may be used in outputing formatted data)
*     MadparmList *   madparmList - list of Madrigal parameters desired
*     MadfilterList * madFiltList - list of Madfilters to apply
*     FILE * errFile   - errFile to write an error messages to
*
*
*   returns - pointer to newly created Maddata.  Use
*             destroyMaddata when done
*
*   Allocates memory to store all data, so all input may be released or changed
*   after this method is called.  Maddata is the main data structure, and is meant
*   to be the main way to expose Madrigal data from a single cedar file that
*   applies filtering and calculates derived data.
*
*   Returns NULL if failure.
*/
Maddata * createMaddata(char * filename,
                        char * infoStr,
                        MadparmList * requestParmList,
                        MadfilterList * madfilterList,
                        FILE * errFile)



/*****************************************************************
*
* createNonfileMaddata   creates a new Maddata using user-supplied data
*                         rather than data from a file
*
*   arguments:
*
*     MadparmList * madparmList - list of Madrigal parameters desired
*     double ut1                - start time of integration period
```

```
*     double ut2                - end time of integration period
*     int kinst                 - kinst id - needed since its in the prolog
*     MadparmList * oneDParms,  - list of 1D parameters for which you plan
*                                 to provide data - may be 0 length
*     MadparmList * twoDParms,  - list of 2D parameters for which you plan
*                                  to provide data - may be 0 length
*     int num2Drows             - number of 2D rows - may be zero
*     double * oneDdata         - array of 1D data in order of oneDParms
*     double ** twoDdata        - array of num2Drows double * to 2D data
*                                 Each double * points to array of doubles of
*                                 length = length of twoDParms
*     FILE * errFile   - errFile to write an error messages to
*
*
*   returns - pointer to newly created Maddata.  Use
*             destroyMaddata when done.  Will contain only one Madrecord.
*
*   Allocates memory to store all data, so all input may be released or changed
*   after this method is called.  Use this method to calculate Maddata when
*   you want to directly provide measured data, rather than get it from a file.
*/
Maddata * createNonfileMaddata(MadparmList * requestedParms,
                               double ut1,
                               double ut2,
                               int kinst,
                               MadparmList * oneDParms,
                               MadparmList * twoDParms,
                               int num2Drows,
                               double * oneDdata,
                               double ** twoDdata,
                               FILE * errFile)



/*****************************************************************
*
* destroyMaddata   releases an existing Maddata
*
*   arguments: pointer to existing Maddata
*
*   returns - void
*/
void destroyMaddata(Maddata * maddata)



/*****************************************************************
*
* appendMadrecParmType   appends a new MadrecParmType onto maddata
*
*   arguments:
*
*     Maddata * maddata - pointer to Maddata to append new cycle to
*     MadparmList * parm1DList - the list of 1D parameters in that type
*     MadparmList * parm1DList - the list of 2D parameters in that type
*
*
*   returns - index of new type.  Starts at 0.  If failure,
*             returns -1
*
*   Allocates memory to store all data, so all input may be released or changed
```

```
*   after this method is called.
*/
int appendMadrecParmType(Maddata * maddata,
                         MadparmList * parm1DList,
                         MadparmList * parm2DList)



/*****************************************************************
*
* appendMadcycle   appends a new Madcycle onto maddata
*
*   arguments:
*
*     Maddata * maddata - pointer to Maddata to append new cycle to
*     int   cycleId     - cycle id (identifies cycle type)
*     char *   cycleDesc - Additional cycle description (may be empty string)
*
*
*   returns - cycle index of new cycle.  Starts at 0.  If failure,
*             returns -1
*
*   Allocates memory to store all data, so all input may be released or changed
*   after this method is called.
*/
int appendMadcycle(Maddata * maddata,
                   int cycleId,
                   char * cycleDesc)



/*****************************************************************
*
* createMadcycle   creates a new Madcycle
*
*   arguments:
*
*     int   cyclenum  - cycle number
*     int   cycleId   - cycle id (identifies cycle type)
*     char *   cycleDesc - Additional cycle description (may be empty string)
*
*
*   returns - pointer to newly created Madcycle.  Use
*             destroyMadcycle when done
*
*   Allocates memory to store all data, so all input may be released or changed
*   after this method is called.  Use appendMadrecord to append a new Madrecord
*/
Madcycle * createMadcycle(int cyclenum,
                          int cycleId,
                          char * cycleDesc)



/*****************************************************************
*
* destroyMadcycle   releases an existing Madcycle
*
*   will also free all Madrecords in this cycle
*
*   arguments: pointer to existing Madcycle
```

```
*
*   returns - void
*/
void destroyMadcycle(Madcycle * madcycle)



/*****************************************************************
*
* createMadrecord   creates a new Madrecord
*
*   arguments:
*
*   Rec_type  rectype - Record type: HEADER_REC, CATALOG_REC, or DATA_REC.  If DATA_REC,
*                       text will be empty string, no matter what passed in. If not,
*                       data1Dparms will be null.
*   int numType       - index into maddata.madrecParmTypeList that defines the parm type of
*                       record (that is, its list of 1D and 2D parameters)
*   char   *  text    - Text of header or catalog record. Empty string if data rec.
*   int num1DParms    - Number of 1D parameters to copy
*   double *  data1Dparms - pointer to array of doubles containing 1D data
*   int  kinst  - instrument id
*   double starttime - start time of record in seconds since 1/1/1950
*   double endtime - end time of record in seconds since 1/1/1950
*
*     Number of 1D parameters and order must correspond to
*     maddata.parm1DList
*
*   returns - pointer to newly created Madrecord.  Use
*             destroyMadrecord when done
*
*   Allocates memory to store all data, so all input arrays may be released or changed
*   after this method is called.  Use createMadrecord to create a record with just the
*   1D data; then append each 2D row using append2DRow
*/
Madrecord * createMadrecord(Rec_type rectype,
                            int numType,
                            char * text,
                            int num1DParms,
                            double * data1Dparms,
                            int kinst,
                            double starttime,
                            double endtime)



/*****************************************************************
*
* destroyMadrecord   releases an existing Madrecord
*
*   arguments: pointer to existing Madrecord
*
*   returns - void
*/
void destroyMadrecord(Madrecord * madrecord)



/*****************************************************************
*
* simpleMadrecordPrint - a simple method that prints all data from one Madrecord
```

```
*
*
*   arguments:
*         Maddata * maddata - pointer to Maddata
*         int cycId - cycle number of Madrecord
*         int recId - record number in cycle of Madrecord to print
*         FILE * fp - file to print to (may be stdout)
*
*   returns - void
*
*    Prints simple version of Madrecord to FILE
*/
void simpleMadrecordPrint(Maddata * maddata,
                          int cycId,
                          int recId,
                          FILE * fp)



/**************************************************************************
*
* simpleMadfilterPrint - a simple method that prints all data from Maddata
*                        a single Madfilter
*
*   arguments:
*         Madfilter * madfilter - pointer to Madfilter
*         FILE * fp - file to print to (may be stdout)
*
*   returns - void
*
*    Prints simple version of Madfilter to FILE
*/
void simpleMadfilterPrint(Madfilter * madfilter, int filterNum, FILE * fp)



/**************************************************************************
*
* simpleMaddataPrint - a simple method that prints all data from Maddata
*
*
*   arguments:
*         Maddata * maddata - pointer to Maddata
*         FILE * fp - file to print to (may be stdout)
*
*   returns - void
*
*    Prints simple version of Maddata to FILE
*/
void simpleMaddataPrint(Maddata * maddata, FILE * fp)



/**************************************************************************
*
* classicIsprint - a method that prints all data in standard isprint format
*
*
*   arguments:
*         Maddata * maddata - pointer to Maddata
*         int displayHeaders - if 1, display headers, if 0, don't
```

```
*         int displaySummary - if 1, display summary at top,
*                              if 0, don't
*         int maxCharsPerLine - if 0, no limit, if <ISPRINT_MIN_CHARS_PER_LINE,
*                               limit line to ISPRINT_MIN_CHARS_PER_LINE, else
*                               limit line to maxCharsPerLine
*         char * missingStr - string to use when data missing
*         char * assumedStr - string to use when error data assumed
*         char * knownBadStr - string to use when error data knownBad
*         FILE * fp - file to print to (may be stdout)
*
*   returns - void
*
*    Prints Maddata in standard isprint format to FILE.  Isprint format
*    does not differentiate between 1 and 2D data; everything is treated
*    as 2D data.  Cycle ignored.  If both displayHeaders and displaySummary == 0,
*    only data will be printed without any labels.
*/
void classicIsprint(Maddata * maddata,
                    int displayHeaders,
                    int displaySummary,
                    int maxCharsPerLine,
                    char * missingStr,
                    char * assumedStr,
                    char * knownBadStr,
                    FILE * fp)



/*****************************************************************************
*
* printIsprintHeader - prints isprint header (time and instrument)
*
*
*   arguments:
*         Maddata * maddata - pointer to Maddata
*         int cycleNum - cycle number
*         int recNum - record in that cycle
*         FILE * fp - file to print to (may be stdout)
*
*   returns - void
*
*    Prints isprint header line.
*/
void printIsprintHeader(Maddata * maddata,
                        int cycleNum,
                        int recNum,
                        FILE * fp)



/*****************************************************************************
*
* getIsprintHeader - returns malloced string containing isprint header (time and instrument)
*
*
*   arguments:
*         Maddata * maddata - pointer to Maddata
*         int cycleNum - cycle number
*         int recNum - record in that cycle
*
*   returns - char * to malloced string containing isprint header (time and instrument)
```

```
*
*    Similiar to printIsprintHeader except returns string instead of printing it.
*    User must free returned string when done with it.
*/
char * getIsprintHeader(Maddata * maddata,
                        int cycleNum,
                        int recNum)



/*****************************************************************
*
* printIsprintLabel - prints isprint header
*
*
*   arguments:
*         Maddata * maddata - pointer to Maddata
*         int maxCharsPerLine - limit line to maxCharsPerLine
*         FILE * fp - file to print to (may be stdout)
*
*   returns - void
*
*    Prints isprint header line.
*/
void printIsprintLabel(Maddata * maddata,
                        int maxCharsPerLine,
                        FILE * fp)



/*****************************************************************
*
* getIsprintLabel - creates malloced strings containing isprint label
*                   and comma-separated mnemonics
*
*
*   arguments:
*         Maddata * maddata - pointer to Maddata
*         int maxCharsPerLine - limit line to maxCharsPerLine
*         char ** mnemStr - string containing comma-separated requested mnemonics
*         char ** labelStr - string containing mnemonics labels as formatted for isprint
*
*   returns - void
*
*    User must free malloced strings mnemStr and labelStr when done with them.
*/
void getIsprintLabel(Maddata * maddata,
                     int maxCharsPerLine,
                     char ** mnemStr,
                     char ** labelStr)



/*****************************************************************
*
* classicMadrecordPrint - prints a single Madrecord in isprint format
*
*
*   arguments:
*         Maddata * maddata - pointer to Maddata
*         int cycleNum - cycle number
```

```
*         int recNum - record in that cycle
*         int displayHeaders - if 1, display headers, if 0, don't
*         int maxCharsPerLine - limit line to maxCharsPerLine
*         char * missingStr - string to use when data missing
*         char * assumedStr - string to use when error data assumed
*         char * knownBadStr - string to use when error data knownBad
*         FILE * fp - file to print to (may be stdout)
*
*   returns - void
*
*    Prints isprint header line.
*/
void classicMadrecordPrint(Maddata * maddata,
                           int cycleNum,
                           int recNum,
                           int displayHeaders,
                           int maxCharsPerLine,
                           char * missingStr,
                           char * assumedStr,
                           char * knownBadStr,
                           FILE * fp)



/*****************************************************************
*
* getClassicMadrecordStrings - returns strings describing a single Madrecord
*                              in isprint format
*
*   similar to classicMadrecordPrint except returns data as strings instead
*   of directly fprintf'ing output
*
*
*   arguments:
*         Maddata * maddata - pointer to Maddata
*         int cycleNum - cycle number
*         int recNum - record in that cycle
*         int maxCharsPerLine - limit line to maxCharsPerLine
*         char * missingStr - string to use when data missing
*         char * assumedStr - string to use when error data assumed
*         char * knownBadStr - string to use when error data knownBad
*         char ** headerStr - string containing header as formatted for isprint
*         char ** mnemStr - string containing comma-separated requested mnemonics
*         char ** labelStr - string containing mnemonics labels as formatted for isprint
*         char ** dataStr - string containing data as formatted for isprint, except
*                           rows separated by commas instead of carriage return.
*
*   The strings mnemStr, headerStr, and dataStr are allocated on the heap, and are
*   the resposiblity of the caller to free when no longer needed.
*
*   returns - void
*/
void getClassicMadrecordStrings(Maddata * maddata,
                                int cycleNum,
                                int recNum,
                                int maxCharsPerLine,
                                char * missingStr,
                                char * assumedStr,
                                char * knownBadStr,
                                char ** headerStr,
                                char ** mnemStr,
```

```
                                char ** labelStr,
                                char ** dataStr)


/*****************************************************************
*
* lookerMadrecordPrint - prints a single Madrecord in looker format
*
*
*   arguments:
*         Maddata * maddata - pointer to Maddata (assumed to have one record only)
*         char * missingStr - string to use when data missing
*         char * assumedStr - string to use when error data assumed
*         char * knownBadStr - string to use when error data knownBad
*         int printHeaderFlag - if zero, surpress printing header line
*         FILE * fp - file to print to (may be stdout)
*
*   returns - void
*
*/
void lookerMadrecordPrint(Maddata * maddata,
                          char * missingStr,
                          char * assumedStr,
                          char * knownBadStr,
                          int printHeaderFlag,
                          FILE * fp)



/*****************************************************************
*
* populate1DDataFromStr - uses a command string to populate a MadparmList
*                         and an array of doubles
*
*   For example, if onedString = "gdalt=100.0 glon=45.0 gdlat=-20.0",
*   oneDParms would have gdalt, glon, and gdlat, and oneDdata = {100.0, 45.0, -20.0}
*
*   arguments:
*         char * onedString - string that describes 1D data
*         MadparmList ** oneDParms - created list of 1D parameters
*         double ** oneDdata - pointer to array of doubles.  Memory malloc'ed here
*         and must be free'd by the user when done
*
*   returns - 0 if success, -1 if problem
*
*/
int populate1DDataFromStr(char * onedString,
                          MadparmList ** oneDParms,
                          double ** oneDdata)



/*****************************************************************
*
* populate2DDataFromStr - uses a command string to populate a MadparmList
*                         and an array of doubles
*
*   For example, if twodString is:
*
*      "gdlat=45,45,45,45,50,50,50,50 glon=20,20,30,30,20,20,30,30 gdalt=500,600,500,600,500,60
```

```
*
*   (Note that each parameters must have same number of values or an error is thrown)
*
*   twoDParms would have gdalt, glon, and gdlat, and
*   twoDdata = { {45.0,45.0,45.0,45.0,50.0,50.0,50.0,50.0},
*                {20.0,20.0,30.0,30.0,20.0,20.0,30.0,30.0},
*                {500.0,600.0,500.0,600.0,500.0,600.0,500.0,600.0}}
*
*   arguments:
*         char * twodString - string that describes 2D data
*         MadparmList ** twoDParms - created list of 2D parameters
*         double *** twoDdata - pointer to array of arrays of doubles.  Memory malloc'ed here
*         and must be free'd by the user when done
*         int * num2Drows - number of rows found in each 2D parameter
*
*   returns - number of 2D values per parameter if success, -1 if problem
*
*/
int populate2DDataFromStr(char * twodString,
                          MadparmList ** twoDParms,
                          double *** twoDdata,
                          int * num2Drows)
```

Private madrec and maddata methods

These methods are private to the madc library, and should not need to be used by application developers. They are documented here for maintenance. They involve low-level routines to access various Cedar files, and also the details of how the madDeriveEngine module works

---

getNextMadrigalRecord	putNextMadrigalRecord	getNextCedarAsciiRecord	putNext
getNextCosRecord	putNextCedarCbfRecord	putNextCosRecord	flushCe
endDataCedarCbfRecord	writeCbfControlWord	getNextCedarBlockedRecord	putNext
getNextCedarUnblockedRecord	putNextCedarUnblockedRecord	getMemNextCedarUnblockedRecord	putMem
editMemNextCedarUnblockedRecord	cedarFileType	isProlog	madptr
idmyk1	setCheckSum	int	encodeB
setbit	dumpCedarRecord	fread16	fwrite16
createInfoMethod	destroyInfoMethod	createInfoMultiRowMethod	destroyI
destroyInfoDervFile	getRecType	load1DMeasData	load2DM
load1DMultiRowDataNonfile	load2DMultiRowData	load2DMultiRowDataNonfile	evaluate
updateMadrecordWithMultiRow	append2DRow	createInfoDerived	destroyI
dispatchMultiRowMethod	checkIf1DMethodNeeded	checkIf2DMethodNeeded	make1D
checkIfMultiRowMethodNeeded	hasOutput		

```
/*****************************************************************
*
* getNextMadrigalRecord  reads a madrigal record
*
* The file should be positioned at the beginning of a record before
* calling this function, and on the first call, the Cedar record
* pointer, *cedarRecord, should be NULL. The pointers at the beginning
* of each block (words 1 and 2) and the checksum (word(blockSize-1) are
```

```
* ignored.
*
* Madrec completely ignores the 5 extra control words, words 17-21, in the
* Madrigal format prolog. They are removed when a Madrigal record is read
* and added when a Madrigal record is written.
* These extra words are fully compliant with the CEDAR binary format in
* the case of data records, since the prolog length is specified in the
* data record.  They are not fully compliant with the CEDAR standard in
* the case of binary catalog and header records, which have fixed prolog
* lengths of 40, of which only the first 12 and 15 respectively are
* significant. The standard requires that the remaining words in the
* prolog must be zero. However, in practice this has often been ignored,
* even at NCAR, and it is unlikely that any CEDAR software actually
* chokes at non-zero words beyond 12 and 15 in catalog and header
* records.
*
*    fp          - File pointer to the Madrigal file
*    cedarRecord - The Cedar record
*    blockSize   - Madrigal file block size. Normally 6720 16-bit integers.
*    sigWords    - Number of signifcant words in the current block
*
*/

int
getNextMadrigalRecord (FILE *fp, Int16 **cedarRecordpp,
                       int blockSize,
                       int *sigWordsp)



/*****************************************************************
*
* putNextMadrigalRecord   adds madrigal record to a Madrigal file
*
* Madrigal files written by the Fortran library have an extra 0 after
* the last record. This may serve as an EOF indicator (zero-length record).
* This is not in the specification and is not added by this routine.
* getNextMadrigalRecord handles the extra zero correctly.
*
* Cedar data records may have a shorter or longer prolog than data records
* in a Madrigal file, which always have a 21-word prolog. So, this routine
* transfers data from the input Cedar record to a Madrigal record,
* modifying the prolog as required, and then outputs the Madrigal
* record.
*
*    fp          - File pointer to the Madrigal file
*    cedarRecord - The Cedar record
*    blockSize   -
*    blockIndex  - Madrigal file block size. Normally 6720 16-bit integers.
*    block       - The Madrigal block - normally 2*6720=13440 bytes.
*    prevRec     - Position of previous record in its block
*    thisRec     - Position of current record in its block.
*
*/

int
putNextMadrigalRecord (FILE *fp, Int16 **cedarRecordpp,
                       int blockSize,
                       int *blockIndexp,
                       Int16 **blockpp,
                       int *prevRecp,
```

```
                        int *thisRecp)


/*****************************************************************
*
* getNextCedarAsciiRecord   reads record in CEDAR ASCII format
*
*    fp          - File pointer to the Madrigal file
*    cedarRecord - The Cedar record
*
*/

int
getNextCedarAsciiRecord (FILE *fp, Int16 **cedarRecordpp)



/*****************************************************************
*
* putNextCedarAsciiRecord   writes record in CEDAR ASCII format
*
*    fp          - File pointer to the Madrigal file
*    cedarRecord - The Cedar record
*
*/

int
putNextCedarAsciiRecord (FILE *fp, Int16 **cedarRecordpp)



/*****************************************************************
*
* getNextCedarCbfRecord   Gets the next CEDAR record from a CBF file
*
*    fp          - File pointer to the Madrigal file
*    cedarRecord - The Cedar record
*    blockSize   -
*    lCosBlock   -
*    pos         - position within Cos record, which contains multiple
*                     Cedar records
*    fwi         -
*    cosRecord   -
*
*/
int
getNextCedarCbfRecord(FILE *fp, Int16 **cedarRecordpp,
                      int forceCosRead,
                      int blockSize,
                      int *initPos8p,
                      int *initFwip,
                      int *initPosp,
                      int *initLCosBlockp,
                      int *lCosBlockp,
                      int *posp,
                      int *fwip,
                      Int16 **cosRecordpp)
```

```
/*****************************************************************
*
* getNextCosRecord   Gets the next CEDAR record from a CBF file
*
*    fp          - File pointer to the Madrigal file
*    cedarRecord - The Cedar record
*    fwi         -
*
*/
int
getNextCosRecord(FILE *fp, Int16 **cosRecordpp, int *fwip)



/*****************************************************************
*
* putNextCedarCbfRecord   Puts the next CEDAR record into a CBF file
*
*    fp                  - File pointer to the Madrigal file
*    cedarRecord         - The Cedar record
*    blockSize           - Cos Blocksize (normally 4096)
*    lbuf                -
*    pos                 -
*    cosRecord           -
*    blockNumber         -
*    previousFileIndex   -
*    previousRecordIndex -
*    lastControlWord     -
*
*/
int
putNextCedarCbfRecord(FILE *fp, Int16 **cedarRecordpp,
                      int blockSize,
                      int *lbufp,
                      int *posp,
                      Int16 **cosRecordpp,
                      int *blockNumberp,
                      int *previousFileIndexp,
                      int *previousRecordIndexp,
                      long *lastControlWordp)



/*****************************************************************
*
* putNextCosRecord   Writes the next Cos record to a CBF file
*
*    fp                  - File pointer to the Madrigal file
*    cedarRecord         - The Cedar record
*    blockSize           -
*    lbuf                -
*    blockNumber         -
*    previousFileIndex   -
*    previousRecordIndex -
*    lastControlWord     -
*
*/
int
putNextCosRecord(FILE *fp, Int16 **cosRecordpp,
                 int blockSize,
                 int *lbufp,
```

```
                int *blockNumberp,
                int *previousFileIndexp,
                int *previousRecordIndexp,
                long *lastControlWordp)


/*****************************************************************
*
* flushCedarCbfRecord   Flushes the last CEDAR record into a CBF file
*
*    fp                 - File pointer to the Madrigal file
*    cedarRecord        - The Cedar record
*    blockSize          -
*    lbuf               -
*    blockNumber        -
*    previousFileIndex  -
*    previousRecordIndex -
*    lastControlWord    -
*
*/
int
flushCedarCbfRecord(FILE *fp, Int16 **cedarRecordpp,
                    int blockSize,
                    int *lbufp,
                    int *posp,
                    Int16 **cosRecordpp,
                    int *blockNumberp,
                    int *previousFileIndexp,
                    int *previousRecordIndexp,
                    long *lastControlWordp)


/*****************************************************************
*
* endFileCedarCbfRecord   Writes end-of-file to a CBF file
*
*    fp                 - File pointer to the Madrigal file
*    cedarRecord        - The Cedar record
*    blockSize          -
*    previousFileIndex  -
*    lastControlWord    -
*
*/
int
endFileCedarCbfRecord(FILE *fp, Int16 **cedarRecordpp,
                              int blockSize,
                              int *previousFileIndexp,
                              long *lastControlWordp)


/*****************************************************************
*
* endDataCedarCbfRecord   Writes end-of-data CBF file
*
*    fp                 - File pointer to the Madrigal file
*    cedarRecord        - The Cedar record
*    blockSize          -
*    lastControlWord    -
```

```
*
*/
int
endDataCedarCbfRecord(FILE *fp, Int16 **cedarRecordpp,
                                int blockSize,
                                long *lastControlWordp)



/*****************************************************************
*
* writeCbfControlWord   Gets the next CEDAR record from a CBF file
*
*    fp           - File pointer to the Madrigal file
*    cedarRecord  - The Cedar record
*    m            -
*    bdf          -
*    bn           -
*    fwi          -
*    ubc          -
*    pfi          -
*    pri          -
*/
int
writeCbfControlWord(FILE *fp,
                    int m,
                    int bdf,
                    int bn,
                    int ubc,
                    int pfi,
                    int pri,
                    long *lastControlWordp)



/*****************************************************************
*
* getNextCedarBlockedRecord   Gets the next CEDAR record from a file
*
*    fp          - File pointer to the Madrigal file
*    cedarRecord - The Cedar record
*    lBlockp     - pointer to the length of the current block
*    posp        - pointer to the position within the current block
*
*/
int
getNextCedarBlockedRecord(FILE *fp, Int16 **cedarRecordpp,
                          Int16 *lBlockp,
                          int *posp)



/*****************************************************************
*
* putNextCedarBlockedRecord   Puts the next CEDAR record into a file
*
*    fp          - File pointer to the Madrigal file
*    cedarRecord - The Cedar record
*    maxBlock    -
*    lbuf        -
*    posp        -
```

```
*    block       -
*/
int
putNextCedarBlockedRecord(FILE *fp, Int16 **cedarRecordpp,
                          int *maxBlock,
                          int *lbufp,
                          int *posp,
                          Int16 **blockpp)



/*****************************************************************
*
* flushCedarBlockedRecord   Flushes the last CEDAR record into a CBF file
*
*    fp          - File pointer to the Madrigal file
*    cedarRecord - The Cedar record
*    maxBlock    -
*    lbuf        -
*    posp        -
*    block       -
*
*/
int
flushCedarBlockedRecord(FILE *fp, Int16 **cedarRecordpp,
                        int *maxBlock,
                        int *lbufp,
                        int *posp,
                        Int16 **blockpp)



/*****************************************************************
*
* getNextCedarUnblockedRecord   Gets the next CEDAR record from a file
*
*    fp          - File pointer to the Madrigal file
*    cedarRecord - The Cedar record
*
*/
int
getNextCedarUnblockedRecord(FILE *fp, Int16 **cedarRecordpp)



/*****************************************************************
*
* putNextCedarUnblockedRecord   Puts the next CEDAR record into a file
*
*    fp          - File pointer to the Madrigal file
*    cedarRecord - The Cedar record
*
*/
int
putNextCedarUnblockedRecord(FILE *fp, Int16 **cedarRecordpp)



/*****************************************************************
*
* getMemNextCedarUnblockedRecord   Gets the next CEDAR record from memory
```

```
*
*    cedarFilepp   - The block of memory holding a list of Cedar records
*    cedarRecordpp - The Cedar record to be populated from cedarFilepp
*    posp          - The present number of Int16's already read from cedarFilepp
*    recordpInMem  - Determines whether cedarRecordpp is pointing into cedarFilepp
*                    or separate block of memory on the heap
*
*/
int getMemNextCedarUnblockedRecord(Int16 **cedarFilepp,
                                   Int16 **cedarRecordpp,
                                   int *posp,
                                   int *recordpInMem)



/*****************************************************************
*
* putMemNextCedarUnblockedRecord   Puts the next CEDAR record into memory
*
*    cedarFilepp   - The block of memory being filled with a list of Cedar records
*    cedarRecordpp - The Cedar record to be appended to cedarFilepp
*    fileSizep     - The present number of bytes in cedarFilepp
*    posp          - The present number of Int16's in cedarFilepp, not
*                    including trailing 0 (see below)
*
*    To indicate this is last record, append Int16 = 0 at end,
*    so ltot will be zero (last record indicator)
*/
int
putMemNextCedarUnblockedRecord(Int16 **cedarFilepp, Int16 **cedarRecordpp,
                               int *fileSizep, int *posp)



/*****************************************************************
*
* putMemFastNextCedarUnblockedRecord   Puts the next CEDAR record into memory with
*                                      fewer realloc calls.  Allocates more memory
*                                      than needed to increase speed.  Is meant to be
*                                      a private method only called from madrecOpen.
*                                      Requires that cedarFilepp be realloc'ed to right
*                                      size after file is fully loaded into memory
*
*    cedarFilepp   - The block of memory being filled with a list of Cedar records
*    cedarRecordpp - The Cedar record to be appended to cedarFilepp
*    fileSizep     - The present number of bytes in cedarFilepp
*    posp          - The present location of Int16 pointer in file
*    memPos        - The present location of Int16 pointer in memory
*
*    To indicate this is last record, append Int16 = 0 at end,
*    so ltot will be zero (last record indicator)
*/
int putMemFastNextCedarUnblockedRecord(Int16 **cedarFilepp,
                                       Int16 **cedarRecordpp,
                                       int *fileSizep,
                                       int *posp,
                                       int *memPos)



/*****************************************************************
```

```
*
* editMemNextCedarUnblockedRecord   Gets pointer to the next CEDAR record
*
*    cedarFilepp   - The block of memory holding a list of Cedar records
*    cedarRecordpp - The Cedar record to be editted - will point to somewhere in
*                    cedarFilepp
*    posp          - The record in cedarFilepp for cedarRecordpp to point at
*    recordpInMem  - Determines whether cedarRecordpp is pointing into cedarFilepp
*                    or separate block of memory on the heap
*
*/
int
editMemNextCedarUnblockedRecord(Int16 **cedarFilepp,
                                Int16 **cedarRecordpp,
                                int *posp,
                                int *recordpInMem)



/*************************************************************************
*
* cedarFileType   Gets Cedar File Type
*
*    Supported formats:
*        0 - Madrigal
*        1 - Blocked Binary
*        2 - Cbf
*        3 - Unblocked Binary
*        4 - Ascii
*
*    Supported record types:
*        0 - Catalog
*        1 - Header
*        2 - Data
*
*/

int
cedarFileType(char *fileName, int madrigalBlockSize, int cbfBlockSize)



/*************************************************************************
*
* isProlog - Checks for consistent Cedar prolog
*   prolog - Pointer to the prolog
*
*/
int
isProlog(Int16 *prolog)



/*************************************************************************
*
*/
int
madptr(Int16 *time)
```

```
/*****************************************************************
*
*
*
*/
int
jday1(int day, int month, int year)



/*****************************************************************
*
*
*
*/
int
idmyk1(int day, int month, int year)



/*****************************************************************
*
* setCheckSum   sets block checksum
*
*/

int
setCheckSum (int blockSize, Int16 **blockpp)



/*****************************************************************
*
* decodeBits    Returns bits b1 to b2 of buf as an unsigned integer
*
*/

unsigned int
decodeBits(unsigned char *buf, unsigned int b1, unsigned int b2)



/*****************************************************************
*
* encodeBits    Encodes unsigned integer val in bits b1 to b2 of buf
*
*/

void
encodeBits(unsigned char *buf, unsigned int b1, unsigned int b2, unsigned int val)



/*****************************************************************
*
* getbit    returns 1 if bit b in array buf is 1, else 0.
*
*/

unsigned int
getbit(unsigned char *buf, unsigned int b)
```

```
/*****************************************************************
*
* setbit    Sets bit b in array buf to last bit in bv.
*           Thus, if bv=0, bit b in buf will be set to 0,
*            else if bv=1, bit b in buf will be set to 1.
*           No other bit in buf will be changed.
*
*/

void
setbit(unsigned char *buf, unsigned int b, unsigned int bv)



/*****************************************************************
*
* dumpCedarRecord    Dumps beginning and end of Cedar record
*
*/

void
dumpCedarRecord(Int16 **recordpp, char *title)



/*****************************************************************
*
* fread16   Reads big-endian 16-bit integers
*
*    ptr           - Pointer to array of 16-bit integers
*    size_t size   - Must be 2
*    size_t nitems - Number of 16-bit integers to be read
*    stream        - Pointer to input file. The file should be
*                    positioned at the beginning of an sequence of at
*                    least nitems big-endian 16-bit integers.
*
*    The CEDAR format represents data stored on tape or disk as 16-bit big
*    endian integers. fread16 is endian-neutral. It works on both big endian
*    and little endian computers. However, this flexibility imposes a
*    performance penalty on big endian computers. This penalty can be
*    reduced by changing the BIGENDIAN constant to 1.
*
*/

size_t
fread16(void *ptr, size_t size, size_t nitems, FILE *stream)



/*****************************************************************
*
* fwrite16   Writes big-endian 16-bit integers
*
*    ptr           - Pointer to array of 16-bit integers
*    size_t size   - Must be 2
*    size_t nitems - Number of 16-bit integers to write
*    stream        - Pointer to input file. Nitems 16-bit integers will
*                    be written to stream in big-endian order.
```

```
*
*    The CEDAR format represents data stored on tape or disk as 16-bit big
*    endian integers. fread16 is endian-neutral. It works on both big endian
*    and little endian computers. However, this flexibility imposes a
*    performance penalty on big endian computers. This penalty can be
*    reduced by changing the BIGENDIAN constant to 1.
*
*/

size_t
fwrite16(void *ptr, size_t size, size_t nitems, FILE *stream)



/*****************************************************************
*
* reorderBytes   returns byte order of original file
*
*    ptr           - Pointer to Int16 array
*    numInt16s     - Size of ptr Int16 array
*
*    orderBytes undoes the effect of using fread16, and returns an array of
*    chars read via fread16 to its original order as found in the file.  This is
*    used in reading header and catalog records, where the data is read as chars
*    and not as Int16's.  For big-endian machines it does nothing, for little-endian
*    machines it switches adjacent bytes.
*
*/
void reorderBytes(Int16 * ptr, int numInt16s)



/*****************************************************************
*
* createInfoMethod - sets up a InfoMethod struct that describes one
*                    particular method used to derive parameters
*
*   arguments:
*      int numInputs - the number of inputs required
*      int numOutputs - the number of outputs required
*
*   Number of inputs and outputs defined in gCompExtList.
*
*   returns - pointer to newly created InfoMethod struct; destroy
*             using destroyInfoMethod
*/
InfoMethod * createInfoMethod(int numInputs, int numOutputs)



/*****************************************************************
*
* destroyInfoMethod - frees all memory for given InfoMethod struct
*
*   arguments:
*      InfoMethod * infoMethod - pointer to struct to free
*
*   returns - void
*/
void destroyInfoMethod(InfoMethod * infoMethod)
```

```
/*****************************************************************
*
* createInfoMultiRowMethod - sets up a InfoMultiRowMethod struct that describes one
*                    particular MultiRow method used to derive parameters
*
*   arguments:
*      int numInputs - the number of inputs required
*      int * inputType - an array of ints = 1 or 2, depending in 1 or 2D input,
*                        len =  numInputs
*      int numOutputs - the number of outputs required
*      int * outputType - an array of ints = 1 or 2, depending in 1 or 2D output,
*                        len =  numOutputs
*
*   Number of inputs and outputs and types defined in gCompExtMultiRowList.
*
*   Creates inputArr and outputArr.  For now they are arrays of pointers to
*   arrays of doubles of length 1 double.  If this method actually turns out to
*   be needed, the 2D arrays will be reallocated to MAX_2D_ROWS
*
*   returns - pointer to newly created InfoMultiRowMethod struct; destroy
*             using destroyInfoMultiRowMethod
*/

InfoMultiRowMethod * createInfoMultiRowMethod(int numInputs,
                                              const int * inputType,
                                              int numOutputs,
                                              const int * outputType)



/*****************************************************************
*
* destroyInfoMultiRowMethod - frees all memory for given InfoMultiRowMethod struct
*
*   arguments:
*      InfoMultiRowMethod * infoMultiRowMethod - pointer to struct to free
*
*   returns - void
*/
void destroyInfoMultiRowMethod(InfoMultiRowMethod * infoMultiRowMethod)



/*****************************************************************
*
* createInfoDervFile - sets up a InfoDervFile struct in preparation for
*                      calculating all derived parameters for a file
*
*   arguments:
*      madrec * madrecp - pointer to madrec struct that has an in-memory file
*      MadparmList * requestParm - list of parameters requested
*      MadfilterList * filtList - list of Madfilters to apply
*      FILE * errFile   - errFile to write an error messages to
*
*   Examines each record in the in-memory file to get measured 1d and 2d parameters.
*   If first time that unique record type found, calls createInfoDerived
*   to create an analysis plan.  For each record, records its type and
*   location in returned struct InfoDervFile.  Call destroyInfoDervFile
*   when done with this struct.
```

```
*
*   returns - pointer to created InfoDervFile struct; if failure,
*             returns NULL and writes error to errFile
*/
InfoDervFile * createInfoDervFile(Madrec * madrecp,
                                  MadparmList * requestParm,
                                  MadfilterList * filtList,
                                  FILE * errFile)



/**************************************************************************
*
* destroyInfoDervFile - frees all memory for given InfoDervFile struct
*
*   arguments:
*      InfoDervFile * info - pointer to struct to free
*
*   returns - void
*/
void destroyInfoDervFile(InfoDervFile * infoDervFile)



/**************************************************************************
*
* getRecType - returns record type index in InfoDervFile for recno
*
*   arguments:
*      InfoDervFile * infoDervFile - pointer to InfoDervFile containing analysis
*      int recno - record number in file (starting at 0)
*
*   returns - record type index in InfoDervFile for recno.  If not
*             found, returns -1
*/
int getRecType(InfoDervFile * infoDervFile, int recno)



/**************************************************************************
*
* load1DMeasData - copies 1D measured data into infoDervFile's memory
*
*   arguments:
*      InfoDervFile * infoDervFile - pointer to InfoDervFile containing analysis
*      Madrec * madrecp - pointer to Madrec holding data from file
*      int recType - record type of present record
*      double first_ibyr - year of first record (faster than rewinding to get these four)
*      double first_ibdt - date of first record
*      double first_ibhm - hm*100+min of first record
*      double first_ibcs - centisecs of first record
*
*   returns - void
*
*   affects - copies measured 1D data into appropriate place in
*             infoDervFile->infoDervList[recType]->all1DParm
*/
void load1DMeasData(InfoDervFile * infoDervFile,
                    Madrec * madrecp,
                    int recType,
                    double first_ibyr,
```

```
                    double first_ibdt,
                    double first_ibhm,
                    double first_ibcs)


/*****************************************************************
*
* load2DMeasData - copies 2D measured data into infoDervFile's memory
*
*   arguments:
*      InfoDervFile * infoDervFile - pointer to InfoDervFile containing analysis
*      Madrec * madrecp - pointer to Madrec holding data from file
*      int recType - record type of present record
*      int row - 2D row to load (starts at 0)
*
*   returns - void
*
*   affects - copies measured 2D data into appropriate place in
*             infoDervFile->infoDervList[recType]->all2DParm
*/
void load2DMeasData(InfoDervFile * infoDervFile,
                    Madrec * madrecp,
                    int recType,
                    int row)


/*****************************************************************
*
* load1DMultiRowData - copies 1D data from infoDervFile into
*                      InfoMultiRowMethod->inputArr for all needed
*                      multi-row methods
*
*   arguments:
*      InfoDervFile * infoDervFile - pointer to InfoDervFile containing analysis
*      int recType - record type of present record
*
*   returns - void
*
*   affects - copies 1D data from infoDervFile into
*             InfoMultiRowMethod->inputArr
*/
void load1DMultiRowData(InfoDervFile * infoDervFile,
                        int recType)


/*****************************************************************
*
* load1DMultiRowDataNonfile - copies 1D data from infoDerv into
*                             InfoMultiRowMethod->inputArr for all needed
*                             multi-row methods - used when no file used
*
*   arguments:
*      InfoDerived * infoDerv - pointer to InfoDerived containing analysis
*
*   returns - void
*
*   affects - copies 1D data from infoDerv into
*             InfoMultiRowMethod->inputArr
```

```
*/
void load1DMultiRowDataNonfile(InfoDerived * infoDerived)



/*****************************************************************
*
* load2DMultiRowData - copies 2D data from infoDervFile into
*                      InfoMultiRowMethod->inputArr for all needed
*                      multi-row methods
*
*   arguments:
*      InfoDervFile * infoDervFile - pointer to InfoDervFile containing analysis
*      int recType - record type of present record
*      int count2D - index into 2D record being added
*
*   returns - void
*
*   affects - copies 2D data from infoDervFile into
*             InfoMultiRowMethod->inputArr
*/
void load2DMultiRowData(InfoDervFile * infoDervFile,
                        int recType,
                        int count2D)



/*****************************************************************
*
* load2DMultiRowDataNonfile - copies 2D data from infoDerived into
*                             InfoMultiRowMethod->inputArr for all needed
*                             multi-row methods - used when no file used
*
*   arguments:
*      InfoDerived * infoDerived - pointer to InfoDerived containing analysis
*      int count2D - index into 2D record being added
*
*   returns - void
*
*   affects - copies 2D data from infoDerive into
*             InfoMultiRowMethod->inputArr
*/
void load2DMultiRowDataNonfile(InfoDerived * infoDerived,
                               int count2D)



/*****************************************************************
*
* evaluateFilter - returns 1 if filter accepts, 0 if fails
*
*   arguments:
*      InfoDervFile * infoDervFile - pointer to InfoDervFile containing analysis
*      int recType - record type of present record
*      int dim - 1 for 1D filter, 2 for 2D filter
*      int filtIndex - index into filter list filt1DList or filt2DList, which
*                      determines which filter to apply
*
*   returns - 1 if filter accepts, 0 if fails
*
*   notes - if any filter parameter is missing, filter fails
```

```
*/
int evaluateFilter(InfoDervFile * infoDervFile,
                   int recType,
                   int dim,
                   int filtIndex)



/*****************************************************************
*
* appendMadrecord   appends a new Madrecord to Maddata
*
*   arguments:
*
*      Maddata * maddata - pointer to Maddata to append to
*      InfoDerived * infoDeriv - pointer to InfoDerived that contains the data
*      int cycIndex      - index of cycle to add Madrecord to
*      int typeIndex     - index of type of record. Refers to maddata.madrecParmTypeList
*      Rec_type  rectype - Record type: HEADER_REC, CATALOG_REC, or DATA_REC.  If DATA_REC,
*                          text will be empty string, no matter what passed in. If not,
*                          data1Dparms will be null.
*      char   *  text    - Text of header or catalog record. Empty string if data rec.
*      int  kinst        - instrument id
*      double starttime  - start time of record in seconds since 1/1/1950
*      double endtime    - end time of record in seconds since 1/1/1950
*
*   returns - index of MadRecord added if successful, starting at 0, -1 if error
*/
int appendMadrecord(Maddata * maddata,
                    InfoDerived * infoDerv,
                    int cycIndex,
                    int typeIndex,
                    Rec_type rectype,
                    char * text,
                    int kinst,
                    double starttime,
                    double endtime)



/*****************************************************************
*
* updateMadrecordWithMultiRow   modifies an existing Madrecord with data from a multi-row metho
*
*   arguments:
*
*      Maddata * maddata - pointer to Maddata to append to
*      InfoMultiRowMethod * infoMultiRowMeth - pointer to InfoMultiRowMethod containing new dat
*      int methIndex     - index into which multi-row method this is
*      int cycIndex      - index of cycle that Madrecord is in
*      int recIndex      - index into which record in cycle is being modified
*      int typeIndex     - index of type of record. Refers to maddata.madrecParmTypeList
*      int numRows       - number of 2D rows being updated
*
*   returns - 0 if successful, -1 if error
*/
int updateMadrecordWithMultiRow(Maddata * maddata,
                                InfoMultiRowMethod * infoMultiRowMeth,
                                int methIndex,
                                int cycIndex,
                                int recIndex,
```

```
                                int typeIndex,
                                int numRows)


/*****************************************************************
*
* append2DRow   appends a row of 2D data to a Maddata->Madcycle->Madrecord
*
*   arguments:
*
*      Maddata * maddata - pointer to Maddata to append to
*      InfoDerived * infoDeriv - pointer to InfoDerived that contains the data
*      int cycIndex      - index of cycle to add Madrecord to
*      int recIndex      - index of Madrecord in cycle to append 2D
*      int typeIndex     - index of type of record. Refers to maddata.madrecParmTypeList
*
*   returns - index of 2D row added if successful, starting at 0, -1 if error
*/
int append2DRow(Maddata * maddata,
                InfoDerived * infoDerv,
                int cycIndex,
                int typeIndex,
                int recIndex)



/*****************************************************************
*
* createInfoDerived - sets up a InfoDerived struct in preparation for
*                     calculating all derived parameters for a given
*                     record type
*
*   A record type is a unique ordered combination of 1d measured parameters and
*   2d measured parameters
*
*   arguments:
*      MadparmList * meas1DParmList - list of measured 1D parameters
*      MadparmList * meas2DParmList - list of measured 2D parameters
*      MadparmList * requestedParmList - list of parameters requested
*      MadfilterList * filtList - list of filters requested
*
*   Searchs through the information in gCompExtList to determine which
*   methods need to be called for 1D and 2D cases.  Determines which
*   requested parameters can not be derived.  Sets up allocated memory
*   to hold all measured and derived parameters. Sets up inputMap and
*   outputMap to rapidly move data into input and output arrays required
*   for each method.  Free using destroyInfoDerived.
*
*   returns - pointer to created InfoDerived struct; if failure,
*             returns NULL
*/
InfoDerived * createInfoDerived(MadparmList * meas1DParmList,
                                MadparmList * meas2DParmList,
                                MadparmList * requestedParmList,
                                MadfilterList * filtList)



/*****************************************************************
*
```

```
* destroyInfoDerived - frees all memory for given InfoDerived struct
*
*   arguments:
*      InfoDerived * info - pointer to struct to free
*
*   returns - void
*/
void destroyInfoDerived(InfoDerived * info)



/*****************************************************************
*
* dispatchMethod - calls the derived method set by methIndex
*
*   arguments:
*      InfoDerived * infoDeriv - pointer to struct to update by
*                                calling method
*      int methIndex - index of method being run
*      FILE * errFile - error file for methods to write to
*
*   writes error message to errFile if error occurs
*
*   returns - 0 if no error thrown by underlying method
*/
int dispatchMethod(InfoDerived * infoDeriv, int methIndex, FILE * errFile)



/*****************************************************************
*
* dispatchMultiRowMethod - calls the derived multi-row method set by methIndex
*
*   arguments:
*      InfoDerived * infoDeriv - pointer to struct to update by
*                                calling method
*      int methIndex - index of method being run
*      FILE * errFile - error file for methods to write to
*
*   writes error message to errFile if error occurs
*
*   returns - 0 if no error thrown by underlying method
*/
int dispatchMultiRowMethod(InfoDerived * infoDeriv,
                           int methIndex,
                           int numRows,
                           FILE * errFile)



/*****************************************************************
*
* checkIf1DMethodNeeded - if 1D method is needed, calls make1DMethodNeeded
*
*   arguments:
*      InfoDerived * infoDeriv - pointer to struct to update if
*                                method is needed
*      int methIndex - index of method being checked
*      MadparmList filtParmsNotRequestedList - additional parameters needed
*
*     Method being checked has already been found to be one
```

```
*     that can be used successfully given measured 1D parameter
*     and outputs of earlier derived methods.
*
*   returns - void
*/
void checkIf1DMethodNeeded(InfoDerived * infoDeriv,
                           int methIndex,
                           MadparmList * filtParmsNotRequestedList)



/*****************************************************************
*
* checkIf2DMethodNeeded - if 2D method is needed, calls make2DMethodNeeded
*
*   arguments:
*      InfoDerived * infoDeriv - pointer to struct to update if
*                                method is needed
*      int methIndex - index of method being checked
*      MadparmList filtParmsNotRequestedList - additional parameters needed
*
*     Method being checked has already been found to be one
*     that can be used successfully given measured 1 and 2D parameters
*     and outputs of earlier derived methods.
*
*   returns - void
*/
void checkIf2DMethodNeeded(InfoDerived * infoDeriv,
                           int methIndex,
                           MadparmList * filtParmsNotRequestedList)



/*****************************************************************
*
* make1DMethodNeeded - sets this method and all it depends on as needed
*
*   arguments:
*      InfoDerived * infoDeriv - pointer to struct to update
*      int methIndex - index of method need
*
*     This is a recursive method that returns only after all dependent
*     methods have been set as needed.  It adds all input and output
*     parameters to allUsed1DParmList if not there already.
*
*   returns - void
*/
void make1DMethodNeeded(InfoDerived * infoDeriv, int methIndex)



/*****************************************************************
*
* make2DMethodNeeded - sets this method and all it depends on as needed
*
*   arguments:
*      InfoDerived * infoDeriv - pointer to struct to update
*      int methIndex - index of method need
*
*     This is a recursive method that returns only after all dependent
*     methods have been set as needed.  It adds all input and output
```

```
*     parameters to allUsed2DParmList or allUsed2DParmList if not there already.
*
*   returns - void
*/
void make2DMethodNeeded(InfoDerived * infoDeriv, int methIndex)



/*****************************************************************
*
* checkIfMultiRowMethodNeeded - makes multi-row method needed if required
*
*   arguments:
*      InfoDerived * infoDeriv - pointer to struct to update if
*                                multi-row method is needed
*      int methIndex - index of multi-row method being checked
*
*     Multi-row method being checked has already been found to be one
*     that can be used successfully given measured 1 and 2D parameters
*     and outputs of earlier derived methods. Reallocates InfoMultiRowMethod->
*     inputArr and outputArr if needed, and sets up inputMap.
*
*   returns - void
*/
void checkIfMultiRowMethodNeeded(InfoDerived * infoDeriv,
                                 int methIndex)



/*****************************************************************
*
* hasOutput - returns 1 if CompiledExt outputs given mnem, 0 otherwise
*
*   arguments:
*      const CompiledExt * ext - pointer to compiled extension being analyzed
*      const char *  mnem - pointer to mnemonic
*
*   returns  1 if CompiledExt outputs given mnem, 0 otherwise
*/
int hasOutput(const CompiledExt * exten, const char * mnem)
```

Methods to derive Madrigal parameters

These methods are used to derive Madrigal parameters from other Madrigal parameters. They are documented here to fully describe the derivation algorithms.

checkErrorData	getDebyeFactor	getElecDensity	getTsyganenkoField	traceTsyganenkoField
getTsyganenkoG1Index	getTsyganenkoG2Index	traceMagneticField	run_iri	faraday_rotation
getByear	getTime	getBmd	getBMonthDay	getMd
getUtUnix	getDayno	getBhm	getBhhmmss	getEhhmmss
getHm	getUth	getUts	getBUth	getInttms
getInttmm	getDatntd	getUt	getBegUt	getJdayno
getJulian_date	getUt1	getUt2	getDut21	getFyear
getStation	getAltInc	getAveAlt	getAveDAlt	getResl

getAzmDaz getDAzmDDaz getElmDel getDElmDDel getGeod
getDGeod getGeodGdalt getGeodAlt getAzElRange getSZen
getSltmut getSlt getSdwHt getSuntime getTecGdalt
getGcdist getMag getGeocgm getTsygan getAacgm
fromAacgm getMlt getEregion getAspect getSltc
getAplt getSZenc getConjSun getGeo getDst
getFof2 getPopl getPop getNel getNe
getDNel getDNe getNemaxl getNemax getTr
getTe getTi getDteCctitr getDte getCol
getCo getNeNel getDNeDNel getVisrNe getVisrTe
getVisrTi getVisrVo getVisrHNMax getVisrNeDiff getVisrNelDiff
getVisrTeDiff getVisrTiDiff getVisrVoDiff getSn getSnp3
getChip31 getWchsq1 getChisq1 getChip32 getWchsq2
getChisq2 getVi1Vi1f getVi2Vi2f getVipeVipe1 getVipeVipe2
getVipnVipn1 getVipnVipn2 getVi6Vipu getViGeom getViGeod
getVn1Vn1p2 getVn2Vn2p2 getVnGeom getVnGeod getEFGeom
getEFGeod getJGeom getJGeod getNeut getTn
getTnNoPhp getDTn getCond getDCond getImf
getIri getTestAveAlt

```
/**************************************************************************
*
* checkErrorData - a helper method that looks at input data for methods
*                  that calculate error parameters to find assumed or
*                  knownbad special values.  If found, all outputs
*                  set to missing and return 1.  If not, return 0.
*
*   arguments:
*      inCount - num inputs
*      inputArr - double array
*      outCount - num outputs
*      outputArr - double array
*
*
*   returns - 1 if any input assumed or knownbad, 0 otherwise
*/
int checkErrorData(int inCount,
                   double * inputArr,
                   int outCount,
                   double * outputArr)



/**************************************************************************
*
* getDebyeFactor - a helper method that finds the debye length g given
*                  Tr and Pfac.
*
*   arguments:
*      double Tr - temperature ratio Te/Ti
*      double Pfac - a factor determined from the uncorrected electron density
*
*   Returns:
```

```
*      double g which is a solution to g(1+Tr+g)(1+g)-Pfac=0
*
*      if fails to converge, returns 0.0
*/
double getDebyeFactor(double Tr, double Pfac)



/**********************************************************************
*
* getElecDensity - a helper method that finds the corrected electron density
*                  given Ti, Tr, the log10 of the uncorrected electron
*                  density Popl in lg(m^-3), and the aspect angle.
*
*   Algorithm from Fortran method NELCAL in madlib, modified with
*   aspect angle dependence using algorithm from F.S. Rodrigues and
*   Dave Hysell.  Rodriques/Hysell used if beam within 6 degrees of
*   perpendicular to magnetic field line (84 <aspect  <96).  Fails
*   if within 1 degree of perpendicular.
*
*   arguments:
*      double Ti - ion temperature
*      double Tr - temperature ratio Te/Ti
*      double Popl - the uncorrected electron density Popl in lg(m^-3)
*      double aspect - magnetic field line
*
*   Returns:
*      double - the log10 of the corrected electron density in lg(m^-3)
*      if fails, returns missing
*/
double getElecDensity(double Ti, double Tr, double Popl, double aspect)



/**********************************************************************
*
* getTsyganenkoField - a helper method that finds the XGSM and YGSM point
*     on the equatorial plane for the field line determined by the given
*     point in space and time using the Tsyganenko model.
*
*    Note that the 2001 Tsyganenko model uses IMF and solar wind
*    speed measurements taken every 5 minutes for a hour.  Since
*    Madrigal presently only has hourly measurements, we'll just average
*    two measurements instead of 12.
*
*    Since Tsyganenko uses globals, this code is not thread safe.
*
*
*   arguments:
*      double time - time in seconds since 1/1/1950
*      double gdlat - geodetic latitude
*      double glon - geodetic longitude
*      double gdalt - geodetic altitude in km
*      double swspd_now - solar wind speed now in m/s
*      double swspd_1hour - solar wind speed 1 hour earlier in m/s
*      double imf_ygsm_now - imf in y gsm direction in nTesla measured now
*      double imf_ygsm_1hour - imf in y gsm direction in nTesla measured 1 hour ago
*      double imf_zgsm_now - imf in z gsm direction in nTesla measured now
*      double imf_zgsm_1hour - imf in z gsm direction in nTesla measured 1 hour ago
*      double swden - solar wind density in m^-3
*      double dst - dst in  nTesla
```

```
*           (the following are output parameters)
*      double * eq_xgsm - x point in equatorial plane where field line crosses (in GSM)
*      double * eq_ygsm - y point in equatorial plane where field line crosses (in GSM)
*      double * eq_xgse - x point in equatorial plane where field line crosses (in GSE)
*      double * eq_ygse - y point in equatorial plane where field line crosses (in GSE)
*
*      If failure, all b* parameters will be set to missing
*
*
*   Returns:
*      double - 0 if success, -1 if failure
*/
int getTsyganenkoField(double time,
                       double gdlat,
                       double glon,
                       double gdalt,
                       double swspd_now,
                       double swspd_1hour,
                       double imf_ygsm_now,
                       double imf_ygsm_1hour,
                       double imf_zgsm_now,
                       double imf_zgsm_1hour,
                       double swden,
                       double dst,
                       double * eq_xgsm,
                       double * eq_ygsm,
                       double * eq_xgse,
                       double * eq_ygse)



/*************************************************************************
*
* traceTsyganenkoField - a helper method that finds the point on the
*     magnetic field line determined by  the qualifiers: conjugate, north_alt,
*     south_alt, apex, or GSM XY plane for the field line determined by the given
*     point in space and time using the Tsyganenko model.
*
*    Note that the 2001 Tsyganenko model uses IMF and solar wind
*    speed measurements taken every 5 minutes for a hour.  Since
*    Madrigal presently only has hourly measurements, we'll just average
*    two measurements instead of 12.
*
*    Since Tsyganenko uses globals, this code is not thread safe.
*
*   arguments:
*      double time - time in seconds since 1/1/1950
*      double gdlat - input geodetic latitude
*      double glon - input geodetic longitude
*      double gdalt - input geodetic altitude in km
*      double swspd_now - solar wind speed now in m/s
*      double swspd_1hour - solar wind speed 1 hour earlier in m/s
*      double imf_ygsm_now - imf in y gsm direction in nTesla measured now
*      double imf_ygsm_1hour - imf in y gsm direction in nTesla measured 1 hour ago
*      double imf_zgsm_now - imf in z gsm direction in nTesla measured now
*      double imf_zgsm_1hour - imf in z gsm direction in nTesla measured 1 hour ago
*      double swden - solar wind density in m^-3
*      double dst - dst in  nTesla
*      int qualifier - 0 for conjugate, 1 for north_alt, 2 for south_alt, 3 for apex, 4 for GSM
*      double * stopAlt - altitude to stop trace at, if qualifier is north_alt or south_alt.
*                         If other qualifier, this parameter is ignored
```

```
*           (the following are output parameters)
*      double * end_gdlat (if qualifier == 4 (GSM XY plane), this will be XGSM instead)
*      double * end_glon (if qualifier == 4 (GSM XY plane), this will be YGSM  instead)
*      double * end_gdalt (if qualifier == 4 (GSM XY plane), this will be ZGSM = 0 instead)
*
*      If failure, all end* parameters will be set to missing
*
*
*   Returns:
*      double - 0 if success, -1 if failure
*/
int traceTsyganenkoField(double time,
                         double gdlat,
                         double glon,
                         double gdalt,
                         double swspd_now,
                         double swspd_1hour,
                         double imf_ygsm_now,
                         double imf_ygsm_1hour,
                         double imf_zgsm_now,
                         double imf_zgsm_1hour,
                         double swden,
                         double dst,
                         int    qualifier,
                         double stopAlt,
                         double * end_gdlat,
                         double * end_glon,
                         double * end_gdalt)



/*****************************************************************
*
* getTsyganenkoG1Index - a helper method that calculates the Tsyganenko G1 Index
*    as defined in ftp://nssdcftp.gsfc.nasa.gov/models/magnetospheric/tsyganenko/
*                        2001paper/Paper2/2001ja000220.pdf.
*
*    Note that the 2001 Tsyganenko G1 index uses IMF and solar wind
*    speed measurements taken every 5 minutes for a hour.  Since
*    Madrigal presently only has hourly measurements, we'll just average
*    two measurements instead of 12.
*
*
*   arguments:
*      double swspd_now - solar wind speed now in m/s
*      double swspd_1hour - solar wind speed 1 hour earlier in m/s
*      double imf_ygsm_now - imf in y gsm direction in nTesla measured now
*      double imf_ygsm_1hour - imf in y gsm direction in nTesla measured 1 hour ago
*      double imf_zgsm_now - imf in z gsm direction in nTesla measured now
*      double imf_zgsm_1hour - imf in z gsm direction in nTesla measured 1 hour ago
*
*
*   Returns:
*      double - G1 index.  If problem, returns missing.
*/
double getTsyganenkoG1Index(double swspd_now,
                            double swspd_1hour,
                            double imf_ygsm_now,
                            double imf_ygsm_1hour,
                            double imf_zgsm_now,
                            double imf_zgsm_1hour)
```

```
/**************************************************************************
*
* getTsyganenkoG2Index - a helper method that calculates the Tsyganenko G2 Index
*    as defined in ftp://nssdcftp.gsfc.nasa.gov/models/magnetospheric/tsyganenko/
*                        2001paper/Paper2/2001ja000220.pdf.
*
*    Note that the 2001 Tsyganenko G2 index uses IMF and solar wind
*    speed measurements taken every 5 minutes for a hour.  Since
*    Madrigal presently only has hourly measurements, we'll just average
*    two measurements instead of 12.
*
*
*   arguments:
*      double swspd_now - solar wind speed now in m/s
*      double swspd_1hour - solar wind speed 1 hour earlier in m/s
*      double imf_zgsm_now - imf in z gsm direction in nTesla measured now
*      double imf_zgsm_1hour - imf in z gsm direction in nTesla measured 1 hour ago
*
*
*   Returns:
*      double - G1 index.  If problem, returns missing.
*/
double getTsyganenkoG2Index(double swspd_now,
                            double swspd_1hour,
                            double imf_zgsm_now,
                            double imf_zgsm_1hour)



/**************************************************************************
*
* traceMagneticField - a public method used to support the Magnetic field
*                      line trace web service.
*
*
*
*   arguments:
*      int year
*      int month
*      int day
*      int hour
*      int min
*      int sec
*      double gdlat - geodetic latitude of starting point
*      double glon - longitude of starting point
*      double gdalt - geodetic altitude of starting point
*      int model - 0 for Tsyganenko, 1 for IGRF
*      int qualifier - 0 for conjugate, 1 for north_alt, 2 for south_alt, 3 for apex, 4 for GSM
*            If qualifier == 4, model must be Tsyganenko.
*      double * stopAlt - altitude to stop trace at, if qualifier is north_alt or south_alt.
*                         If other qualifier, this parameter is ignored
*           (the following are output parameters)
*      double * end_gdlat (if qualifier == 4 (GSM XY plane), this will be XGSM instead)
*      double * end_glon (if qualifier == 4 (GSM XY plane), this will be YGSM  instead)
*      double * end_gdalt (if qualifier == 4 (GSM XY plane), this will be ZGSM = 0 instead)
*
*
*   Returns:
```

```
*      double - 0 if success, -1 if failure
*/
int traceMagneticField(int year,
                       int month,
                       int day,
                       int hour,
                       int min,
                       int sec,
                       double gdlat,
                       double glon,
                       double gdalt,
                       int model,
                       int qualifier,
                       double stopAlt,
                       double * end_gdlat,
                       double * end_glon,
                       double * end_gdalt)



/**************************************************************************
*
* run_iri - a public method used to simplify calling the iri model
*
*
*
*   input arguments: *      int year
*      int month
*      int day
*      int hour
*      int min
*      int sec
*      double gdlat - geodetic latitude of starting point
*      double glon - longitude of starting point
*      double gdalt - geodetic altitude of starting point
*      double * iri - array of 11 doubles containing returned data -
*                     allocated by user.  Contains :
*            NE_IRI     Electron density in #/meter3.    units: m-3
*            NEL_IRI    Log of electron density in #/meter3.   units:log10(m-3)
*            TN_IRI  IRI Neutral temperature
*            TI_IRI   IRI Ion temperature
*            TE_IRI   IRI Electron temperature
*            PO+_IRI   IRI Composition - [O+]/Ne
*            PNO+_IRI   IRI Composition - [NO+]/Ne
*            PO2+_IRI   IRI Composition - [O2+]/Ne
*            PHE+_IRI   IRI Composition - [HE+]/Ne
*            PH+_IRI   IRI Composition - [H+]/Ne
*            PN+_IRI   IRI Composition - [N+]/Ne
*   Returns:
*      double - 0 if success, -1 if failure
*
*   Calls IRI method iri_sub, for only one altitude.  Updated for IRI 2007 release.
*/
int run_iri(int year,
            int month,
            int day,
            int hour,
            int min,
            int sec,
            double gdlat,
            double glon,
```

```
            double gdalt,
            double * iri)


/*****************************************************************
 *
 *  faraday_rotation    calculates total phase shift due to a single pass of an
 *                      electromagnetic wave from a ground instrument to a point
 *                      in space.
 *
 *   Uses quasi-longitudinal approximation, so not appropriate if perp
 *   to magnetic field.
 *   Uses coord and IRI from geolib
 *
 *   Inputs: year month day hour min sec sgdlat slon sgdalt gdlat glon gdalt freq
 *           double * tec, double * total_tec
 *
 *   where sgdlat slon sgdalt are geodetic station location,
 *   gdlat glon gdalt are geodetic point locations, and freq is
 *   wave frequency in Hz.
 *   double * tec is a pointer to a double, which is set to total tec from
 *          station to point in electons/m^2, or missing if error.
 *   double * total_tec is a pointer to a double, which is set to total tec from
 *          station to GPS satellite height (22000 km) in electons/m^2 through line containing
 *          point, or missing if error.
 *
 *   Returns: one way faraday rotation in radians, missing if error
 */
double faraday_rotation(int year, int month, int day, int hour, int min, int sec,
                                double sgdlat, double slon, double sgdalt,
                                double gdlat, double glon, double gdalt, double freq,
                                double * tec, double * total_tec)



/*****************************************************************
*
* getByear   derives Byear from IBYR
*
*   arguments:
*      inCount (num inputs) = 1 (IBYR)
*      inputArr - double array holding:
*                 IBYR - year of beginning of record
*      outCount (num outputs) = 1 (BYEAR)
*      outputArr - double array holding:
*                 BYEAR - year of beginning of record
*
*   Algorithm: BYEAR =  IBYR directly from prolog
*
*   returns - 0 (successful)
*/
int getByear(int inCount,
             double * inputArr,
             int outCount,
             double * outputArr,
             FILE * errFile)



/*****************************************************************
```

```
*
* getTime   derives basic time parameters from all prolog time information.
*
*    All outputs set at average time between beginning
*    and end of record.
*
*   arguments:
*      inCount (num inputs) = 8 (IBYR, IBDT, IBHM, IBCS, IEYR, IEDT, IEHM, IECS)
*      inputArr - double array holding inputs from above
*      outCount (num outputs) = 7 (YEAR, MONTH, DAY, HOUR, MIN, SEC, CSEC)
*      outputArr - double array holding:
*                  YEAR - year at avg time in record
*                  MONTH - month at avg time in record
*                  DAY - day at avg time in record
*                  HOUR - hour at avg time in record
*                  MIN - min at avg time in record
*                  SEC - sec at avg time in record
*                  CSEC - centisec at avg time in record
*
*   Algorithm: Determine average time in record, determine time
*
*   returns - 0 (successful)
*/
int getTime(int inCount,
            double * inputArr,
            int outCount,
            double * outputArr,
            FILE * errFile)



/**********************************************************************
*
* getBmd   derives Bmd from IBDT
*
*   arguments:
*      inCount (num inputs) = 1 (IBDT)
*      inputArr - double array holding IBDT - mmdd of beginning of record
*      outCount (num outputs) = 1 (BMD)
*      outputArr - double array holding BMD - mmdd of beginning of record
*
*   Algorithm: BMD =  IBDT directly from prolog
*
*   returns - 0 (successful)
*/
int getBmd(int inCount,
           double * inputArr,
           int outCount,
           double * outputArr,
           FILE * errFile)



/**********************************************************************
*
* getBMonthDay   derives BMONTH and BDAY from IBDT
*
*   arguments:
*      inCount (num inputs) = 1 (IBDT)
*      inputArr - double array holding IBDT - mmdd of beginning of record
*      outCount (num outputs) = 2 (BMONTH and BDAY)
```

```
*      outputArr - double array holding BMONTH and BDAY - month and day of beginning of record
*
*   Algorithm: BMonth =  IBDT directly from prolog / 100
*              BDay = IBDT - 100*BMonth
*
*   returns - 0 (successful)
*/
int getBMonthDay(int inCount,
                 double * inputArr,
                 int outCount,
                 double * outputArr,
                 FILE * errFile)



/*****************************************************************
*
* getMd   derives mmdd from all prolog time information.
*
*    Mmdd is set as mmdd at average time between beginning
*    and end of record.
*
*   arguments:
*      inCount (num inputs) = 8 (IBYR, IBDT, IBHM, IBCS, IEYR, IEDT, IEHM, IECS)
*      inputArr - double array holding inputs from above
*      outCount (num outputs) = 1 (MD)
*      outputArr - double array holding: MD - mmdd at avg time in record
*
*   Algorithm: Determine average time in record, determine its mmdd
*
*   returns - 0 (successful)
*/
int getMd(int inCount,
          double * inputArr,
          int outCount,
          double * outputArr,
          FILE * errFile)



/*****************************************************************
*
* getUtUnix   derives ut1_unix and ut2_unix from all prolog time information.
*
*    ut1_unix is the unix timestamp (referenced to midnight UT 1/1/1970)
*    of the beginning of the record (seconds)
*    ut2_unix is the unix timestamp (referenced to midnight UT 1/1/1970)
*    of the end of the record (seconds)
*
*   arguments:
*      inCount (num inputs) = 8 (IBYR, IBDT, IBHM, IBCS, IEYR, IEDT, IEHM, IECS)
*      inputArr - double array holding inputs from above
*      outCount (num outputs) = 2 (UT1_UNIX, UT2_UNIX)
*      outputArr - double array holding: UT1_UNIX - Unix seconds (1/1/1970) at start
*                                        UT2_UNIX - Unix seconds (1/1/1970) at end
*
*   Algorithm: Determine average time in record, determine its mmdd
*
*   returns - 0 (successful)
*/
int getUtUnix(int inCount,
```

```
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)


/*****************************************************************
*
* getDayno   derives day number from all prolog time information.
*
*    dayno is set as the day number (1-366) at average time between beginning
*    and end of record.
*
*   arguments:
*      inCount (num inputs) = 8 (IBYR, IBDT, IBHM, IBCS, IEYR, IEDT, IEHM, IECS)
*      inputArr - double array holding inputs from above
*      outCount (num outputs) = 1 (DAYNO)
*      outputArr - double array holding: DAYNO - day number at avg time in record
*
*   Algorithm: Determine average time in record, determine its day number
*
*   returns - 0 (successful), -1 if error
*/
int getDayno(int inCount,
             double * inputArr,
             int outCount,
             double * outputArr,
             FILE * errFile)



/*****************************************************************
*
* getBhm   derives Bhm from IBHM
*
*   arguments:
*      inCount (num inputs) = 1 (IBHM)
*      inputArr - double array holding IBHM - hhmm of beginning of record
*      outCount (num outputs) = 1 (BHM)
*      outputArr - double array holding BHM - hhmm of beginning of record
*
*   Algorithm: BHM =  IBHM directly from prolog
*
*   returns - 0 (successful)
*/
int getBhm(int inCount,
           double * inputArr,
           int outCount,
           double * outputArr,
           FILE * errFile)



/*****************************************************************
*
* getBhhmmss   derives Bhhmmss from IBHM and IBCS
*
*   arguments:
*      inCount (num inputs) = 2 (IBHM, IBCS)
*      inputArr - double array holding IBHM - hhmm of beginning of record
```

```
*                                       IBCS - centiseconds at beginning of record
*      outCount (num outputs) = 1 (BHHMMSS)
*      outputArr - double array holding BHHMMSS - hhmmss of beginning of record
*
*   Algorithm: BHHMMSS from  IBHM*100 + IBCS/100
*
*   returns - 0 (successful)
*/
int getBhhmmss(int inCount,
               double * inputArr,
               int outCount,
               double * outputArr,
               FILE * errFile)



/**********************************************************************
*
* getEhhmmss   derives Ehhmmss from IEHM and IECS
*
*   arguments:
*      inCount (num inputs) = 2 (IEHM, IECS)
*      inputArr - double array holding IEHM - hhmm of ending of record
*                                       IECS - centiseconds at ending of record
*      outCount (num outputs) = 1 (EHHMMSS)
*      outputArr - double array holding EHHMMSS - hhmmss of ending of record
*
*   Algorithm: EHHMMSS from  IEHM*100 + IECS/100
*
*   returns - 0 (successful)
*/
int getEhhmmss(int inCount,
               double * inputArr,
               int outCount,
               double * outputArr,
               FILE * errFile)



/**********************************************************************
*
* getHm   derives hhmm from all prolog time information.
*
*    HM is set as hhmm at average time between beginning
*    and end of record.
*
*   arguments:
*      inCount (num inputs) = 8 (IBYR, IBDT, IBHM, IBCS, IEYR, IEDT, IEHM, IECS)
*      inputArr - double array holding inputs from above
*      outCount (num outputs) = 1 (HM)
*      outputArr - double array holding: HM - hhmm at avg time in record
*
*   Algorithm: Determine average time in record, determine its hhmm
*
*   returns - 0 (successful)
*/
int getHm(int inCount,
          double * inputArr,
          int outCount,
          double * outputArr,
          FILE * errFile)
```

```
/*****************************************************************
*
* getUth   derives UTH (hours since midnight UT of first day of experiment)
*          from all prolog time information.
*
*    UTH is set as the number of hours at average time between beginning
*    and end of record since midnight UT of first day of experiment.
*
*   arguments:
*      inCount (num inputs) = 12 (FIRST_IBYR, FIRST_IBDT, FIRST_IBHM, FIRST_IBCS,
*                                 IBYR, IBDT, IBHM, IBCS, IEYR, IEDT, IEHM, IECS)
*      inputArr - double array holding inputs from above
*      outCount (num outputs) = 1 (UTH)
*      outputArr - double array holding: UTH - number of hours at avg time in record
*                  since midnight UT of first day of experiment.
*
*   Algorithm: Determine average time in record, determine when midnight of first day
*              was using FIRST_*, get number of hours since then
*
*   returns - 0 (successful)
*/
int getUth(int inCount,
           double * inputArr,
           int outCount,
           double * outputArr,
           FILE * errFile)



/*****************************************************************
*
* getUts   derives UTS (seconds since midnight UT of first day of experiment)
*          from all prolog time information.
*
*    UTS is set as the number of seconds at average time between beginning
*    and end of record since midnight UT of first day of experiment.
*
*   arguments:
*      inCount (num inputs) = 12 (FIRST_IBYR, FIRST_IBDT, FIRST_IBHM, FIRST_IBCS,
*                                 IBYR, IBDT, IBHM, IBCS, IEYR, IEDT, IEHM, IECS)
*      inputArr - double array holding inputs from above
*      outCount (num outputs) = 1 (UTS)
*      outputArr - double array holding: UTS - number of seconds at avg time in record
*                  since midnight UT of first day of experiment.
*
*   Algorithm: Determine average time in record, determine when midnight of first day
*              was using FIRST_*, get number of seconds since then
*
*   returns - 0 (successful)
*/
int getUts(int inCount,
           double * inputArr,
           int outCount,
           double * outputArr,
           FILE * errFile)
```

```
/**************************************************************************
*
* getBUth   derives B_UTH (hours until start time since midnight UT of first day of experiment)
*           from all prolog time information.
*
*    B_UTH is set as the number of hours at beginning time
*    of record since midnight UT of first day of experiment.
*
*   arguments:
*      inCount (num inputs) = 8 (FIRST_IBYR, FIRST_IBDT, FIRST_IBHM, FIRST_IBCS,
*                                IBYR, IBDT, IBHM, IBCS)
*      inputArr - double array holding inputs from above
*      outCount (num outputs) = 1 (B_UTH)
*      outputArr - double array holding: B_UTH - number of hours at record start time
*                  since midnight UT of first day of experiment.
*
*   Algorithm: Get start time in record, determine when midnight of first day
*              was using FIRST_*, get number of hours since then
*
*   returns - 0 (successful)
*/
int getBUth(int inCount,
            double * inputArr,
            int outCount,
            double * outputArr,
            FILE * errFile)



/**************************************************************************
*
* getInttms   derives integration time in seconds from all prolog time information.
*
*    INTTMS is set as the number of seconds between beginning
*    and end of record.
*
*   arguments:
*      inCount (num inputs) = 8 (IBYR, IBDT, IBHM, IBCS, IEYR, IEDT, IEHM, IECS)
*      inputArr - double array holding inputs from above
*      outCount (num outputs) = 1 (INTTMS)
*      outputArr - double array holding: INTTMS - number of seconds between beginning
*                                        and end of record
*
*   Algorithm: Determine beg and end time in record, determine difference in seconds
*
*   returns - 0 (successful)
*/
int getInttms(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/**************************************************************************
*
* getInttmm   derives integration time in minutes from all prolog time information.
*
*    INTTMM is set as the number of minutes between beginning
*    and end of record.
```

```
*
*   arguments:
*      inCount (num inputs) = 8 (IBYR, IBDT, IBHM, IBCS, IEYR, IEDT, IEHM, IECS)
*      inputArr - double array holding inputs from above
*      outCount (num outputs) = 1 (INTTMM)
*      outputArr - double array holding: INTTMM - number of minutes between beginning
*                                        and end of record
*
*   Algorithm: Determine beg and end time in record, determine difference in minutes
*
*   returns - 0 (successful)
*/
int getInttmm(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/**********************************************************************
*
* getDatntd   derives integration time in days from all prolog time information.
*
*    INTTMM is set as the number of days between beginning
*    and end of record.
*
*   arguments:
*      inCount (num inputs) = 8 (IBYR, IBDT, IBHM, IBCS, IEYR, IEDT, IEHM, IECS)
*      inputArr - double array holding inputs from above
*      outCount (num outputs) = 1 (DATNTD)
*      outputArr - double array holding: DATNTD - number of days between beginning
*                                        and end of record
*
*   Algorithm: Determine beg and end time in record, determine difference in days
*
*   returns - 0 (successful)
*/
int getDatntd(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/**********************************************************************
*
* getUt   derives UT  (Hours since midnight of exp beg, MOD 24).
*
*
*   arguments:
*      inCount (num inputs) = 1 (UTH)
*      inputArr - double array holding inputs from above
*      outCount (num outputs) = 1 (UT)
*      outputArr - double array holding: UT - Hours since midnight
*                                        of exp beg, MOD 24
*
*   Algorithm: UT = fmod(UTH, 24.0)
*
*   returns - 0 (successful)
```

```
*/
int getUt(int inCount,
          double * inputArr,
          int outCount,
          double * outputArr,
          FILE * errFile)



/*****************************************************************
*
* getBegUt   derives Beg_UT  (Hours since midnight of exp beg, MOD 24, using start time).
*
*
*   arguments:
*      inCount (num inputs) = 1 (B_UTH)
*      inputArr - double array holding inputs from above
*      outCount (num outputs) = 1 (BEG_UT)
*      outputArr - double array holding: BEG_UT - Hours since midnight
*                                        of exp beg, MOD 24, using start time
*
*   Algorithm: BEG_UT = fmod(B_UTH, 24.0)
*
*   returns - 0 (successful)
*/
int getBegUt(int inCount,
             double * inputArr,
             int outCount,
             double * outputArr,
             FILE * errFile)



/*****************************************************************
*
* getJdayno   derives Julian day number from all prolog time information.
*
*    Jdayno is set as day number at average time between beginning
*    and end of record.
*
*   arguments:
*      inCount (num inputs) = 8 (IBYR, IBDT, IBHM, IBCS, IEYR, IEDT, IEHM, IECS)
*      inputArr - double array holding inputs from above
*      outCount (num outputs) = 1 (JDAYNO)
*      outputArr - double array holding: JDAYNO (Julian day number)
*
*   Algorithm: Determine average time in record, determine its Julian Day number
*              via date method jday.
*
*   returns - 0 (successful), -1 if error
*/
int getJdayno(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/*****************************************************************
*
```

```
* getJulian_date   derives Julian day wich is a float number from all prolog time information.
*
*    Julian_date is set as Julian date at average time between beginning
*    and end of record.
*
*   arguments:
*      inCount (num inputs) = 9 (IBYR, IBDT, IBHM, IBCS, IEYR, IEDT, IEHM, IECS, JDAYNO)
*      inputArr - double array holding inputs from above
*      outCount (num outputs) = 1 (Julian_date)
*      outputArr - double array holding: Julian_date (is a float)
*
*
*
*   returns - 0 (successful), -1 if error
*/
int getJulian_date(int inCount,
             double * inputArr,
             int outCount,
             double * outputArr,
             FILE * errFile)



/**************************************************************************
*
* getUt1   derives UT1 from prolog start time.
*
*    UT1 is the number of seconds from 1/1/1950 to record start time.
*
*   arguments:
*      inCount (num inputs) = 4 (IBYR, IBDT, IBHM, IBCS)
*      inputArr - double array holding inputs from above
*      outCount (num outputs) = 1 (UT1)
*      outputArr - double array holding: UT1
*
*   Algorithm: Determine start time in record via dmadptr
*
*   returns - 0 (successful)
*/
int getUt1(int inCount,
           double * inputArr,
           int outCount,
           double * outputArr,
           FILE * errFile)



/**************************************************************************
*
* getUt2   derives UT2 from prolog end time.
*
*    UT2 is the number of seconds from 1/1/1950 to record end time.
*
*   arguments:
*      inCount (num inputs) = 4 (IEYR, IEDT, IEHM, IECS)
*      inputArr - double array holding inputs from above
*      outCount (num outputs) = 1 (UT2)
*      outputArr - double array holding: UT2
*
*   Algorithm: Determine end time in record via dmadptr
*
```

```
*   returns - 0 (successful)
*/
int getUt2(int inCount,
           double * inputArr,
           int outCount,
           double * outputArr,
           FILE * errFile)



/*****************************************************************
*
* getDut21   derives Integration time in secs from UT2 - UT1.
*
*
*   arguments:
*      inCount (num inputs) = 2 (UT1, UT2)
*      inputArr - double array holding inputs from above
*      outCount (num outputs) = 1 (DUT21)
*      outputArr - double array holding: DUT21 (record/integration
*                  time in seconds).
*
*   Algorithm: DUT21 = UT2 - UT1
*
*   returns - 0 (successful)
*/
int getDut21(int inCount,
             double * inputArr,
             int outCount,
             double * outputArr,
             FILE * errFile)



/*****************************************************************
*
* getFyear   derives average time as float year from all prolog time information.
*
*    Fyear is set at average time between beginning
*    and end of record in units of years.
*
*   arguments:
*      inCount (num inputs) = 8 (IBYR, IBDT, IBHM, IBCS, IEYR, IEDT, IEHM, IECS)
*      inputArr - double array holding inputs from above
*      outCount (num outputs) = 1 (FYEAR)
*      outputArr - double array holding: FYEAR - ave time in years
*
*   Algorithm: Determine average time in record, determine its fyear
*
*   returns - 0 (successful)
*/
int getFyear(int inCount,
             double * inputArr,
             int outCount,
             double * outputArr,
             FILE * errFile)



/*****************************************************************
*
```

```
* getStation   gets instrument lat, lon, and alt given kinst.
*
*
*   arguments:
*      inCount (num inputs) = 1 (KINST)
*      inputArr - double array holding inputs from above
*      outCount (num outputs) = 3 (GDLATR, GDLONR, GALTR)
*      outputArr - double array holding:
*                          GDLATR - Inst geod latitude (N hemi=pos) - deg
*                          GDLONR - Inst geod longitute - deg
*                          GALTR  - Inst altitute above sea level (km)
*
*   Algorithm: Gets all data from instTab.txt via cedarGetStationPos
*
*   returns - 0 (successful)
*/
int getStation(int inCount,
               double * inputArr,
               int outCount,
               double * outputArr,
               FILE * errFile)



/*****************************************************************
*
* getAltInc   derives GDALT as  ALTB + (ROW*ALTAV)
*
*   arguments:
*      inCount (num inputs) = 3 (ROW, ATLB, ALTAV)
*      inputArr - double array holding:
*                 ROW - 2D row number (start at 0)
*                 ALTB - starting altitude
*                 ALTAV - altitude increment per row
*      outCount (num outputs) = 1 (GDALT in km)
*      outputArr - double array holding:
*                 GDALT - Geodetic altitude in km
*
*   Algorithm: GDALT = ALTB + (ROW*ALTAV)
*
*   returns - 0 (successful)
*/
int getAltInc(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/*****************************************************************
*
* getAveAlt   derives GDALT as average of ALTB and ALTE
*
*   arguments:
*      inCount (num inputs) = 2 (ATLB, ALTE)
*      inputArr - double array holding:
*                 ALTB - starting altitude
*                 ALTE - ending altitude
*      outCount (num outputs) = 1 (GDALT in km)
*      outputArr - double array holding:
```

```
*                GDALT - Geodetic altitude in km
*
*   Algorithm: GDALT = (ATLB + ALTE)/2
*
*   returns - 0 (successful)
*/
int getAveAlt(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/**************************************************************************
*
* getAveDAlt   derives dGDALT given dALTB and dALTE
*
*   arguments:
*      inCount (num inputs) = 2 (DATLB, DALTE)
*      inputArr - double array holding:
*                 DALTB - error in starting altitude (km)
*                 DALTE - error in sending altitude km)
*      outCount (num outputs) = 1 (DGDALT in km)
*      outputArr - double array holding:
*                 DGDALT - error in Geodetic altitude in km
*
*   Algorithm: dGDALT = 1/2(DATLB^2 + DALTE^2)^1/2
*
*   returns - 0 (successful)
*/
int getAveDAlt(int inCount,
               double * inputArr,
               int outCount,
               double * outputArr,
               FILE * errFile)



/**************************************************************************
*
* getResl   derives resl from mresl
*
*   A trivial method that equates the Millstone range resolution with
*   the standard range resolution
*
*   arguments:
*      inCount (num inputs) = 1 (MRESL)
*      inputArr - double array holding:
*                 MRESL - Millstone specific range resolution in km
*      outCount (num outputs) = 1 (RESL in km)
*      outputArr - double array holding:
*                 RESL - range resolution in km
*
*   Algorithm: RESL = MRESL
*
*   returns - 0 (successful)
*/
int getResl(int inCount,
            double * inputArr,
            int outCount,
```

```
            double * outputArr,
            FILE * errFile)


/*****************************************************************
*
* getAzmDaz   derives AZM and DAZ from AZ1 and AZ2
*
*   arguments:
*      inCount (num inputs) = 2 (AZ1, AZ2)
*      inputArr - double array holding:
*                 AZ1 - starting azimuth
*                 AZ2 - ending azimuth
*      outCount (num outputs) = 2 (AZM and DAZ)
*      outputArr - double array holding:
*                 AZM - median azimuth from -180 to 180 deg
*                 DAZ - Degrees of rotation from AZ1 to AZ2
*
*   Algorithm: The assumption is made that a clockwise motion
*   of the antenna always increases AZ, and counterclockwise
*   decreases AZ.  In case this convention is not followed,
*   a warning is printed to errFile whenever abs(DAZ) is greater
*   than 180 degrees.
*
*   This warning surpressed per request SRI on 1/15/2004
*
*   returns - 0 (successful)
*/
int getAzmDaz(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/*****************************************************************
*
* getDAzmDDaz   derives DAZM and DDAZ from DAZ1 and DAZ2
*
*   arguments:
*      inCount (num inputs) = 2 (DAZ1, DAZ2)
*      inputArr - double array holding:
*                 DAZ1 - error in starting azimuth
*                 DAZ2 - error in ending azimuth
*      outCount (num outputs) = 2 (DAZM and DDAZ)
*      outputArr - double array holding:
*                 AZM - error in median azimuth from -180 to 180 deg
*                 DAZ - error in Degrees of rotation from AZ1 to AZ2
*
*   Algorithm: dAZM = 1/2(DDAZ1^2 + DDAZ2^2)^1/2
*              dDAZ = (DDAZ1^2 + DDAZ2^2)^1/2
*
*   returns - 0 (successful)
*/
int getDAzmDDaz(int inCount,
                double * inputArr,
                int outCount,
                double * outputArr,
                FILE * errFile)
```

```
/*****************************************************************
*
* getElmDel   derives ELM and DEL from EL1 and EL2
*
*   arguments:
*      inCount (num inputs) = 2 (EL1, EL2)
*      inputArr - double array holding:
*                 EL1 - starting elevation in deg
*                 EL2 - ending elevation in deg
*      outCount (num outputs) = 2 (ELM, DEL)
*      outputArr - double array holding:
*                 ELM - median elevation in deg
*                 DEL - degrees of rotation between EL1 and EL2
*
*   Algorithm: ELM = EL1+EL2 / 2; DEL = EL2 - EL1
*
*   returns - 0 (successful)
*/
int getElmDel(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/*****************************************************************
*
* getDElmDDel   derives DELM and DDEL from DEL1 and DEL2
*
*   arguments:
*      inCount (num inputs) = 2 (DEL1, DEL2)
*      inputArr - double array holding:
*                 DEL1 - error in starting elevation
*                 DEL2 - error in ending elevation
*      outCount (num outputs) = 2 (DELM and DDEL)
*      outputArr - double array holding:
*                 ELM - error in median elevation
*                 DEL - error in Degrees of rotation from EL1 to EL2
*
*   Algorithm: dELM = 1/2(DDEL1^2 + DDEL2^2)^1/2
*              dDEL = (DDEL1^2 + DDEL2^2)^1/2
*
*   returns - 0 (successful)
*/
int getDElmDDel(int inCount,
                double * inputArr,
                int outCount,
                double * outputArr,
                FILE * errFile)



/*****************************************************************
*
* getGeod   derives Geodetic lat, long and alt from kinst, azm, elm, and range
*
*   arguments:
```

```
*      inCount (num inputs) = 4 (KINST, AZM, ELM, RANGE)
*      inputArr - double array holding:
*                 KINST - instrument id
*                 AZM - mean azimuth in deg
*                 ELM - mean elevation in deg
*                 RANGE - range in km
*      outCount (num outputs) = 3 (GDLAT, GLON, GDALT)
*      outputArr - double array holding:
*                 GDLAT - geodetic latitude of measurement
*                 GLON - geodetic longitude of measurement
*                 GDALT - geodetic altitude of measurement
*
*   Algorithm: Calls los2geodetic from geometry.c
*
*   returns - 0 (successful)
*/
int getGeod(int inCount,
            double * inputArr,
            int outCount,
            double * outputArr,
            FILE * errFile)



/*****************************************************************
*
* getDGeod   derives error in Geodetic lat, long and alt
*            from azm, dazm, elm, delm, range, and drange
*
*   arguments:
*      inCount (num inputs) = 7 (KINST, AZM, DAZM, ELM, DELM, RANGE, DRANGE)
*      inputArr - double array holding:
*                 KINST - instrument id
*                 AZM - mean azimuth in deg
*                 DAZM - error in mean azimuth
*                 ELM - mean elevation in deg
*                 DELM - error in mean elevation
*                 RANGE - range in km
*                 DRANGE - error in range
*      outCount (num outputs) = 3 (DGDLAT, DGLON, DGDALT)
*      outputArr - double array holding:
*                 DGDLAT - error in geodetic latitude
*                 DGLON - error in geodetic longitude
*                 DGDALT - error in geodetic altitude
*
*   Algorithm: Calculate derivatives with respect to AZM, ELM, and
*              RANGE by simply finding the difference with them
*              incremented by 1 degree or 1 km.  These derivitives
*              are:
*                 dlatdaz, dlatdel, dlatdrange
*                 dlondaz, dlondel, dlondrange
*                 daltdaz, daltdel, daltdrange
*
*              then:
*
*     dgdlat = ((dlatdaz*dazm)^2 + (dlatdel*delm)^2 + (dlatdrange*drange)^2)^1/2
*     dgdlon = ((dlondaz*dazm)^2 + (dlondel*delm)^2 + (dlondrange*drange)^2)^1/2
*     dgdalt = ((daltdaz*dazm)^2 + (daltdel*delm)^2 + (daltdrange*drange)^2)^1/2
*
*   returns - 0 (successful)
*/
```

```
int getDGeod(int inCount,
             double * inputArr,
             int outCount,
             double * outputArr,
             FILE * errFile)



/*****************************************************************
*
* getGeodGdalt   derives Geodetic lat, long and alt from kinst, azm, elm, and gdalt
*
*   arguments:
*      inCount (num inputs) = 4 (KINST, AZM, ELM, GDALT)
*      inputArr - double array holding:
*                 KINST - instrument id
*                 AZM - mean azimuth in deg
*                 ELM - mean elevation in deg
*                 GDALT - alt in km
*      outCount (num outputs) = 3 (GDLAT, GLON)
*      outputArr - double array holding:
*                 GDLAT - geodetic latitude of measurement
*                 GLON - geodetic longitude of measurement
*
*   Algorithm: Calls los2geodetic from geometry.c after calculating range
*
*   returns - 0 (successful)
*/
int getGeodGdalt(int inCount,
                 double * inputArr,
                 int outCount,
                 double * outputArr,
                 FILE * errFile)



/*****************************************************************
*
* getGeodAlt   derives Geodetic lat, long  by assuming measured point is
*              directly above instrument
*
*    This method will be called only if GDLAT, GLON cannot be derived
*    any other way
*
*   arguments:
*      inCount (num inputs) = 2 (GDLATR, GDLONR)
*      inputArr - double array holding:
*                 GDLATR - Inst geod latitude (N hemi=pos) - deg
*                 GDLONR - Inst geod longitute - deg
*      outCount (num outputs) = 2 (GDLAT, GLON)
*      outputArr - double array holding:
*                 GDLAT - geodetic latitude of measurement
*                 GLON - longitude of measurement
*
*   Algorithm: Sets GDLAT, GLON to station values
*
*   returns - 0 (successful)
*/
int getGeodAlt(int inCount,
               double * inputArr,
               int outCount,
```

```
                double * outputArr,
                FILE * errFile)



/**************************************************************************
*
* getAzElRange   derives Azm, Elm, and Range given gdlat,glon,gdalt
*                (point position) and gdlatr,glonr,galtr (station position)
*
*    This method will be called only if Azm, Elm, and Range cannot be derived
*    any other way
*
*   arguments:
*      inCount (num inputs) = 6 (GDLAT, GLON, GDALT, GDLATR, GDLONR, GALTR)
*      inputArr - double array holding:
*                 GDLAT - geodetic latitude of measurement
*                 GLON - longitude of measurement
*                 GDALT - geodetic altitude of measurement in km
*                 GDLATR - Inst geod latitude (N hemi=pos) - deg
*                 GDLONR - Inst geod longitute - deg
*                 GALTR - geodetic altitude of station in km
*      outCount (num outputs) = 3 (AZM, ELM, RANGE)
*      outputArr - double array holding:
*                 AZM - mean azimuth in deg
*                 ELM - mean elevation in deg
*                 RANGE - range in km
*
*   Algorithm: See look in geometry.c
*
*   returns - 0 (successful)
*/
int getAzElRange(int inCount,
                 double * inputArr,
                 int outCount,
                 double * outputArr,
                 FILE * errFile)



/**************************************************************************
*
* getSZen   derives Solar zenith angle in deg (0 = overhead)
*
*   arguments:
*      inCount (num inputs) = 4 (UT1, UT2, GDLAT, GLON)
*      inputArr - double array holding:
*                 UT1 - UT at record start
*                 UT2 - UT at record end
*                 GDLAT - geodetic latitude
*                 GLON - geodetic longitude
*      outCount (num outputs) = 1 (SZEN)
*      outputArr - double array holding:
*                 SZEN - Solar zenith angle in deg (0 = overhead)
*
*   Algorithm: See solarzen in geometry.c
*
*   returns - 0 (successful)
*/
int getSZen(int inCount,
            double * inputArr,
```

```
            int outCount,
            double * outputArr,
            FILE * errFile)


/*****************************************************************
*
* getSltmut   derives SLTMUT (local solar time diff)
*
*   arguments:
*      inCount (num inputs) = 3 (UT1, UT2, GLON)
*      inputArr - double array holding:
*                  UT1 - UT at record start
*                  UT2 - UT at record end
*                  GLON - geodetic longitude
*      outCount (num outputs) = 1 (SLTMUT)
*      outputArr - double array holding:
*                  SLTMUT - Local solar time diff (=SLT-UT) +E lon in hhmm
*
*   Algorithm: Add 3600*24*GLON/360 to average UT - convert to hhmm, where
*              GLON goes from -180 to 180
*
*   returns - 0 (successful)
*/
int getSltmut(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/*****************************************************************
*
* getSlt   derives SLT (local solar time  in hours)
*
*   arguments:
*      inCount (num inputs) = 3 (UT1, UT2, GLON)
*      inputArr - double array holding:
*                  UT1 - UT at record start
*                  UT2 - UT at record end
*                  GLON - geodetic longitude
*      outCount (num outputs) = 1 (SLT)
*      outputArr - double array holding:
*                  SLT - Local solar time in hours (0-24)
*
*   Algorithm: Add 3600*24*GLON/360 to average UT - convert to hours, where
*              GLON goes from -180 to 180
*
*   returns - 0 (successful)
*/
int getSlt(int inCount,
           double * inputArr,
           int outCount,
           double * outputArr,
           FILE * errFile)



/*****************************************************************
```

```
*
* getSdwHt   derives shadowheight - the distance directly above any gdlat and glon
*              for a given UT in km at which the earth's shadow terminates.
*              Will be 0.0 on dayside of earth.
*
*   arguments:
*      inCount (num inputs) = 4 (UT1, UT2, GDLAT, GLON)
*      inputArr - double array holding:
*                 UT1 - UT at record start
*                 UT2 - UT at record end
*                 GDLAT - geodetic latitude
*                 GLON - geodetic longitude
*      outCount (num outputs) = 1 (SDWHT)
*      outputArr - double array holding:
*                 SDWHT - the distance in km above the given gdlat and glon
*                         at which any part of the sun is visible (may be 0.0)
*
*   Algorithm: See shadowheight in geometry.c
*
*   returns - 0 (successful)
*/
int getSdwHt(int inCount,
             double * inputArr,
             int outCount,
             double * outputArr,
             FILE * errFile)



/*****************************************************************
*
* getSuntime  derives sunset and sunrise for given point in space and time
*              for that day UT.
*
*   arguments:
*      inCount (num inputs) = 5 (UT1, UT2, GDLAT, GLON, GDALT)
*      inputArr - double array holding:
*                 UT1 - UT at record start
*                 UT2 - UT at record end
*                 GDLAT - geodetic latitude
*                 GLON - geodetic longitude
*                 GDALT - geodetic altitude in km
*      outCount (num outputs) = 4 (SUNRISE, SUNSET, SUNRISE_HOUR, SUNSET_HOUR)
*      outputArr - double array holding:
*                 SUNRISE - time (UT) that day of sunrise at that point in space
*                 SUNSET - time (UT) that day of sunset at that point in space
*                 SUNRISE_HOUR - time (in hours UT) that day of sunrise at that point in space
*                 SUNSET_HOUR - time (in hours UT) that day of sunset at that point in space
*
*   Algorithm: See sunrise_set in geometry.c
*
*   returns - 0 (successful)
*/
int getSuntime(int inCount,
               double * inputArr,
               int outCount,
               double * outputArr,
               FILE * errFile)
```

```
/*****************************************************************
*
* getTecGdalt  returns the altitude associated with a TEC measurement (350 km).
*
*   arguments:
*      inCount (num inputs) = 3 (TEC, GDLAT, GLON)
*      inputArr - double array holding:
*                 TEC - Total electron content
*                 GDLAT - geodetic latitude
*                 GLON - geodetic longitude
*      outCount (num outputs) = 1 (GDALT)
*      outputArr - double array holding:
*                 GDALT - geodetic altitude in km
*
*   Algorithm: Hard-coded return of 350.0 km
*
*   returns - 0 (successful)
*/
int getTecGdalt(int inCount,
                double * inputArr,
                int outCount,
                double * outputArr,
                FILE * errFile)



/*****************************************************************
*
* getGcdist  returns the great circle distance from the instrument to a point in space.
*
*   arguments:
*      inCount (num inputs) = 4 (GDLATR, GDLONR, GDLAT, GLON)
*      inputArr - double array holding:
*                 GDLATR - radar geodetic latitude
*                 GDLONR - radar geodetic longitude
*                 GDLAT - geodetic latitude
*                 GLON - geodetic longitude
*      outCount (num outputs) = 1 (GCDIST)
*      outputArr - double array holding:
*                 GCDIST - great circle distance from the instrument to a point in space
*                          relected onto the earth's surface (assumes spherical earth) in km.
*
*   Algorithm: See http://www.faqs.org/faqs/geography/infosystems-faq/ for great circle calcula
*
*   returns - 0 (successful)
*/
int getGcdist(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/*****************************************************************
*
* getMag   derives magnetic parameters given a point in space and time.
*
*   arguments:
*      inCount (num inputs) = 5 (UT1, UT2, GDLAT, GLON, GDALT)
*      inputArr - double array holding:
```

```
*                UT1 - UT at record start
*                UT2 - UT at record end
*                GDLAT - geodetic latitude
*                GLON - geodetic longitude
*                GDALT - geodetic altitude
*      outCount (num outputs) = 16
*      outputArr - double array holding:
*                BN - Northward comp of geomag field (1E-8 T)
*                BE - Eastward comp of geomag field (1E-8 T)
*                BD - Downward comp of geomag field (1E-8 T)
*                BMAG - geomag field strength (1E-8 T)
*                BDEC - Geomag Field East Declination (deg)
*                BINC - Geomag Field Downward Inclination (deg)
*                LSHELL - L value in measurement volume
*                DIPLAT - Dip latitude in measurement volume (deg)
*                INVLAT - Invariant latitude in measurement volume (deg)
*                APLAT - Apex latitude (deg)
*                APLON - Apex longitude (deg)
*                MAGCONJLAT - Magnetic conjugate geodetic latitude (deg)
*                MAGCONJLON - Magnetic conjugate longitude (deg)
*                CXR - MBperp. Direction Cosine (South [Apex]) m/s
*                CYR - MBperp. Direction Cosine (East [Apex]) m/s
*                CZR - MBperp. Direction Cosine (Up field line [Apex]) m/s
*
*   Algorithm: See coord in coord.f
*
*   returns - 0 (successful)
*/
int getMag(int inCount,
           double * inputArr,
           int outCount,
           double * outputArr,
           FILE * errFile)



/**********************************************************************
*
* getGeocgm   derives magnetic parameters given a point in space and time using CGM code.
*
*    Uses the GEO-CGM code available at http://nssdc.gsfc.nasa.gov/space/cgm/cgm.html
*
*    Since slower that coord.f, used only to find cgm_lat, cgm_long, and mlt
*
*   arguments:
*      inCount (num inputs) = 5 (UT1, UT2, GDLAT, GLON, GDALT)
*      inputArr - double array holding:
*                UT1 - UT at record start
*                UT2 - UT at record end
*                GDLAT - geodetic latitude
*                GLON - geodetic longitude
*                GDALT - geodetic altitude
*      outCount (num outputs) = 2
*      outputArr - double array holding:
*                CGM_LAT - Corrected geomagnetic latitude
*                CGM_LONG - Corrected geomagnetic longitude
*
*   Algorithm: See geocgm01 in geo-cgm.f
*
*   returns - 0 (successful)
*/
```

```
int getGeocgm(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/**********************************************************************
*
* getTsygan   derives field line crossing points using Tsyganenko model.
*
*    Finds the crossing point of any given field line determined by
*    the input time and point of either the dipole equatorial plane (that
*    is, the XY plane of the GSM coordinate system), or of the equatorial plane (that
*    is, the XY plane of the GSE coordinate system).  Returns the crossing point
*    of the GSM XY plane as TSYG_EQ_XGSM and TSYG_EQ_YGSM (ZGSM is by definition zero),
*    and of the GSE XY plane as TSYG_EQ_XGSE and TSYG_EQ_YGSE (ZGSE is by definition zero).
*
*   arguments:
*      inCount (num inputs) = 5 (UT1, UT2, GDLAT, GLON, GDALT)
*      inputArr - double array holding:
*                 UT1 - UT at record start
*                 UT2 - UT at record end
*                 GDLAT - geodetic latitude
*                 GLON - geodetic longitude
*                 GDALT - geodetic altitude
*      outCount (num outputs) = 4
*      outputArr - double array holding:
*                 TSYG_EQ_XGSM - X GSM value where field line crosses GSM XY plane
*                 TSYG_EQ_YGSM - Y GSM value where field line crosses GSM XY plane
*                 TSYG_EQ_XGSE - X GSE value where field line crosses GSE XY plane
*                 TSYG_EQ_YGSE - Y GSE value where field line crosses GSE XY plane
*
*   Algorithm: See Geopack_2003.f, T01_01.f
*
*   returns - 0 (successful)
*/
int getTsygan(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/**********************************************************************
*
* getAacgm   derives AACGM (adjusted altitude corrected geomagnetic) or PACE coordinates.
*
*    Finds AACGM (adjusted altitude corrected geomagnetic) or PACE lat and lon
*    given geodetic lat and lon.
*
*   arguments:
*      inCount (num inputs) = 3 (GDLAT, GLON, GDALT)
*      inputArr - double array holding:
*                 GDLAT - geodetic latitude
*                 GLON - geodetic longitude
*                 GDALT - geodetic altitude
*      outCount (num outputs) = 2
*      outputArr - double array holding:
```

```
*                AACGM (adjusted altitude corrected geomagnetic) or PACE latitude
*                AACGM (adjusted altitude corrected geomagnetic) or PACE longitude
*
*   Algorithm: See sfc_convert_geo_coord.f
*
*   returns - 0 (successful)
*/
int getAacgm(int inCount,
             double * inputArr,
             int outCount,
             double * outputArr,
             FILE * errFile)



/*********************************************************************
*
* fromAacgm   derives geodetic from AACGM (adjusted altitude corrected geomagnetic) or PACE coo
*
*    Finds geodetic lat and lon given AACGM (adjusted altitude corrected geomagnetic) or PACE l
*
*   arguments:
*      inCount (num inputs) = 3 (PACLAT, PACLON, GDALT)
*      inputArr - double array holding:
*                 PACLAT - AACGM or PACE latitude
*                 PACLON - AACGM or PACE longitude
*                 GDALT - geodetic altitude
*      outCount (num outputs) = 2
*      outputArr - double array holding:
*                 GDLAT - geodetic latitude
*                 GLON - geodetic longitude
*
*   Algorithm: See sfc_convert_geo_coord.f
*
*   returns - 0 (successful)
*/
int fromAacgm(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/*********************************************************************
*
* getMlt   derives AACGM magnetic local time.
*
*    Finds AACGM MLT given time and PACLAT.
*
*   arguments:
*      inCount (num inputs) = 3 (UT1, UT2, PACLAT)
*      inputArr - double array holding:
*                 UT1 - UT at record start
*                 UT2 - UT at record end
*                 PACLAT - PACE magnetic latitude of meas volume
*      outCount (num outputs) = 1
*      outputArr - double array holding:
*                 MLT (Magnetic local time)
*
*   Algorithm: See mlt.f
```

```
*
*   returns - 0 (successful)
*/
int getMlt(int inCount,
           double * inputArr,
           int outCount,
           double * outputArr,
           FILE * errFile)



/*****************************************************************
*
* getEregion   derives parameters associated with field lines intercepting the E region.
*
*    Finds 6 parameters related to where the magnetic field line associated with a given
*      input point intersects the E region.
*
*   inCount (num inputs) = 5 (UT1, UT2, GDLAT, GLON, GDALT)
*      inputArr - double array holding:
*                 UT1 - UT at record start
*                 UT2 - UT at record end
*                 GDLAT - geodetic latitude
*                 GLON - geodetic longitude
*                 GDALT - geodetic altitude
*      outCount (num outputs) = 6
*      outputArr - double array holding:
*                      E_REG_S_LAT - The latitude of the southern point where the magnetic
*                                    field line defined by the input point hits the  E region (
*                      E_REG_S_LON - The longitude of the southern point where the magnetic
*                                    field line defined by the input point hits the E region (1
*                      E_REG_S_SDWHT - The shadow height of the southern point where the magnet
*                                      line defined by the input point hits the E region (100 k
*                                      Shadow height is the altitude of the lowest point on the
*                                      constant geodetic lat and lon in sunlight.
*                      E_REG_N_LAT - The latitude of the northern point where the magnetic
*                                    field line defined by the input point hits the  E region (
*                      E_REG_N_LON - The longitude of the northern point where the magnetic
*                                    field line defined by the input point hits the E region (1
*                      E_REG_N_SDWHT - The shadow height of the northern point where the magnet
*                                      line defined by the input point hits the E region (100 k
*                                      Shadow height is the altitude of the lowest point on the
*                                      constant geodetic lat and lon in sunlight.
*
*
*   Algorithm: Uses traceMagneticField
*
*   returns - 0 (successful)
*/
int getEregion(int inCount,
               double * inputArr,
               int outCount,
               double * outputArr,
               FILE * errFile)



/*****************************************************************
*
* getAspect   derives magnetic aspect angle of radar beam and magnetic field
*
```

```
*   inCount (num inputs) = 8 (UT1, UT2, GDLATR, GDLONR, GALTR,
                              AZM, ELM, RANGE)
*      inputArr - double array holding:
*                 UT1 - UT at record start
*                 UT2 - UT at record end
*                 GDLATR - radar geodetic latitude
*                 GDLONR - radar geodetic longitude
*                 GALTR - radar geodetic altitude
*                 AZM - median azimuth
*                 ELM - median elevation
*                 RANGE - range to point in km
*      outCount (num outputs) = 1
*      outputArr - double array holding:
*                       ASPECT - Magnetic aspect angle
*
*   returns - 0 (successful)
*/
int getAspect(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/**************************************************************************
*
* getSltc   derives SLTC (local solar time at magnetic conjugate point in hours)
*
*   arguments:
*      inCount (num inputs) = 3 (UT1, UT2, MAGCONJLON)
*      inputArr - double array holding:
*                 UT1 - UT at record start
*                 UT2 - UT at record end
*                 MAGCONJLON - magnetic conjugate longitude
*      outCount (num outputs) = 1 (SLTC)
*      outputArr - double array holding:
*                 SLTC - Local solar time at magnetic conjugate in hours (0-24)
*
*   Algorithm: Add 3600*24*MAGCONJLON/360 to average UT - convert to hours, where
*              MAGCONJLON goes from -180 to 180
*
*   returns - 0 (successful)
*/
int getSltc(int inCount,
            double * inputArr,
            int outCount,
            double * outputArr,
            FILE * errFile)



/**************************************************************************
*
* getAplt   derives APLT (local solar time at magnetic apex point in hours)
*
*   arguments:
*      inCount (num inputs) = 3 (UT1, UT2, APLON)
*      inputArr - double array holding:
*                 UT1 - UT at record start
*                 UT2 - UT at record end
```

```
*                 APLON - magnetic apex longitude
*      outCount (num outputs) = 1 (APLT)
*      outputArr - double array holding:
*                 APLT - Local solar time at magnetic apex in hours (0-24)
*
*   Algorithm: Add 3600*24*APLON/360 to average UT - convert to hours, where
*              MAGCONJLON goes from -180 to 180
*
*   returns - 0 (successful)
*/
int getAplt(int inCount,
            double * inputArr,
            int outCount,
            double * outputArr,
            FILE * errFile)



/*********************************************************************
*
* getSZenc   derives Solar zenith angle at magnetic conjugate in deg (0 = overhead)
*
*   arguments:
*      inCount (num inputs) = 4 (UT1, UT2, MAGCONJLAT, MAGCONJLON)
*      inputArr - double array holding:
*                 UT1 - UT at record start
*                 UT2 - UT at record end
*                 MAGCONJLAT - magnetic conjugate geodetic latitude
*                 MAGCONJLON - magnetic conjugate longitude
*      outCount (num outputs) = 1 (SZENC)
*      outputArr - double array holding:
*                 SZEN - Solar zenith angle at magnetic conjugate in deg (0 = overhead)
*
*   Algorithm: See solarzen in geometry.c
*
*   returns - 0 (successful)
*/
int getSZenc(int inCount,
             double * inputArr,
             int outCount,
             double * outputArr,
             FILE * errFile)



/*********************************************************************
*
* getConjSun  derives sunset, sunrise, and shadow height for the magnetic conjugate
*             point in space and time for that day UT.
*
*   arguments:
*      inCount (num inputs) = 5 (UT1, UT2, MAGCONJLAT, MAGCONJLON, GDALT)
*      inputArr - double array holding:
*                 UT1 - UT at record start
*                 UT2 - UT at record end
*                 MAGCONJLAT - magnetic conjugate latitude
*                 MAGCONJLON - magnetic conjugate longitude
*                 GDALT - geodetic altitude in km
*      outCount (num outputs) = 5 (CONJ_SUNRISE, CONJ_SUNSET, CONJ_SUNRISE_H, CONJ_SUNSET_H, MA
*      outputArr - double array holding:
*                 CONJ_SUNRISE - time (UT) that day of sunrise at magnetic conjugate
```

```
*                CONJ_SUNSET - time (UT) that day of sunset at magnetic conjugate
*                CONJ_SUNRISE_H - time (in hours UT) that day of sunrise at magnetic conjugate
*                CONJ_SUNSET_H - time (in hours UT) that day of sunset at magnetic conjugate
*                MAGCONJSDWHT - shadow height at that time at magnetic conjugate
*
*   Algorithm: See sunrise_set, shadowheight in geometry.c
*
*   returns - 0 (successful)
*/
int getConjSun(int inCount,
               double * inputArr,
               int outCount,
               double * outputArr,
               FILE * errFile)



/**************************************************************************
*
* getGeo   derives geophysical parameter Kp, Ap3, Ap, f10.7, and fbar
*          given a time
*
*   arguments:
*      inCount (num inputs) = 2 (UT1, UT2)
*      inputArr - double array holding:
*                UT1 - UT at record start
*                UT2 - UT at record end
*      outCount (num outputs) = 5 (KP, AP3, AP, F10.7, and FBAR)
*      outputArr - double array holding:
*                KP - Kp Index
*                AP3 - ap index (3-hourly)
*                AP - AP index (daily)
*                F10.7 - solar flux observed (Ottawa)
*                FBAR - Multiday average observed (Ott)
*
*   Algorithm: Get data from geo500101g.002 at average UT.  Only
*              the first time any thread calls this method will the
*              data file be read into static variable geoDayArr.  Array
*              of GeoDay structs is used since it has a more compact
*              memory footprint than the madrec data structure, and
*              geo500101g.002 is a large file.  The old style file
*              geo500101g.001 will be used if geo500101g.002 is not found.
*
*   returns - 0 if successful, -1 if problem finding either file
*/
int getGeo(int inCount,
           double * inputArr,
           int outCount,
           double * outputArr,
           FILE * errFile)



/**************************************************************************
*
* getDst   derives geophysical parameter Dst given a time
*
*   arguments:
*      inCount (num inputs) = 2 (UT1, UT2)
*      inputArr - double array holding:
*                UT1 - UT at record start
```

```
*                 UT2 - UT at record end
*      outCount (num outputs) = 1 (DST)
*      outputArr - double array holding:
*                 DST - DST index in nT
*
*   Algorithm: Get data from dst570101g.002 at average UT.  Only
*              the first time any thread calls this method will the
*              data file be read into static variable dstDayArr. Array
*              of DstDay structs is used since it has a more compact
*              memory footprint than the madrec data structure, and
*              dst570101g.002 is a large file.  The old style file
*              dst570101g.001 will be used if dst570101g.002 is not found.
*
*   returns - 0 if successful, -1 if problem finding file
*/
int getDst(int inCount,
           double * inputArr,
           int outCount,
           double * outputArr,
           FILE * errFile)



/*****************************************************************
*
* getFof2   derives geophysical parameter Fof2 (above particular station) given a time
*
*   arguments:
*      inCount (num inputs) = 2 (UT1, UT2)
*      inputArr - double array holding:
*                 UT1 - UT at record start
*                 UT2 - UT at record end
*                 KINST - station kinst above which Fof2 is measured
*      outCount (num outputs) = 1 (FOF@)
*      outputArr - double array holding:
*                 FOF2 - Fof2 in MHz above station
*
*   Algorithm: For now, only works for Millstone, but can be extended
*              to work for any instrument for which data is available. For
*              kinst = 30,31,32, get data from uld761203g.002 at average UT.  Only
*              the first time any thread calls this method will the
*              data file be read into static variable fof2DayArr. Array
*              of Fof2Day structs is used since it has a more compact
*              memory footprint than the madrec data structure, and
*              uld761203g.002 is a large file.  The old style file
*              uld761203g.001 will be used if uld761203g.002 is not found.
*
*   returns - 0 if successful, -1 if problem finding file
*/
int getFof2(int inCount,
            double * inputArr,
            int outCount,
            double * outputArr,
            FILE * errFile)



/*****************************************************************
*
* getPopl   derives Popl from log10(Pop).
*
```

```
*
*   arguments:
*      inCount (num inputs) = 1 (POP)
*      inputArr - double array holding:
*         POP - Uncorrected electron density (Te/Ti=1) (m-3)
*      outCount (num outputs) = 1 (POPL)
*      outputArr - double array holding:
*         POPL -  Log10(uncorrected electron density)  lg(m-3)
*
*   returns - 0 (successful)
*/
int getPopl(int inCount,
            double * inputArr,
            int outCount,
            double * outputArr,
            FILE * errFile)



/**************************************************************************
*
* getPop   derives Pop from 10^(Popl).
*
*
*   arguments:
*      inCount (num inputs) = 1 (POPL)
*      inputArr - double array holding:
*         POPL -  Log10(uncorrected electron density) lg(m-3)
*      outCount (num outputs) = 1 (POP)
*      outputArr - double array holding:
*         POP -  Uncorrected electron density (Te/Ti=1) (m-3)
*
*   returns - 0 (successful)
*/
int getPop(int inCount,
           double * inputArr,
           int outCount,
           double * outputArr,
           FILE * errFile)



/**************************************************************************
*
* getNel   derives Nel from log10(Ne).
*
*
*   arguments:
*      inCount (num inputs) = 1 (NE)
*      inputArr - double array holding:
*         NE - electron density (m-3)
*      outCount (num outputs) = 1 (NEL)
*      outputArr - double array holding:
*         NEL -  Log10(electron density) lg(m-3)
*
*   returns - 0 (successful)
*/
int getNel(int inCount,
           double * inputArr,
           int outCount,
           double * outputArr,
```

```
            FILE * errFile)


/**************************************************************************
*
* getNe   derives Ne from 10^(Nel).
*
*
*   arguments:
*      inCount (num inputs) = 1 (NEL)
*      inputArr - double array holding:
*         NEL -  Log10(electron density) lg(m-3)
*      outCount (num outputs) = 1 (NE)
*      outputArr - double array holding:
*         NE -  electron density (m-3)
*
*   returns - 0 (successful)
*/
int getNe(int inCount,
           double * inputArr,
           int outCount,
           double * outputArr,
           FILE * errFile)



/**************************************************************************
*
* getDNel   derives DNel from DNe.
*
*
*   arguments:
*      inCount (num inputs) = 1 (DNE)
*      inputArr - double array holding:
*         DNE - error in electron density (m-3)
*      outCount (num outputs) = 1 (DNEL)
*      outputArr - double array holding:
*         DNEL -  Log10(error in electron density) lg(m-3)
*
*   returns - 0 (successful)
*/
int getDNel(int inCount,
            double * inputArr,
            int outCount,
            double * outputArr,
            FILE * errFile)



/**************************************************************************
*
* getDNe   derives DNe from DNel.
*
*
*   arguments:
*      inCount (num inputs) = 1 (DNEL)
*      inputArr - double array holding:
*         DNEL -  Log10(error in electron density) lg(m-3)
*      outCount (num outputs) = 1 (DNE)
*      outputArr - double array holding:
```

```
*         DNE -  error in electron density (m-3)
*
*   returns - 0 (successful)
*/
int getDNe(int inCount,
           double * inputArr,
           int outCount,
           double * outputArr,
           FILE * errFile)



/*****************************************************************
*
* getNemaxl   derives Nemaxl from log10(Nemax).
*
*
*   arguments:
*      inCount (num inputs) = 1 (NEMAX)
*      inputArr - double array holding:
*         NEMAX - Maximum electron density (m-3)
*      outCount (num outputs) = 1 (NEMAXL)
*      outputArr - double array holding:
*         NEMAXL -  Log10(maximum electron density) lg(m-3)
*
*   returns - 0 (successful)
*/
int getNemaxl(int inCount,
            double * inputArr,
            int outCount,
            double * outputArr,
            FILE * errFile)



/*****************************************************************
*
* getNemax   derives Nemax from 10^(Nemaxl).
*
*
*   arguments:
*      inCount (num inputs) = 1 (NEMAXL)
*      inputArr - double array holding:
*         NEMAXL -  Log10(maximum electron density) lg(m-3)
*      outCount (num outputs) = 1 (NEMAX)
*      outputArr - double array holding:
*         NEMAX -  Maximum electron density (m-3)
*
*   returns - 0 (successful)
*/
int getNemax(int inCount,
           double * inputArr,
           int outCount,
           double * outputArr,
           FILE * errFile)



/*****************************************************************
*
* getTr   derives temperature ratio Te/Ti.
```

```
*
*
*   arguments:
*      inCount (num inputs) = 2 (TE, TI)
*      inputArr - double array holding:
*         TE -  Electron temperature - K
*         TI -  Ion temperature - K
*      outCount (num outputs) = 1 (TR)
*      outputArr - double array holding:
*         TR -  Temperature ratio (Te/Ti)
*
*   returns - 0 (successful)
*/
int getTr(int inCount,
          double * inputArr,
          int outCount,
          double * outputArr,
          FILE * errFile)



/**************************************************************************
*
* getTe   derives electron temperature Te from Tr*Ti.
*
*
*   arguments:
*      inCount (num inputs) = 2 (TR, TI)
*      inputArr - double array holding:
*         TR -  Temperature ratio (Te/Ti)
*         TI -  Ion temperature - K
*      outCount (num outputs) = 1 (TE)
*      outputArr - double array holding:
*         TE -  Electron temperature - K
*
*   returns - 0 (successful)
*/
int getTe(int inCount,
          double * inputArr,
          int outCount,
          double * outputArr,
          FILE * errFile)



/**************************************************************************
*
* getTi   derives ion temperature Ti from Te/Tr.
*
*
*   arguments:
*      inCount (num inputs) = 2 (TE, TR)
*      inputArr - double array holding:
*         TE -  Electron temperature - K
*         TR -  Temperature ratio (Te/Ti)
*      outCount (num outputs) = 1 (TI)
*      outputArr - double array holding:
*         TI -  Ion temperature - K
*
*   returns - 0 (successful)
*/
```

```
int getTi(int inCount,
          double * inputArr,
          int outCount,
          double * outputArr,
          FILE * errFile)



/*****************************************************************
*
* getDteCctitr   derives error in electron temperature given CCTITR, TI, TR, DTI, and DTR.
*
*
*   arguments:
*      inCount (num inputs) = 5 (CCTITR, TI, TR, DTI, DTR)
*      inputArr - double array holding:
*         CCTITR -  Ti, Tr correlation coefficient
*         TI -  Ion temperature - K
*         TR -  Temperature ratio (Te/Ti)
*         DTI -  Error in Ion temperature - K
*         DTR -  Error in Temperature ratio (Te/Ti)
*      outCount (num outputs) = 1 (DTE)
*      outputArr - double array holding:
*         DTE -  Error in Electron temperature - K
*
*   Algorithm: DTE = SQRT(TR**2*DTI**2 + TI**2*DTR**2 + 2.0D0*TR*TI*CVTITR),
*       where CVTITR = DTI*DTE*CCTITR.
*
*   Uses Ti, Tr correlation coefficient coefficient, unlike getDte, which uses TE
*
*   returns - 0 (successful)
*/
int getDteCctitr(int inCount,
                 double * inputArr,
                 int outCount,
                 double * outputArr,
                 FILE * errFile)



/*****************************************************************
*
* getDte   derives error in electron temperature given TE, TI, TR, DTI, and DTR.
*
*
*   arguments:
*      inCount (num inputs) = 5 (TE, TI, TR, DTI, DTR)
*      inputArr - double array holding:
*         TE -  Electron temperature - K
*         TI -  Ion temperature - K
*         TR -  Temperature ratio (Te/Ti)
*         DTI -  Error in Ion temperature - K
*         DTR -  Error in Temperature ratio (Te/Ti)
*      outCount (num outputs) = 1 (DTE)
*      outputArr - double array holding:
*         DTE -  Error in Electron temperature - K
*
*   Algorithm: DTE = TE * ((DTR/TR)**2 + (DTI/TI)**2)**1/2.
*
*   returns - 0 (successful)
*/
```

```
int getDte(int inCount,
           double * inputArr,
           int outCount,
           double * outputArr,
           FILE * errFile)



/**************************************************************************
*
* getCol   derives Col from log10(Co).
*
*
*   arguments:
*      inCount (num inputs) = 1 (CO)
*      inputArr - double array holding:
*         CO - Ion-neutral collision frequency - (s^-1)
*      outCount (num outputs) = 1 (COL)
*      outputArr - double array holding:
*         COL -  Log10(Ion-neutral collision frequency) - lg(s^-1)
*
*   returns - 0 (successful)
*/
int getCol(int inCount,
            double * inputArr,
            int outCount,
            double * outputArr,
            FILE * errFile)



/**************************************************************************
*
* getCo   derives Co from 10^(Col).
*
*
*   arguments:
*      inCount (num inputs) = 1 (COL)
*      inputArr - double array holding:
*         COL -  Log10(Ion-neutral collision frequency) - lg(s^-1)
*      outCount (num outputs) = 1 (CO)
*      outputArr - double array holding:
*         CO -  Ion-neutral collision frequency - (s^-1)
*
*   returns - 0 (successful)
*/
int getCo(int inCount,
           double * inputArr,
           int outCount,
           double * outputArr,
           FILE * errFile)



/**************************************************************************
*
* getNeNel   derives Ne and Nel from Ti, Tr, Popl, and aspect angle.
*
*
*   arguments:
*      inCount (num inputs) = 4 (TI, TR, POPL, ASPECT)
```

```
*      inputArr - double array holding:
*         TI -  Ion temperature - K
*         TR -  Temperature ratio (Te/Ti)
*         POPL -  Log10(uncorrected electron density) lg(m-3)
*         ASPECT - magnetic aspect angle (0 = up B)
*      outCount (num outputs) = 2 (NE, NEL)
*      outputArr - double array holding:
*         NE -  Corrected electron density (m^-3)
*         NEL -  log10 of Corrected electron density lg(m^-3)
*
*   Algorithm - see getElecDensity
*
*   returns - 0 (successful)
*/
int getNeNel(int inCount,
             double * inputArr,
             int outCount,
             double * outputArr,
             FILE * errFile)



/*****************************************************************
*
* getDNeDNel   derives DNe and DNel from Ti, Tr, Popl, and aspect angle and associated errors.
*
*
*   arguments:
*      inCount (num inputs) = 7 (TI, TR, POPL, ASPECT, DTI, DTR, DPOPL)
*      inputArr - double array holding:
*         TI -  Ion temperature - K
*         TR -  Temperature ratio (Te/Ti)
*         POPL -  Log10(uncorrected electron density) lg(m-3)
*         ASPECT - magnetic aspect angle (0 = up B)
*         DTI -  Error in Ion temperature - K
*         DTR -  Error in Temperature ratio (Te/Ti)
*         DPOPL -  Error in Log10(uncorrected electron density) lg(m-3)
*      outCount (num outputs) = 2 (DNE, DNEL)
*      outputArr - double array holding:
*         DNE -  Error in Corrected electron density (m^-3)
*         DNEL -  Error in log10 of Corrected electron density lg(m^-3)
*
*   Algorithm - see getElecDensity
*
*   returns - 0 (successful)
*/
int getDNeDNel(int inCount,
               double * inputArr,
               int outCount,
               double * outputArr,
               FILE * errFile)



/*****************************************************************
*
* getVisrNe   derives Model Ne, Nel using Shunrong's model
*
*
*   arguments:
*      inCount (num inputs) = 6 (UT1, KINST, SLT, GDALT, GDLAT, ELM)
```

```
*      inputArr - double array holding:
*         UT1 - UT at record start
*         KINST - instrument id
*         SLT - Local solar time in hours (0-24)
*         GDALT - geodetic altitude in km
*         GDLAT - geodetic latitude
*         ELM - mean elevation in degrees
*      outCount (num outputs) = 2 (NE_MODEL, NEL_MODEL)
*      outputArr - double array holding:
*         NE_MODEL -  Model electron density (m^-3)
*         NEL_MODEL -  log10 of Model electron density lg(m^-3)
*
*   Algorithm - see Shunrong's model at http://madrigal.haystack.mit.edu/models/
*
*   returns - 0 (successful)
*/
int getVisrNe(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/**********************************************************************
*
* getVisrTe   derives Model Te using Shunrong's model
*
*
*   arguments:
*      inCount (num inputs) = 6 (UT1, KINST, SLT, GDALT, GDLAT, ELM)
*      inputArr - double array holding:
*         UT1 - UT at record start
*         KINST - instrument id
*         SLT - Local solar time in hours (0-24)
*         GDALT - geodetic altitude in km
*         GDLAT - geodetic latitude
*         ELM - mean elevation in degrees
*      outCount (num outputs) = 1 (TE_MODEL)
*      outputArr - double array holding:
*         TE_MODEL -  Model electron temperature (K)
*
*   Algorithm - see Shunrong's model at http://madrigal.haystack.mit.edu/models/
*
*   returns - 0 (successful)
*/
int getVisrTe(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/**********************************************************************
*
* getVisrTi   derives Model Ti using Shunrong's model
*
*
*   arguments:
*      inCount (num inputs) = 6 (UT1, KINST, SLT, GDALT, GDLAT, ELM)
```

```
*      inputArr - double array holding:
*         UT1 - UT at record start
*         KINST - instrument id
*         SLT - Local solar time in hours (0-24)
*         GDALT - geodetic altitude in km
*         GDLAT - geodetic latitude
*         ELM - mean elevation in degrees
*      outCount (num outputs) = 1 (TI_MODEL)
*      outputArr - double array holding:
*         TI_MODEL -  Model ion temperature (K)
*
*   Algorithm - see Shunrong's model at http://madrigal.haystack.mit.edu/models/
*
*   returns - 0 (successful)
*/
int getVisrTi(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/*****************************************************************
*
* getVisrVo   derives Model Vo using Shunrong's model
*
*
*   arguments:
*      inCount (num inputs) = 6 (UT1, KINST, SLT, GDALT, GDLAT, ELM)
*      inputArr - double array holding:
*         UT1 - UT at record start
*         KINST - instrument id
*         SLT - Local solar time in hours (0-24)
*         GDALT - geodetic altitude in km
*         GDLAT - geodetic latitude
*         ELM - mean elevation in degrees
*      outCount (num outputs) = 1 (VO_MODEL)
*      outputArr - double array holding:
*         VO_MODEL -  Model line-of sight velocity (up=+) (m/s)
*
*   Algorithm - see Shunrong's model at http://madrigal.haystack.mit.edu/models/
*
*   returns - 0 (successful)
*/
int getVisrVo(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/*****************************************************************
*
* getVisrHNMax   derives Model HMax and NMax using Shunrong's model
*
*
*   arguments:
*      inCount (num inputs) = 6 (UT1, KINST, SLT, GDALT, GDLAT, ELM)
*      inputArr - double array holding:
```

```
*         UT1 - UT at record start
*         KINST - instrument id
*         SLT - Local solar time in hours (0-24)
*         GDALT - geodetic altitude in km
*         GDLAT - geodetic latitude
*         ELM - mean elevation in degrees
*      outCount (num outputs) = 2 (HMAX_MODEL, NMAX_MODEL)
*      outputArr - double array holding:
*         HMAX_MODEL -  ISR-based empirical model of HMAX
*         NMAX_MODEL -  ISR-based empirical model of NMAX
*
*   Algorithm - see Shunrong's model at http://madrigal.haystack.mit.edu/models/
*
*   returns - 0 (successful)
*/
int getVisrHNMax(int inCount,
                 double * inputArr,
                 int outCount,
                 double * outputArr,
                 FILE * errFile)



/**********************************************************************
*
* getVisrNeDiff   derives ne_modeldiff - the difference between measured and model ne
*
*
*   arguments:
*      inCount (num inputs) = 2 (NE, NE_MODEL)
*      inputArr - double array holding:
*         NE -  Measured electron density (m^-3)
*         NE_MODEL -  Model electron density (m^-3)
*      outCount (num outputs) = 1 (NE_MODELDIFF)
*      outputArr - double array holding:
*         NE_MODELDIFF -  Measured Ne - Model electron density (m^-3)
*
*   returns - 0 (successful)
*/
int getVisrNeDiff(int inCount,
                  double * inputArr,
                  int outCount,
                  double * outputArr,
                  FILE * errFile)



/**********************************************************************
*
* getVisrNelDiff   derives nel_modeldiff - the difference between measured and model nel
*
*
*   arguments:
*      inCount (num inputs) = 2 (NEL, NEL_MODEL)
*      inputArr - double array holding:
*         NEL -  Measured electron density log10(m^-3)
*         NEL_MODEL -  Model electron density log10(m^-3)
*      outCount (num outputs) = 1 (NEL_MODELDIFF)
*      outputArr - double array holding:
*         NEL_MODELDIFF -  Measured Nel - Model electron density log10(m^-3)
*
```

```
*   returns - 0 (successful)
*/
int getVisrNelDiff(int inCount,
                   double * inputArr,
                   int outCount,
                   double * outputArr,
                   FILE * errFile)



/**************************************************************************
*
* getVisrTeDiff   derives te_modeldiff - the difference between measured and model te
*
*
*   arguments:
*      inCount (num inputs) = 2 (TE, TE_MODEL)
*      inputArr - double array holding:
*         TE -  Measured electron temperature (K)
*         TE_MODEL -  Model electron temperature (K)
*      outCount (num outputs) = 1 (TE_MODELDIFF)
*      outputArr - double array holding:
*         TE_MODELDIFF -  Measured Te - Model electron temperature (K)
*
*   returns - 0 (successful)
*/
int getVisrTeDiff(int inCount,
                  double * inputArr,
                  int outCount,
                  double * outputArr,
                  FILE * errFile)



/**************************************************************************
*
* getVisrTiDiff   derives ti_modeldiff - the difference between measured and model ti
*
*
*   arguments:
*      inCount (num inputs) = 2 (TI, TI_MODEL)
*      inputArr - double array holding:
*         TI -  Measured ion temperature (K)
*         TI_MODEL -  Model ion temperature (K)
*      outCount (num outputs) = 1 (TI_MODELDIFF)
*      outputArr - double array holding:
*         TI_MODELDIFF -  Measured TI - Model ion temperature (K)
*
*   returns - 0 (successful)
*/
int getVisrTiDiff(int inCount,
                  double * inputArr,
                  int outCount,
                  double * outputArr,
                  FILE * errFile)



/**************************************************************************
*
* getVisrVoDiff   derives vo_modeldiff - the difference between measured and model vo
```

```
*
*
*   arguments:
*      inCount (num inputs) = 2 (VO, VO_MODEL)
*      inputArr - double array holding:
*         VO -  Measured line-of sight velocity (up=+) (m/s)
*         VO_MODEL -  Model line-of sight velocity (up=+) (m/s)
*      outCount (num outputs) = 1 (VO_MODELDIFF)
*      outputArr - double array holding:
*         VO_MODELDIFF -  Measured Vo - Model line-of sight velocity (up=+) (m/s)
*
*   returns - 0 (successful)
*/
int getVisrVoDiff(int inCount,
                  double * inputArr,
                  int outCount,
                  double * outputArr,
                  FILE * errFile)



/**************************************************************************
*
* getSn   derives Sn given Snp3
*
*
*   arguments:
*      inCount (num inputs) = 1 (SNP3)
*      inputArr - double array holding:
*         SNP3 - Signal to noise ratio
*      outCount (num outputs) = 1 (SN)
*      outputArr - double array holding:
*         SN - Signal to noise ratio
*
*   Algorithm - SN and SNP3 differ only in Cedar scaling factor, which
*               is handled at a lower level
*
*   returns - 0 (successful)
*/
int getSn(int inCount,
          double * inputArr,
          int outCount,
          double * outputArr,
          FILE * errFile)



/**************************************************************************
*
* getSnp3   derives Snp3 given Sn
*
*
*   arguments:
*      inCount (num inputs) = 1 (SN)
*      inputArr - double array holding:
*         SN - Signal to noise ratio
*      outCount (num outputs) = 1 (SNP3)
*      outputArr - double array holding:
*         SNP3 - Signal to noise ratio
*
*   Algorithm - SN and SNP3 differ only in Cedar scaling factor, which
```

```
*               is handled at a lower level
*
*   returns - 0 (successful)
*/
int getSnp3(int inCount,
            double * inputArr,
            int outCount,
            double * outputArr,
            FILE * errFile)



/**************************************************************************
*
* getChip31   derives Chip3 given Chisq
*
*
*   arguments:
*      inCount (num inputs) = 1 (CHISQ)
*      inputArr - double array holding:
*         CHISQ - Reduced-chi square of fit
*      outCount (num outputs) = 1 (CHIP3)
*      outputArr - double array holding:
*         CHIP3 - Reduced-chi square of fit
*
*   Algorithm - CHISQ and CHIP3 differ only in Cedar scaling factor, which
*               is handled at a lower level
*
*   returns - 0 (successful)
*/
int getChip31(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/**************************************************************************
*
* getWchsq1   derives Wchsq given Chip3
*
*
*   arguments:
*      inCount (num inputs) = 1 (CHIP3)
*      inputArr - double array holding:
*         CHIP3 - Reduced-chi square of fit
*      outCount (num outputs) = 1 (WCHSQ)
*      outputArr - double array holding:
*         WCHSQ - Reduced-chi square of fit
*
*   Algorithm - WCHSQ and CHIP3 differ only in Cedar scaling factor, which
*               is handled at a lower level
*
*   returns - 0 (successful)
*/
int getWchsq1(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)
```

```
/**********************************************************************
*
* getChisq1   derives Chisq given Wchsq
*
*
*   arguments:
*      inCount (num inputs) = 1 (WCHSQ)
*      inputArr - double array holding:
*         WCHSQ - Reduced-chi square of fit
*      outCount (num outputs) = 1 (CHISQ)
*      outputArr - double array holding:
*         CHISQ - Reduced-chi square of fit
*
*   Algorithm - CHISQ and WCHSQ differ only in Cedar scaling factor, which
*               is handled at a lower level
*
*   returns - 0 (successful)
*/
int getChisq1(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/**********************************************************************
*
* getChip32   derives Chip3 given Wchsq
*
*
*   arguments:
*      inCount (num inputs) = 1 (WCHSQ)
*      inputArr - double array holding:
*         WCHSQ - Reduced-chi square of fit
*      outCount (num outputs) = 1 (CHIP3)
*      outputArr - double array holding:
*         CHIP3 - Reduced-chi square of fit
*
*   Algorithm - WCHSQ and CHIP3 differ only in Cedar scaling factor, which
*               is handled at a lower level
*
*   returns - 0 (successful)
*/
int getChip32(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/**********************************************************************
*
* getWchsq2   derives Wchsq given Chisq
*
*
*   arguments:
```

```
*      inCount (num inputs) = 1 (CHISQ)
*      inputArr - double array holding:
*         CHISQ - Reduced-chi square of fit
*      outCount (num outputs) = 1 (WCHSQ)
*      outputArr - double array holding:
*         WCHSQ - Reduced-chi square of fit
*
*   Algorithm - CHISQ and WCHSQ differ only in Cedar scaling factor, which
*               is handled at a lower level
*
*   returns - 0 (successful)
*/
int getWchsq2(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/*****************************************************************
*
* getChisq2   derives Chisq given Chip3
*
*
*   arguments:
*      inCount (num inputs) = 1 (CHIP3)
*      inputArr - double array holding:
*         CHIP3 - Reduced-chi square of fit
*      outCount (num outputs) = 1 (CHISQ)
*      outputArr - double array holding:
*         CHISQ - Reduced-chi square of fit
*
*   Algorithm - CHISQ and CHIP3 differ only in Cedar scaling factor, which
*               is handled at a lower level
*
*   returns - 0 (successful)
*/
int getChisq2(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/*****************************************************************
*
* getVi1Vi1f   derives Vi1 given Vi1f
*
*
*   arguments:
*      inCount (num inputs) = 1 (VI1F)
*      inputArr - double array holding:
*         VI1F - F-region ion velocity in dir 1 (east)
*      outCount (num outputs) = 1 (VI1)
*      outputArr - double array holding:
*         VI1 - Ion velocity in direction 1 (eastward)
*
*   Algorithm - VI1 is just a generalized version of VI1F, so equal
*
```

```
*   returns - 0 (successful)
*/
int getVi1Vi1f(int inCount,
               double * inputArr,
               int outCount,
               double * outputArr,
               FILE * errFile)



/**************************************************************************
*
* getVi2Vi2f   derives Vi2 given Vi2f
*
*
*   arguments:
*      inCount (num inputs) = 1 (VI2F)
*      inputArr - double array holding:
*         VI2F - F-region ion velocity in dir 2 (north)
*      outCount (num outputs) = 1 (VI2)
*      outputArr - double array holding:
*         VI2 - Ion velocity in direction 2 (northward)
*
*   Algorithm - VI2 is just a generalized version of VI2F, so equal
*
*   returns - 0 (successful)
*/
int getVi2Vi2f(int inCount,
               double * inputArr,
               int outCount,
               double * outputArr,
               FILE * errFile)



/**************************************************************************
*
* getVipeVipe1   derives Vipe given Vipe1
*
*
*   arguments:
*      inCount (num inputs) = 1 (VIPE1)
*      inputArr - double array holding:
*         VIPE1 - Ion velocity in direction 4 (perp east)
*      outCount (num outputs) = 1 (VIPE)
*      outputArr - double array holding:
*         VIPE - Ion velocity in direction 4 (perp east)
*
*   Algorithm - VIPE1 and VIPE differ only in Cedar scaling factor, which
*               is handled at a lower level
*
*   returns - 0 (successful)
*/
int getVipeVipe1(int inCount,
                 double * inputArr,
                 int outCount,
                 double * outputArr,
                 FILE * errFile)
```

```
/*****************************************************************
*
* getVipeVipe2   derives Vipe given Vipe2
*
*
*   arguments:
*      inCount (num inputs) = 1 (VIPE2)
*      inputArr - double array holding:
*         VIPE2 - Ion velocity in direction 4 (perp east)
*      outCount (num outputs) = 1 (VIPE)
*      outputArr - double array holding:
*         VIPE - Ion velocity in direction 4 (perp east)
*
*   Algorithm - VIPE2 and VIPE differ only in Cedar scaling factor, which
*               is handled at a lower level
*
*   returns - 0 (successful)
*/
int getVipeVipe2(int inCount,
                 double * inputArr,
                 int outCount,
                 double * outputArr,
                 FILE * errFile)



/*****************************************************************
*
* getVipnVipn1   derives Vipn given Vipn1
*
*
*   arguments:
*      inCount (num inputs) = 1 (VIPN1)
*      inputArr - double array holding:
*         VIPN1 - Ion velocity in direction 5 (perp north)
*      outCount (num outputs) = 1 (VIPN)
*      outputArr - double array holding:
*         VIPN - Ion velocity in direction 5 (perp north)
*
*   Algorithm - VIPN1 and VIPN differ only in Cedar scaling factor, which
*               is handled at a lower level
*
*   returns - 0 (successful)
*/
int getVipnVipn1(int inCount,
                 double * inputArr,
                 int outCount,
                 double * outputArr,
                 FILE * errFile)



/*****************************************************************
*
* getVipnVipn2   derives Vipn given Vipn2
*
*
*   arguments:
*      inCount (num inputs) = 1 (VIPN2)
*      inputArr - double array holding:
*         VIPN2 - Ion velocity in direction 5 (perp north)
```

```
*      outCount (num outputs) = 1 (VIPN)
*      outputArr - double array holding:
*         VIPN - Ion velocity in direction 5 (perp north)
*
*   Algorithm - VIPN2 and VIPN differ only in Cedar scaling factor, which
*               is handled at a lower level
*
*   returns - 0 (successful)
*/
int getVipnVipn2(int inCount,
                 double * inputArr,
                 int outCount,
                 double * outputArr,
                 FILE * errFile)



/**************************************************************************
*
* getVi6Vipu   derives Vipu given Vi6
*
*
*   arguments:
*      inCount (num inputs) = 1 (VI6)
*      inputArr - double array holding:
*         VI6 - Ion velocity in dir 6 (antiparallel)
*      outCount (num outputs) = 1 (VIPU)
*      outputArr - double array holding:
*         VIPU - Parallel ion velocity (+ upward)
*
*   Algorithm - VI6 and VIPU differ only in sign
*
*   returns - 0 (successful)
*/
int getVi6Vipu(int inCount,
               double * inputArr,
               int outCount,
               double * outputArr,
               FILE * errFile)



/**************************************************************************
*
* getViGeom   derives geomagnetic ion velocity given geodetic ion veleocity
*
*
*   arguments:
*      inCount (num inputs) = 7 (VI1, VI2, VI3, BN, BE, BD, BMAG)
*      inputArr - double array holding:
*         VI1 - Ion velocity in direction 1 (eastward)
*         VI2 - Ion velocity in direction 2 (northward)
*         VI3 - Ion velocity in direction 3 (up)
*         BN - Northward component of geomagnetic fld
*         BE - Eastward component of geomagnetic field
*         BD - Downward component of geomagnetic field
*         BMAG - Geomagnetic field strength
*      outCount (num outputs) = 3 (VIPE, VIPN, VIPU)
*      outputArr - double array holding:
*         VIPE - Ion velocity in direction 4 (perp east)
*         VIPN - Ion velocity in direction 5 (perp north)
```

```
*         VIPU - Parallel ion velocity (+ upward)
*
*   Algorithm - See http://www.openradar.org/pipermail/openradar-madrigal/2011-September/000086
*
*   returns - 0 (successful)
*/
int getViGeom(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/**********************************************************************
*
* getViGeod   derives geodetic ion velocity given geomagnetic ion veleocity
*
*
*   arguments:
*      inCount (num inputs) = 7 (VIPE, VIPN, VIPU, BN, BE, BD, BMAG)
*      inputArr - double array holding:
*         VIPE - Ion velocity in direction 4 (perp east)
*         VIPN - Ion velocity in direction 5 (perp north)
*         VIPU - Parallel ion velocity (+ upward)
*         BN - Northward component of geomagnetic fld
*         BE - Eastward component of geomagnetic field
*         BD - Downward component of geomagnetic field
*         BMAG - Geomagnetic field strength
*      outCount (num outputs) = 3 (VI1, VI2, VI3)
*      outputArr - double array holding:
*         VI1 - Ion velocity in direction 1 (eastward)
*         VI2 - Ion velocity in direction 2 (northward)
*         VI3 - Ion velocity in direction 3 (up)
*
*   Algorithm - See http://www.openradar.org/pipermail/openradar-madrigal/2011-September/000086
*
*   returns - 0 (successful)
*/
int getViGeod(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/**********************************************************************
*
* getVn1Vn1p2   derives Vn1 given Vn1P2
*
*
*   arguments:
*      inCount (num inputs) = 1 (VN1P2)
*      inputArr - double array holding:
*         VN1P2 - Neutral wind in direction 1 (eastward)
*      outCount (num outputs) = 1 (VN1)
*      outputArr - double array holding:
*         VN1 - Neutral wind in direction 1 (eastward)
*
*   Algorithm - VN1 and VN1P2 differ only in Cedar scaling factor, which
```

```
*               is handled at a lower level
*
*   returns - 0 (successful)
*/
int getVn1Vn1p2(int inCount,
                double * inputArr,
                int outCount,
                double * outputArr,
                FILE * errFile)



/**********************************************************************
*
* getVn2Vn2p2   derives Vn2 given Vn2P2
*
*
*   arguments:
*      inCount (num inputs) = 1 (VN2P2)
*      inputArr - double array holding:
*         VN2P2 - Neutral wind in direction 2 (northward)
*      outCount (num outputs) = 1 (VN2)
*      outputArr - double array holding:
*         VN2 - Neutral wind in direction 2 (northward)
*
*   Algorithm - VN2 and VN2P2 differ only in Cedar scaling factor, which
*               is handled at a lower level
*
*   returns - 0 (successful)
*/
int getVn2Vn2p2(int inCount,
                double * inputArr,
                int outCount,
                double * outputArr,
                FILE * errFile)



/**********************************************************************
*
* getVnGeom   derives geomagnetic neutral velocity given geodetic neutral velocity
*
*
*   arguments:
*      inCount (num inputs) = 7 (VN1, VN2, VN3, BN, BE, BD, BMAG)
*      inputArr - double array holding:
*         VN1 - Neutral wind in direction 1 (eastward)
*         VN2 - Neutral wind in direction 2 (northward)
*         VN3 - Neutral wind in direction 3 (up)
*         BN - Northward component of geomagnetic fld
*         BE - Eastward component of geomagnetic field
*         BD - Downward component of geomagnetic field
*         BMAG - Geomagnetic field strength
*      outCount (num outputs) = 3 (VN4, VN5, VN6)
*      outputArr - double array holding:
*         VN4 - Neutral wind in direction 4 (perp east)
*         VN5 - Neutral wind in direction 5 (perp north)
*         VN6 - Neutral wind in dir 6 (antiparallel)
*
*   Algorithm - See http://www.openradar.org/pipermail/openradar-madrigal/2011-September/000086
*
```

```
*   returns - 0 (successful)
*/
int getVnGeom(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/**************************************************************************
*
* getVnGeod   derives geodetic neutral velocity given geomagnetic neutral velocity
*
*
*   arguments:
*      inCount (num inputs) = 7 (VN4, VN5, VN6, BN, BE, BD, BMAG)
*      inputArr - double array holding:
*         VN4 - Neutral wind in direction 4 (perp east)
*         VN5 - Neutral wind in direction 5 (perp north)
*         VN6 - Neutral wind in dir 6 (antiparallel)
*         BN - Northward component of geomagnetic fld
*         BE - Eastward component of geomagnetic field
*         BD - Downward component of geomagnetic field
*         BMAG - Geomagnetic field strength
*      outCount (num outputs) = 3 (VN1, VN2, VN3)
*      outputArr - double array holding:
*         VN1 - Neutral wind in direction 1 (eastward)
*         VN2 - Neutral wind in direction 2 (northward)
*         VN3 - Neutral wind in direction 3 (up)
*
*   Algorithm - See http://www.openradar.org/pipermail/openradar-madrigal/2011-September/000086
*
*   returns - 0 (successful)
*/
int getVnGeod(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/**************************************************************************
*
* getEFGeom   derives geomagnetic electric field given geodetic electric field
*
*
*   arguments:
*      inCount (num inputs) = 7 (EE, EN, EU, BN, BE, BD, BMAG)
*      inputArr - double array holding:
*         EE - Electric field in direction 1 (eastward)
*         EN - Electric field in direction 2 (nrthward)
*         EU - Electric field in direction 3 (up)
*         BN - Northward component of geomagnetic fld
*         BE - Eastward component of geomagnetic field
*         BD - Downward component of geomagnetic field
*         BMAG - Geomagnetic field strength
*      outCount (num outputs) = 3 (EPE, EPN, EAP)
*      outputArr - double array holding:
*         EPE - Electric field in direction 4 (perp est)
```

```
*         EPN - Electric field in direction 5 (perp nth)
*         EAP - Electric field in direction 6 (antipara)
*
*   Algorithm - See http://www.openradar.org/pipermail/openradar-madrigal/2011-September/000086
*
*   returns - 0 (successful)
*/
int getEFGeom(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/*****************************************************************
*
* getEFGeod   derives geodetic electric field given geomagnetic electric field
*
*
*   arguments:
*      inCount (num inputs) = 7 (EPE, EPN, EAP, BN, BE, BD, BMAG)
*      inputArr - double array holding:
*         EPE - Electric field in direction 4 (perp est)
*         EPN - Electric field in direction 5 (perp nth)
*         EAP - Electric field in direction 6 (antipara)
*         BN - Northward component of geomagnetic fld
*         BE - Eastward component of geomagnetic field
*         BD - Downward component of geomagnetic field
*         BMAG - Geomagnetic field strength
*      outCount (num outputs) = 3 (EE, EN, EU)
*      outputArr - double array holding:
*         EE - Electric field in direction 1 (eastward)
*         EN - Electric field in direction 2 (nrthward)
*         EU - Electric field in direction 3 (up)
*
*   Algorithm - See http://www.openradar.org/pipermail/openradar-madrigal/2011-September/000086
*
*   returns - 0 (successful)
*/
int getEFGeod(int inCount,
              double * inputArr,
              int outCount,
              double * outputArr,
              FILE * errFile)



/*****************************************************************
*
* getJGeom   derives geomagnetic current density given geodetic current density
*
*
*   arguments:
*      inCount (num inputs) = 7 (J1, J2, J3, BN, BE, BD, BMAG)
*      inputArr - double array holding:
*         J1 - Electric current density eastward
*         J2 - Electric current density northward
*         J3 - Electric current density up
*         BN - Northward component of geomagnetic fld
*         BE - Eastward component of geomagnetic field
```

```
*         BD - Downward component of geomagnetic field
*         BMAG - Geomagnetic field strength
*      outCount (num outputs) = 3 (J4, J5, J6)
*      outputArr - double array holding:
*         J4 - Electric current density perp east
*         J5 - Electric current density perp north
*         J6 - Electric current density antiparallel
*
*   Algorithm - See http://www.openradar.org/pipermail/openradar-madrigal/2011-September/000086
*
*   returns - 0 (successful)
*/
int getJGeom(int inCount,
             double * inputArr,
             int outCount,
             double * outputArr,
             FILE * errFile)



/*****************************************************************
*
* getJGeod   derives geodetic current density given geomagnetic current density
*
*
*   arguments:
*      inCount (num inputs) = 7 (J4, J5, J6, BN, BE, BD, BMAG)
*      inputArr - double array holding:
*         J4 - Electric current density perp east
*         J5 - Electric current density perp north
*         J6 - Electric current density antiparallel
*         BN - Northward component of geomagnetic fld
*         BE - Eastward component of geomagnetic field
*         BD - Downward component of geomagnetic field
*         BMAG - Geomagnetic field strength
*      outCount (num outputs) = 3 (J1, J2, J3)
*      outputArr - double array holding:
*         J1 - Electric current density eastward
*         J2 - Electric current density northward
*         J3 - Electric current density up
*
*   Algorithm - See http://www.openradar.org/pipermail/openradar-madrigal/2011-September/000086
*
*   returns - 0 (successful)
*/
int getJGeod(int inCount,
             double * inputArr,
             int outCount,
             double * outputArr,
             FILE * errFile)



/*****************************************************************
*
* getNeut  uses MSIS 2000 model to predict neutral atmosphere parameters based
*          on geophysical parameters.  Includes anomalous oxygen component,
*          and also uses array of historical and present AP values, rather than
*          simply present AP value.  For now, errors are all set to 20% of
*          output value
*
```

```
*   arguments:
*      inCount (num inputs) = 5 (UT1, UT2, GDLAT, GLON, GDALT)
*      inputArr - double array holding:
*                 UT1 - UT at record start
*                 UT2 - UT at record end
*                 GDLAT - geodetic latitude
*                 GLON - geodetic longitude
*                 GDALT - geodetic altitude
*      outCount (num outputs) = 26 (TNM, TINFM, MOL, NTOTL, NN2L,
*                                   NO2L, NOL, NARL, NHEL, NHL,
*                                   NN4SL, NPRESL, PSH, DTNM, DTINFM, DMOL, DNTOTL, DNN2L,
*                                   DNO2L, DNOL, DNARL, DNHEL, DNHL,
*                                   DNN4SL, DNPRESL, DPSH)
*      outputArr - double array holding:
*                 TNM - Model neutral atmosphere temperature
*                 TINFM - Model Exospheric temperature
*                 MOL - Log10 (nutrl atm mass density in kg/m3)
*                 NTOTL - Log10 (nutrl atm number density in m-3)
*                 NN2L - Nutrl atm compositn-log10([N2] in m-3)
*                 NO2L - Nutrl atm compositn-log10([O2] in m-3)
*                 NOL - Nutrl atm composition-log10([O] in m-3)
*                 NARL - Nutrl atm compositn-log10([AR] in m-3)
*                 NHEL - Nutrl atm compositn-log10([HE] in M-3)
*                 NHL - Nutrl atm composition-log10([H] in m-3)
*                 NN4SL - Nutrl atm compstn-log10([N(4S)] in m-3)
*                 NPRESL - Neutral atmospher log10(pressure in Pa)
*                 PSH - Pressure scale height (m)
*                 DTNM - Error in Model neutral atmosphere temperature
*                 DTINFM - Error in Model Exospheric temperature
*                 DMOL - Error in Log10 (nutrl atm mass density in kg/m3)
*                 DNTOTL - Error in Log10 (nutrl atm number density in m-3)
*                 DNN2L - Error in Nutrl atm compositn-log10([N2] in m-3)
*                 DNO2L - Error in Nutrl atm compositn-log10([O2] in m-3)
*                 DNOL - Error in Nutrl atm composition-log10([O] in m-3)
*                 DNARL - Error in Nutrl atm compositn-log10([AR] in m-3)
*                 DNHEL - Error in Nutrl atm compositn-log10([HE] in M-3)
*                 DNHL - Error in Nutrl atm composition-log10([H] in m-3)
*                 DNN4SL - Error in Nutrl atm compstn-log10([N(4S)] in m-3)
*                 DNPRESL - Error in Neutral atmospher log10(pressure in Pa)
*                 DPSH - Error in Pressure scale height (m)
*
*
*   Algorithm: Calls MSIS subroutine gtd7d after collecting geophysical data.
*              Includes AP data from present and up to 57 hours prior to
*              present.
*
*              Gets pressure via ideal gas law (p=nkT)
*              Gets scale height via kT/mg, where m is average mass
*
*   returns - 0 (successful)
*/
int getNeut(int inCount,
            double * inputArr,
            int outCount,
            double * outputArr,
            FILE * errFile)



/*****************************************************************
*
```

```
* getTn  uses MSIS model and ISR data to derive Tn.
*
*   arguments:
*      inCount (num inputs) = 9 (TI, TE, NE, PH+, NOL, NHL, NN4SL, NO2L, NHEL)
*      inputArr - double array holding:
*                 TI - Ion temperature (K)
*                 TE - Electron temperature (K)
*                 NE - Electron density (m^3)
*                 PHP - Composition - [HE+]/Ne
*                 NOL - Nutrl atm composition-log10([O] in m-3) (MSIS)
*                 NHL - Nutrl atm composition-log10([H] in m-3) (MSIS)
*                 NN4SL - Nutrl atm compstn-log10([N(4S)] in m-3) (MSIS)
*                 NO2L - Nutrl atm compositn-log10([O2] in m-3) (MSIS)
*                 NHEL - Nutrl atm compositn-log10([HE] in M-3) (MSIS)
*      outCount (num outputs) = 1 (TN )
*      outputArr - double array holding:
*                 TN - Neutral atmosphere temperature (K)
*
*   Algorithm: See tnf.f in madf/geolib
*
*   returns - 0 (successful)
*/
int getTn(int inCount,
          double * inputArr,
          int outCount,
          double * outputArr,
          FILE * errFile)



/*****************************************************************
*
* getTnNoPhp  uses MSIS model and ISR data to derive Tn - uses Php = 0.
*
*   Meant to be called to get Tn if Php not in measured data
*
*   arguments:
*      inCount (num inputs) = 8 (TI, TE, NE, NOL, NHL, NN4SL, NO2L, NHEL)
*      inputArr - double array holding:
*                 TI - Ion temperature (K)
*                 TE - Electron temperature (K)
*                 NE - Electron density (m^3)
*                 NOL - Nutrl atm composition-log10([O] in m-3) (MSIS)
*                 NHL - Nutrl atm composition-log10([H] in m-3) (MSIS)
*                 NN4SL - Nutrl atm compstn-log10([N(4S)] in m-3) (MSIS)
*                 NO2L - Nutrl atm compositn-log10([O2] in m-3) (MSIS)
*                 NHEL - Nutrl atm compositn-log10([HE] in M-3) (MSIS)
*      outCount (num outputs) = 1 (TN )
*      outputArr - double array holding:
*                 TN - Neutral atmosphere temperature (K)
*
*   Algorithm: See tnf.f in madf/geolib
*
*   returns - 0 (successful)
*/
int getTnNoPhp(int inCount,
               double * inputArr,
               int outCount,
               double * outputArr,
               FILE * errFile)
```

```
/*****************************************************************
*
* getDTn  uses MSIS model and ISR data to derive error in Tn.
*
*   arguments:
*      inCount (num inputs) = 16 (TI, TE, NE, NOL, NHL, NN4SL, NO2L, NHEL,
*                                 DTI, DTE, DNE, DNOL, DNHL, DNN4SL, DNO2L, DNHEL)
*      inputArr - double array holding:
*                 TI - Ion temperature (K)
*                 TE - Electron temperature (K)
*                 NE - Electron density (m^3)
*                 NOL - Nutrl atm composition-log10([O] in m-3) (MSIS)
*                 NHL - Nutrl atm composition-log10([H] in m-3) (MSIS)
*                 NN4SL - Nutrl atm compstn-log10([N(4S)] in m-3) (MSIS)
*                 NO2L - Nutrl atm compositn-log10([O2] in m-3) (MSIS)
*                 NHEL - Nutrl atm compositn-log10([HE] in M-3) (MSIS)
*                 DTI - Error in Ion temperature (K)
*                 DTE - Error in Electron temperature (K)
*                 DNE - Error in Electron density (m^3)
*                 DNOL - Error in Nutrl atm composition-log10([O] in m-3) (MSIS)
*                 DNHL - Error in Nutrl atm composition-log10([H] in m-3) (MSIS)
*                 DNN4SL - Error in Nutrl atm compstn-log10([N(4S)] in m-3) (MSIS)
*                 DNO2L - Error in Nutrl atm compositn-log10([O2] in m-3) (MSIS)
*                 DNHEL - Error in Nutrl atm compositn-log10([HE] in M-3) (MSIS)
*      outCount (num outputs) = 1 (DTN )
*      outputArr - double array holding:
*                 DTN - Error in Neutral atmosphere temperature (K)
*
*   Algorithm: See tnf.f in madf/geolib
*
*   returns - 0 (successful)
*/
int getDTn(int inCount,
           double * inputArr,
               int outCount,
               double * outputArr,
               FILE * errFile)



/*****************************************************************
*
* getCond  uses Shunrong's code from Schunk and Nagy to derive Pederson and Hall
*          local conductivities.
*
*   arguments:
*      inCount (num inputs) = 10 (TI, TE, NE, PH+_IRI, PO+_IRI, NOL, NN2L, NO2L, TNM, BMAG)
*      inputArr - double array holding:
*                 TI - Ion temperature (K)
*                 TE - Electron temperature (K)
*                 NE - Electron density (m^3)
*                 PH+_IRI - IRI Composition % = [H+]/Ne * 100
*                 PO+_IRI - IRI Composition % = [O+]/Ne * 100
*                 NOL - Nutrl atm composition-log10([O] in m-3) (MSIS)
*                 NN2L - Nutrl atm compositn-log10([N2] in m-3) (MSIS)
*                 NO2L - Nutrl atm compositn-log10([O2] in m-3) (MSIS)
*                 TNM - MSIS Neutral atmosphere temperature (K)
*                 BMAG - Geomagnetic field strength (Tesla)
*      outCount (num outputs) = 4 (PDCON, PDCONL, HLCON, HLCONL)
```

```
*      outputArr - double array holding:
*                  PDCON - Pedersen Conductivity in mho/m3
*                  PDCONL - Log10(Pedersen Conductivity in mho/m3)
*                  HLCON - Hall Conductivity in mho/m3
*                  HLCONL - Log10(Hall Conductivity in mho/m3)
*
*   Algorithm: See conduct.f in madf/geolib
*
*   returns - 0 (successful)
*/
int getCond(int inCount,
            double * inputArr,
            int outCount,
            double * outputArr,
            FILE * errFile)



/**********************************************************************
*
* getDCond  uses Shunrong's code from Schunk and Nagy to derive errors in Pederson and Hall
*          local conductivities.
*
*   arguments:
*      inCount (num inputs) = 16 (TI, TE, NE, PH+_IRI, PO+_IRI, NOL, NN2L, NO2L, TNM, BMAG,
*                                 DTI, DTE, DNE, DNOL, DNN2L, DNO2L)
*      inputArr - double array holding:
*                  TI - Ion temperature (K)
*                  TE - Electron temperature (K)
*                  NE - Electron density (m^3)
*                  PH+_IRI - IRI Composition % = [H+]/Ne * 100
*                  PO+_IRI - IRI Composition % = [O+]/Ne * 100
*                  NOL - Nutrl atm composition-log10([O] in m-3) (MSIS)
*                  NN2L - Nutrl atm compositn-log10([N2] in m-3) (MSIS)
*                  NO2L - Nutrl atm compositn-log10([O2] in m-3) (MSIS)
*                  TNM - MSIS Neutral atmosphere temperature (K)
*                  BMAG - Geomagnetic field strength (Tesla)
*                  DTI - Error in Ion temperature (K)
*                  DTE - Error in Electron temperature (K)
*                  DNE - Error in Electron density (m^3)
*                  DNOL - Error in Nutrl atm composition-log10([O] in m-3) (MSIS)
*                  DNN2L - Error in Nutrl atm compositn-log10([N2] in m-3) (MSIS)
*                  DNO2L - Error in Nutrl atm compositn-log10([O2] in m-3) (MSIS)
*      outCount (num outputs) = 4 (DPDCON, DPDCONL, DHLCON, DHLCONL)
*      outputArr - double array holding:
*                  PDCON - Error in Pedersen Conductivity in mho/m3
*                  PDCONL - Error in Log10(Pedersen Conductivity in mho/m3)
*                  HLCON - Error in Hall Conductivity in mho/m3
*                  HLCONL - Error in Log10(Hall Conductivity in mho/m3)
*
*   Algorithm: See conduct.f in madf/geolib
*
*   returns - 0 (successful)
*/
int getDCond(int inCount,
             double * inputArr,
             int outCount,
             double * outputArr,
             FILE * errFile)
```

```
/*****************************************************************
*
* getImf   gets interplanetary magnetic field data given a time
*
*   arguments:
*      inCount (num inputs) = 2 (UT1, UT2)
*      inputArr - double array holding:
*                 UT1 - UT at record start
*                 UT2 - UT at record end
*      outCount (num outputs) = 10 (BXGSM, BYGSM, BZGSM, BIMF,
*                                   BXGSE, BYGSE, BZGSE,
*                                   SWDEN, SWSPD, SWQ)
*      outputArr - double array holding:
*                 BXGSM - Interplanetary Mag Field (Bx GSM coord)
*                 BYGSM - Interplanetary Mag Field (By GSM coord)
*                 BYGSM - Interplanetary Mag Field (Bz GSM coord)
*                 BIMF - Interplanetary Mag Field strength
*                 BXGSE - Interplanetary Mag Field (Bx GSE coord)
*                 BYGSE - Interplanetary Mag Field (By GSE coord)
*                 BZGSE - Interplanetary Mag Field (Bz GSE coord)
*                 SWDEN - Solar Wind Plasma Density
*                 SWSPD - Solar Wind Plasma Speed
*                 SWQ - IMF/Solar Wind Qualifier
*
*
*   Algorithm: Get data from imf631127g.001 at average UT.  Only
*              the first time any thread calls this method will the
*              data file be read into static variable imfDayArr.  Array
*              of ImfDay structs is used since it has a more compact
*              memory footprint than the madrec data structure, and
*              imf631127g.002 is a large file.  The old style file
*              imf631127g.001 will be used if imf631127g.002 is not found. Note that
*              BXGSM and BXGSE are equal.
*
*   returns - 0 if successful, -1 if problem finding file
*/
int getImf(int inCount,
           double * inputArr,
           int outCount,
           double * outputArr,
           FILE * errFile)



/*****************************************************************
*
* getIri
*   arguments:
*      inCount (num inputs) = 5 (UT1, UT2, GDLAT, GDLON, GDALT)
*      inputArr - double array holding:
*                 UT1 - UT at record start
*                 UT2 - UT at record end
*                 GDLAT - geodetic latitude
*                 GLON - geodetic longitude
*                 GDALT - geodetic altitude


*      outCount (num outputs) = 11 (NE_IRI, NEL_IRI, TN_IRI, TI_IRI, TE_IRI,
*                                   PO+_IRI, PNO+_IRI, PO2+_IRI, PHE+_IRI, PH+_IRI, PN+_IRI)
*      outputArr - double array holding:
```

```
*                  NE_IRI - electron density
                   NEL_IRI - log of electron density
                   TN_IRI - IRI neutral temperature
                   TI_IRI - IRI ion temperature
                   TE_IRI - IRI electron temperature
                   PO+_IRI - IRI composition [O+]/Ne
                   PNO+_IRI - IRI compostion [NO+]/Ne
                   PO2_IRI - IRI composition [O2+]/Ne
                   PHE+_IRI - IRI composition [HE+]/Ne
                   PH+_IRI - IRI composition [H+]/Ne
                   PN+_IRI  - IRI composition [N+]/Ne

*     Written by Alicia Fernandez, May 2007
*
*
*   returns - 0 (successful)
*/
int getIri (int inCount,
            double * inputArr,
            int outCount,
            double * outputArr,
            FILE * errFile)



/**********************************************************************
*
* getTestAveAlt   dummy method that prints UT, then gets lowest value
*                 of gdlat as 1d parameter and average value as 2D parameter
*
*   arguments:
*      numRows - number of rows of 2D data in record
*      inCount (num inputs) = 2 (UT1, GDALT)
*      inputArr - array of double arrays holding:
*                 UT - 1D parameter
*                 GDALT - 2D parameter
*      outCount (num outputs) = 2 (LOW_GDALT, AVE_GDALT)
*      outputArr - array of double arrays holding:
*                 LOW_GDALT (len = 1)
*                 AVE_GDALT (len = numRows)
*
*   Algorithm: Average all GDALT values and find lowest.  This
*              is only a dummy method
*
*   returns - 0
*/
int getTestAveAlt(int numRows,
                  int inCount,
                  double ** inputArr,
                  int outCount,
                  double ** outputArr,
                  FILE * errFile)
```

Madrigal Fortran API

Madrigal contains a comprehensive set of Fortran-language procedures for working with Madrigal Database files. This part of the Fortran API is simply a very thin wrapper around the C API. Like the C API, this Fortran API is split into two modules, a high level module Maddata, which hides the difference between derived and measured data from the user, and a low-level module Madrec, which allows users to work directly with Cedar files. Examples of using the Maddata and Madrec modules are also given.

The Fortran API also includes the geolib library, which includes methods dealing with geometry and time, which is written completely in Fortran.

The following are suggested lines to add to your Makefile when using the Madrigal Fortran API:

```
# Library directory
LIBDIR = $(MADROOT)/lib

LDLIBS = -L$(LIBDIR) -lmadrec -lgeo -lm -lnsl

    if solaris:

LDFLAGS = -R$(LIBDIR)

    if gnu:

LDFLAGS = -Xlinker -R$(LIBDIR)
```

Maddata module

```
/*****************************************************************************************
      Fortran callable C routines for Fortran access to the C-language
      Madrigal API - Maddata module.

      These comments show how to call these methods from Fortran.

      To build, use -L$MADROOT/lib -lmadrec -lgeo
       See example program source/madc/testF77/tmaddataF77.f
      ------------------------------------------------------------------

      SUBROUTINE CRMADD(FILEC,
     *                  PARMS,
     *                  FLTSTR,
     *                  RFMADD,
     *                  NUMREC,
     *                  STATUS)
```

```
      CHARACTER FILEC*(*)
      CHARACTER PARMS*(*)
      CHARACTER FLTSTR*(*)
      INTEGER RFMADD, NUMREC, STATUS
      (for 64 bit machines, use INTEGER*8 RFMADD)
C
      CRMADD  Creates Maddata given a file, a list of desired parameters,
      and filters.
      CRMADD is meant to stand for "CReate MADData"
C
      Input parameters:
      FILEC - file name
      PARMS - comma delimited list of desired parameters
              (not case-sensitive)  Example PARMS: "gdalt,Azm,F10.7,PH+,Fof2"
      FLTSTR - The filter string is the same string that is used in the new isprint
       command line.  Filters are separated by spaces.  The allowed filters
       are:
C
          date1=mm/dd/yyyy  (starting date to be examined. If time1 not given, defaults to 0 UT
             Example: date1=01/20/1998
C
          time1=hh:mm:ss (starting UT time to be examined. If date1 given, is applied to date1.
                          If not, applies on the first day of the experiment.)
             Example: time1=13:30:00
C
          date2=mm/dd/yyyy (ending date to be examined.  If time2 not given, defaults to 0 UT.)
             Example: date2=01/21/1998
C
          time2=hh:mm:ss (ending UT time to be examined - If date2 not given, ignored.)
             Example: time2=15:45:00
C
          In the follow arguments ranges are used.  If any range value is not given, it may be
          indicate no lower or upper limit (but the comma is always required). Ranges are inclu
          of the end points:
C
          z=lower alt limit1, upper alt limit1 [or lower alt limit2 , upper alt limit2 ...] (km
             Example 1: z=100,500  (This would limit the geodetic altitude to 100 to 500 km.)
             Example 2: z=100,200or300,400  (This would limit the geodetic altitude to 100 to 2
                                             or 300 to 400 km.)
             Example 3: z=,200or300,400   (Since the lower limit of the first range is missing,
                                           would limit the geodetic altitude to anything below
                                           or from 300 to 400 km.)
C
          az=lower az limit1, upper az limit1 [or lower az limit2 , upper az limit2 ...] (from
             Example 1: az=100,120  (This would limit the azimuth to 100 to 120 degrees.)
             Example 2: az=-180,-90or90,180  (This would limit the azimuth to between -180 and
                                              to between 90 and 180 degrees.  Note this allows a
                                              through 180 degrees.)
C
          el=lower el limit1, upper el limit1 [or lower el limit2 , upper el limit2 ...] (from
             Example 1: el=0,45  (This would limit the elevation from 0 to 45 degrees.)

          plen=lower pl limit1, upper pl limit1 [or lower pl limit2 , upper pl limit2 ...] (pul
             Example 1: el=,5e-4  (This would limit the pulse length to 5e-4 seconds or less.)
C
C
          Free form filters using any mnemonic, or two mnemonics added, subtracted, multiplied,
          Any number of filters may be added:
C
          filter=[mnemonic] or [mnemonic1,[+*-/]mnemonic2] , lower limit1 , upper limit1 [or lo
             Example 1: filter=ti,500,1000or2000,3000   (Limits the data to points where Ti is
```

```
                                                    or between 2000 and 3000 degrees.  Note
                                                    those of the Cedar standard.)
            Example 2: filter=gdalt,-,sdwht,0,     (This filter implies "gdalt - sdwht" must be
                                                    sdwht is shadow height - the distance above
                                                    where the sun is first visible - this filte
                                                    in direct sunlight will be displayed.)
            Example 3: filter=ti,/,Dti,100,    (Limits the data to points where the ratio Ti/dT
C
          So an full FLTSTR argument might be:
C
            "date1=01/20/1998 time1=13:30:00 z=,200or300,400 filter=gdalt,-,sdwht,0, filter=ti
C
      Output parameters:
      RFMADD - a long integer reference to the Maddata created,
      used by all other methods.  Will return 0 if error
      occurs.  (for 64 bit machines, use INTEGER*8 RFMADD)
      NUMREC - number of records in Maddata
      STATUS - 0 if success, -1 if failure

      ************************************************************

      SUBROUTINE CRNFMD(PARMS,
     *                  UT,
     *                  KINST,
     *                  ONED,
     *                  TWOD,
     *                  RFMADD,
     *                  NROW)
      CHARACTER PARMS*(*)
      DOUBLE PRECISION UT
      CHARACTER ONED*(*)
      CHARACTER TWOD*(*)
      INTEGER KINST, RFMADD, NROW
C     (for 64 bit machines, use INTEGER*8 RFMADD)
C
      CRNFMD  Creates a Maddata record given some input data (no file needed) for a set time.
      CRNFMD is meant to stand for "CReate NonFile MadData"
C
      Input parameters:
C
      PARMS - comma delimited list of desired parameters
              (not case-sensitive)  Example PARMS: "gdalt,Azm,F10.7,PH+,Fof2"
C
      UT - seconds since 1/1/1950 to calculate data at
C
      KINST - instrument id.  If not important, use 0
C
      ONED - A string that sets one dimension data (That is, one value per parameter).  For exa
             "gdalt=100.0 glon=45.0 gdlat=-20.0"
C
      TWOD - A string that sets two dimension data (That is, NROW values per parameter, where e
             parameter must have the same number of values, if more than one).  For example, th
             follow TWOD string would produce 8 rows with every combination of gdlat = 45 or 50
             glon = 20 or 30, and gdalt = 500 or 600:
C
             "gdlat=45,45,45,45,50,50,50,50 glon=20,20,30,30,20,20,30,30 gdalt=500,600,500,600,
C
      Output parameters:
C
      RFMADD - a long integer reference to the Maddata created,
      used by all other methods.  Will return 0 if error
```

```
      occurs.  (for 64 bit machines, use INTEGER*8 RFMADD)
C
      NROW - number of rows in Record returned.  If TWOD string used, should be equal to the nu
             for each parameter.  If no TWOD data, should be 1.  If error, will be 0.
C
      ************************************************************


      SUBROUTINE GTNROW(RFMADD, RECNUM, NROW)
      INTEGER RFMADD
      INTEGER RECNUM, NROW
      (for 64 bit machines, use INTEGER*8 RFMADD)
C
      GTNROW  Gets the number of rows of data in record RECNUM.
      GTNROW is meant to stand for "GeT Number of ROWs"
C
      Input parameters:
      RFMADD - reference to data created by CRMADD
      RECNUM - record number of interest (starts at 1)
C
      Output parameter:
      NROW - number of rows of data.  If only 1D data requested, will
      always be 1.


      ************************************************************

      SUBROUTINE GTMADD(RFMADD, RECNUM, NROW, DATROW, STATUS)
      INTEGER RFMADD
      (for 64 bit machines, use INTEGER*8 RFMADD)
      INTEGER RECNUM
      INTEGER NROW, STATUS
      DOUBLE PRECISION DATROW(*)
C
      GTMADD  Gets one row of data via the DATROW array.
      GTMADD is meant to stand for "GeT MADData"
C
      Input parameters:
      RFMADD - reference to data created by CRMADD
      RECNUM - record number of interest (starts at 1)
      NROW - row number of interest (starts at 1)
C
      Output parameters:
      DATROW - Array of doubles to be filled with data.  Order of
      data is the same as the order of parameters in parms string
      passed into CRMADD.  User must be sure array size is equal to
      (or larger than) number of parameters requested.
C
      STATUS - 0 if success, -1 if failure


      ************************************************************


      SUBROUTINE FRMADD(RFMADD)
      INTEGER RFMADD
      (for 64 bit machines, use INTEGER*8 RFMADD)
C
      FRMADD  Frees the memory allocated by CRMADD.
      FRMADD is meant to stand for "FRee MADData"
C
```

```
      Input parameters:
      RFMADD - reference to data created by CRMADD
C
      This method should be called if your program wants to call
      CRMADD more than once.  Call FRMADD just before CRMADD to
      avoid a memory leak.  Will set RFMADD = 0. (for 64 bit machines, use INTEGER*8 RFMADD)

      ************************************************************


      SUBROUTINE STALOC(KINST,SLATGD,SLON,SALTGD,SLATGC,SR,ISTAT)
      INTEGER KINST,ISTAT
      DOUBLE PRECISION SLATGD,SLON,SALTGD,SLATGC,SR

C
      STALOC  Returns location of a given instrument (kinst)
      STALOC is meant to stand for "STAtion LOCation"
C
      Input parameters:
      KINST - instrument id
C
      Output parameters:
      SLATGD - instrument geodetic latitude
      SLON - instrument longitude (0 to 360)
      SALTGD - instrument geodetic altitude (km)
      SLATGC- instrument geocentric latitude
      SR - distance from center of earth (km)
      ISTAT - 0 if successful, -1 if not

*****************************************************************************/
```

Madrec Module

```
/*****************************************************************************************

Fortran callable C routines for Fortran access to the C-language
Madrigal API - madrec module.
_____________________________________________________________________________

INTEGER FUNCTION MDOPEN(IOTYPE, FILEC)
   INTEGER IOTYPE
   CHARACTER FILEC(128)
C
   Opens CEDAR file for sequential reading or writing.
C
   IOTYPE -  file type as described below
       Open Cedar file for sequential reading:
             1 - Determine file type automatically
            10 - Madrigal file
            11 - Blocked Binary file
            12 - Cbf file
            13 - Unblocked Binary file
            14 - Ascii file
       Create Cedar file for update; discard previous contents
          if any:
             2 - Madrigal file
            20 - Madrigal file
            21 - Blocked Binary file
            22 - Cbf file
```

```
            23 - Unblocked Binary file
            24 - Ascii file
       Create Cedar file in memory for sequential and random read and write.
            30 - Determine file type automatically
            40 - Madrigal file
            41 - Blocked Binary file
            42 - Cbf file
            43 - Unblocked Binary file
            44 - Ascii file
       Fast create Cedar file in memory for sequential and random read and write.
          Does not calculate min and and max parameter values
            50 - Determine file type automatically
            60 - Madrigal file
            61 - Blocked Binary file
            62 - Cbf file
            63 - Unblocked Binary file
            64 - Ascii file

   FILEC - file name
C
   Returns file handle (0-9) or negative value if file cannot be opened.
   Call MDGERR to get error description. At most 10 files can be open
   simultaneously.
```

```
INTEGER FUNCTION MDCLOS(MADFIL)
  INTEGER MADFIL

     Closes CEDAR file.

  MADFIL - File handle

  Returns: 0 - File closed successfully
           1 - Error closing file
           2 - File not open
           3 - Error flushing file
```

```
INTEGER FUNCTION MDREAD(MADFIL)
  INTEGER MADFIL

  Reads next CEDAR record

  MADFIL - File handle

  Returns: 0 - Record read successfully
           1 - Illegal file type
          -n - Error
```

```
INTEGER FUNCTION MDWRIT(MADFIL)
  INTEGER MADFIL

  Writes next Madrigal record

  MADFIL - File handle

  Returns: 0 - Record written successfully
           1 - Illegal file type
          -n - Error
```

```
INTEGER FUNCTION REWIND(MADFIL)
  INTEGER MADFIL

  Resets a file in memory to point to first record

  MADFIL - File handle

  Returns: 0 - Success
           1 - Illegal file type
          -n - Error
```

```
INTEGER FUNCTION GRECNO(MADFIL, RECNO)
  INTEGER MADFIL
  INTEGER RECNO

  Finds a given record number in a file (GRECNO stands for
  Get record by record number)

  MADFIL - File handle
  RECNO  - Record number (1 <= RECNO <= Total number of records)

  Returns: 0 - Success
          -1 - Specified record not in file
```

```
INTEGER FUNCTION GRCKEY(MADFIL, KEY)
  INTEGER MADFIL
  DOUBLE PRECISION KEY

  Finds a given record number in a file using the time key
  The record is the first data record for which key is
  greater than or equal to the start key of the
  record, and less than the start time of the
  following record. Thus, if the specified key
  corresponds to a time within a record, the
  first such record is returned. Header or catalog
  records are never returned. (GRCKEY stands for
  Get record by key)

  MADFIL - File handle
  KEY  - time in seconds since 1/1/1950

  Returns: 0 - Success
          -1 - Specified record not in file
```

```
INTEGER FUNCTION MDCOPY(MADFL1, MADFL2)
  INTEGER MADFL1
  INTEGER MADFL2

  Copies a record from one file to another

  MADFL1 - File handle of source record
  MADFL2 - File handle of destination record
```

```
  Returns: 0 - Success
           1 - Failure
```

```
INTEGER FUNCTION MDISDR(MADFIL)
  INTEGER MADFIL

  Identifies data records

  MADFIL - File handle

  Returns: 0 - Current record is not a data record
           1 - Current record is a data record
```

```
DOUBLE PECISION FUNCTION GTPMIN(MADFIL, PARCOD)
  INTEGER MADFIL
  INTEGER PARCOD

  Returns minimum value in file of given parcode

  MADFIL - File handle
  PARCOD - Parameter code

  Returns: Scaled minimum parameter value.  File must
  be opened in memory (types 30-44).  If not found,
  returns missing (1E-38).  Data rows with all error
  parameters invalid are not counted.
```

```
DOUBLE PECISION FUNCTION GTPMAX(MADFIL, PARCOD)
  INTEGER MADFIL
  INTEGER PARCOD

  Returns maximum value in file of given parcode

  MADFIL - File handle
  PARCOD - Parameter code

  Returns: Scaled maximum parameter value.  File must
  be opened in memory (types 30-44)  If not found,
  returns missing (1E-38).  Data rows with all error
  parameters invalid are not counted.
```

```
SUBROUTINE MDCREA(MADFIL,
            LPROL, JPAR, MPAR, NROW,
            KREC, KINST, KINDAT,
            YEAR1, MONTH1, DAY1,
            HOUR1, MIN1, SEC1, CSEC1,
            YEAR2, MONTH2, DAY2,
            HOUR2, MIN2, SEC2, CSEC2)
  MADFIL - File handle
  INTEGER MADFIL,LPROL, JPAR, MPAR, NROW,KREC, KINST, KINDAT,
          YEAR1, MONTH1, DAY1, HOUR1, MIN1, SEC1, CSEC1,
          YEAR2, MONTH2, DAY2, HOUR2, MIN2, SEC2, CSEC2
```

```
    Creates a madrigal record with the specified size and prolog. The
    1d and 2d parameters must be inserted later by calls to
    MDS1DP and MDS2DP.

       MADFIL - File handle
       LPROL   - Length of prolog
       JPAR    - Number of parameters in 1d array
       MPAR    - Number of parameters in 2d array
       NROW    - Number of rows in 2d array
       KREC    - Kind of record
       KINST   - Instrument code
       KINDAT  - Kind-of-data code
       YEAR1-CSEC1 - Start date and time
       YEAR2-CSEC2 - End date and time
```

```
SUBROUTINE MDG1DP(MADFIL, PARCOD, PARM)
  INTEGER MADFIL, PARCOD
  DOUBLE PRECISION PARM

  Gets a 1d parameter from the current record.

  MADFIL - File handle
  PARCOD - Parameter code
  PARM    - Parameter value
```

```
INTEGER FUNCTION MDGNRW(MADFIL) {
  INTEGER MADFIL

  Gets number of rows in 2d array.

  MADFIL - File handle

  Returns: Number of rows in 2d array.
```

```
INTEGER FUNCTION MDGKST(MADFIL) {
  INTEGER MADFIL

  Gets Kind of instrument (Kinst) integer.
  MDGKST stands for MaDrec Get KinST

  MADFIL - File handle

  Returns: Kind of instrument (Kinst) integer.
```

```
INTEGER FUNCTION MDGKDT(MADFIL) {
  INTEGER MADFIL

  Gets Kind of data (Kindat) integer.
  MDGKDT stands for MaDrec Get Kind of DaTa

  MADFIL - File handle

  Returns: Kind of data (Kindat) integer.
```

```
INTEGER FUNCTION MDIBYR(MADFIL) {
  INTEGER MADFIL

  Gets beginning year integer.

  MADFIL - File handle

  Returns: beginning year integer.
```

```
INTEGER FUNCTION MDIBDT(MADFIL) {
  INTEGER MADFIL

  Gets beginning date (MMDD) integer.

  MADFIL - File handle

  Returns: beginning date (MMDD) integer.
```

```
INTEGER FUNCTION MDIBHM(MADFIL) {
  INTEGER MADFIL

  Gets beginning hour/minute IBHM (HHMM) integer.

  MADFIL - File handle

  Returns: beginning hour/minute (HHMM) integer.
```

```
INTEGER FUNCTION MDIBCS(MADFIL) {
  INTEGER MADFIL

  Gets beginning second/centisecond IBCS (SSCC) integer.

  MADFIL - File handle

  Returns: beginning second/centisecond (SSCC) integer.
```

```
INTEGER FUNCTION MDIEYR(MADFIL) {
  INTEGER MADFIL

  Gets ending year integer.

  MADFIL - File handle

  Returns: ending year integer.
```

```
INTEGER FUNCTION MDIEDT(MADFIL) {
  INTEGER MADFIL

  Gets ending date (MMDD) integer.

  MADFIL - File handle

  Returns: ending date (MMDD) integer.
```

```
INTEGER FUNCTION MDIEHM(MADFIL) {
  INTEGER MADFIL

  Gets ending hour/minute IEHM (HHMM) integer.

  MADFIL - File handle

  Returns: ending hour/minute (HHMM) integer.
```

```
INTEGER FUNCTION MDIECS(MADFIL) {
  INTEGER MADFIL

  Gets ending second/centisecond IESC (SSCC) integer.

  MADFIL - File handle

  Returns: ending second/centisecond (SSCC) integer.
```

```
SUBROUTINE MDG2DP(MADFIL, PARCOD, PARM)
  INTEGER MADFIL, PARCOD
  DOUBLE PRECISION PARM(NRANMX)

  Gets a 2d parameter from the current record.

  MADFIL - File handle
  PARCOD - Parameter code
  PARM    - Parameter value array
```

```
SUBROUTINE MDGFLT(MADFIL, PARCOD, PARM)
  INTEGER MADFIL, PARCOD
  DOUBLE PRECISION PARM(NRANMX)

  Gets a flattened parameter from the current record. If its
  a 1D parameter, its value is copied into the first NROW
  doubles in PARM array.  If its a 2D parameter, it acts just
  like MDG2DP.  With this method all parameters act like 2D
  parameters.

  Stands for MaDrigal Get FLaTtened parameter

  MADFIL - File handle
  PARCOD - Parameter code (can be 1D or 2D)
  PARM    - Parameter value array
```

```
SUBROUTINE MDS1DP(MADFIL, PARCOD, PARM, INDEX)
  INTEGER MADFIL, PARCOD, INDEX
  DOUBLE PRECISION PARM

  Sets 1d parameter

  MADFIL - File handle
  PARCOD - Parameter code
  PARM    - Parameter value
  INDEX   - Position of parameter in 1d array (1<=INDEX<=JPAR)
```

```
  Note: to set special value missing, use PARM=1.e-38.
      To set special value assumed (for error parms only), use PARM=2.e-38
      To set special value knownbad (for error parms only), use PARM=3.e-38
```

```
SUBROUTINE MDS2DP(MADFIL, PARCOD, PARM, INDEX)
  INTEGER MADFIL

  Sets 2d parameter array

  MADFIL - File handle
  PARCOD - Parameter code
  PARM    - Parameter value
  INDEX   - Position of parameter in 2d array (1<=INDEX<=MPAR)

    Note: to set special value missing, use PARM=1.e-38.
      To set special value assumed (for error parms only), use PARM=2.e-38
      To set special value knownbad (for error parms only), use PARM=3.e-38
```

```
DOUBLE PRECISION FUNCTION MDSSFA(PARCOD)
  INTEGER PARCOD

  Gets parameter scale factor

  PARCOD - Parameter code

  Returns: Parameter scale factor
```

```
SUBROUTINE MDGERR(ERROR)
  CHARACTER ERROR(128)

  Gets most recent error message

  ERROR  - Error message
```

```
DOUBLE PRECISION FUNCTION GETKEY(YEAR,MON,DAY,HOUR,MIN,SEC)
  INTEGER YEAR,MON,DAY,HOUR,MIN,SEC

  Gets key (number of seconds) since YEAR,MON,DAY,HOUR,MIN,SEC
```

```
SUBROUTINE MDGEOD(MADFIL, ROW, GDLAT, GLON, GDALT)
  INTEGER MADFIL
  DOUBLE PRECISION GDLAT(NRANMX)
  DOUBLE PRECISION GLON(NRANMX)
  DOUBLE PRECISION GDALT(NRANMX)

  Gets GDLAT, GLON, GDALT from current record via cedarGetGeodetic.

  MADFIL - File handle, pointing to current record
  GDLAT - geodetic latitude array
  GLON - longitude array
  GDALT - geodetic altitude array
```

```
SUBROUTINE GTROOT(STROOT,LENGTH)
  CHARACTER STROOT(128)
  INTEGER LENGTH

  Gets MADROOT as set either in env variable or in cedar.h

  STROOT  - String holding MADROOT
  LENGTH - length of string copied
_____________________________________________________________________________


*****************************************************************************/
```

Example Fortran programs using maddataF77

The basic premise of the MaddataF77 module is to hide the difference between measured and derived parameters when working with Madrigal data. There are two ways to input measured data with the MaddataF77 module, with a file and without a file. In the first case, measured data comes from a particular file, and in the second the application supplies it directly through input parameters.

This example page includes simple examples of using measured data from a file, and using measured data supplied by the application.

Using the MaddataF77 module with a file

The following example prints both measured and derived data from a particular file, with various filters applied using a string very similar to the isprint command line. The key method is CRMADD.

```
C
C     This sample program prints out all requested parameters
C     from a madrigal file for given filters.  The requested
C     parameters can be either measured or derived - the API
C     hides these details from the user.
C
C     Note that RFMADD must be declared INTEGER or INTEGER*8,
C     depending on whether 32-bit or 64-bit platform

C
      PROGRAM TMADDATA
C
C     .. Local Scalars ..
      INTEGER RFMADD
C     ..If 64-bit, should be INTEGER*8 RFMADD
      INTEGER NUMREC
C     .. External Subroutines ..
      EXTERNAL CRMADD,GTNROW, GTMADD, FRMADD, STALOC
C     ..
C     .. Local Arrays ..
      CHARACTER*128 FILEC, MDROOT
      CHARACTER PARMS*100
      CHARACTER FLTSTR*1000
C     .. Local variables ..
      INTEGER STATUS
      DOUBLE PRECISION DATROW(100)
      DOUBLE PRECISION SLATGD,SLON,SALTGD,SLATGC,SR
```

```
      INTEGER REC, NROW, ROW, ISTAT, KINST
C     ..
C     ..
      RFMADD = 0
      NUMREC = 0
C     the requested madrigal file
      CALL GTROOT(MDROOT,LEN)
      FILEC=MDROOT(:LEN)//'/experiments/1998/mlh/20jan98/mil980120g.002'
C     the requested parms (measured or derived)
      PARMS = 'gdalt,ti,kp,nel'
C     the desired filters (exactly like isprint command line)
      FLTSTR = 'z=1000, filter=range,2500, filter=ti,1000,2000'
      CALL CRMADD(FILEC,PARMS,FLTSTR,RFMADD,NUMREC,STATUS)
      IF (STATUS .NE. 0) THEN
         PRINT*, "Create Maddata failed for ", FILEC
         STOP
      END IF
C     print all records (ignoring formatting)
      DO 30, REC = 1, NUMREC
         CALL GTNROW(RFMADD, REC, NROW)
         PRINT*,PARMS
C        loop over all rows
         DO 20, ROW=1, NROW
            CALL GTMADD(RFMADD, REC, ROW, DATROW, STATUS)
C           Note that data is stored in datrow in the order of parms
            PRINT "(F10.5)",DATROW(1),DATROW(2),DATROW(3),DATROW(4)
20       CONTINUE
30    CONTINUE
C     free all data
      CALL FRMADD(RFMADD)
C
      KINST = 31
      CALL STALOC(KINST,SLATGD,SLON,SALTGD,SLATGC,SR,ISTAT)
      if (ISTAT .EQ. 0) THEN
         print*, 'Kinst 31:', SLATGD,SLON,SALTGD,SLATGC,SR,ISTAT
      else
         print *, 'now returned error code ', ISTAT, SLATGD
      END IF

      CALL STALOC(3331,SLATGD,SLON,SALTGD,SLATGC,SR,ISTAT)
      if (ISTAT .EQ. 0) THEN
         print*, 'Kinst 3331:', SLATGD,SLON,SALTGD,SLATGC,SR,ISTAT
      else
         print*, 'Kinst 3331 failed, as it should'
      END IF

      END
```

Using the MaddataF77 module without a file

The following example prints both measured and derived data based on user supplied data alone. The key method is CRNFMD.

```
C     $Id: madrigal_doc.pdf 3751 2011-10-27 17:57:48Z brideout $
C
C     This sample program prints out all requested parameters
C     from a madrigal data passed in as arguments - no file needed.  The requested
C     parameters can be either measured or derived - the API
```

```
C     hides these details from the user.
C
      PROGRAM TMADDATA_NOFILE
C
C     .. Local Scalars ..
      INTEGER RFMADD, KINST
      DOUBLE PRECISION UT
C     .. External Subroutines ..
      EXTERNAL CRNFMD,GTNROW,GTMADD,FRMADD
C     .. External Functions ..
      DOUBLE PRECISION GETKEY
C     ..
C     .. Local Arrays ..
      CHARACTER*300 ONED, TWOD
      CHARACTER PARMS*100
C     .. Local variables ..
      INTEGER STATUS
      DOUBLE PRECISION DROW(100)
      INTEGER NROW, ROW
C     ..
C     ..
      RFMADD = 0
      NUMREC = 0
      KINST=31
      NROW=0
      UT = GETKEY(1998,1,20,15,0,0)
C     the requested parms
      PARMS = 'gdlat,glon,gdalt,bmag,kp,sdwht'
C     the one-D data (not used in this example, but can't hurt)
      ONED = 'pl=.001 sn=10'
C     the two-D data
      TWOD="gdlat=45,45,50,50 glon=20,30,20,30 gdalt=500,500,500,500"
      CALL CRNFMD(PARMS,UT,KINST,ONED,TWOD,RFMADD,NROW)
      IF (NROW .EQ. 0) THEN
         PRINT*, "Create non-file Maddata failed"
         STOP
      END IF
C     print the record (only 1 returned, so its always rec 1)
      PRINT*,PARMS
      CALL GTNROW(RFMADD, 1, NROW)
C     loop over all rows
      DO 20, ROW=1, NROW
         CALL GTMADD(RFMADD, 1, ROW, DROW, STATUS)
C        Note that data is stored in DROW in the order of parms
         PRINT*,DROW(1),DROW(2),DROW(3),DROW(4),DROW(5),DROW(6)
20    CONTINUE
C     free all data
      CALL FRMADD(RFMADD)
C
      END
```

Example Fortran program using madrecF77

```
C
      PROGRAM TMADREC
C
C     This program is a simple example illustrating the use of the madrec
C     module from Fortran - see madrecF77.c. This module is appropriate for
C     dealing with Madrigal files - writing them or modifying them.  To deal
```

```
C     with Madrigal data at a higher level, where the differences between
C     derived and measured data can be ignored, use maddataF77.c methods.
C
C     This program contain 4 examples:
C        1. Reading an existing madrigal file in sequentially
C        2. Writing a new madrigal file sequentially
C        3. Appending to an existing madrigal file using in mem file
C        4. Searching and summarizing a file in memory
C
C     .. Local Scalars ..
      INTEGER STATUS,NUMREC,LEN
      CHARACTER*128 ERROR,FNAME,MDROOT
      DOUBLE PRECISION PARM, KEY
      INTEGER LPROL, JPAR, MPAR, NROW, KREC, KINST, KINDAT
      INTEGER YEAR1, MONTH1, DAY1, HOUR1, MIN1, SEC1, CSEC1
      INTEGER YEAR2, MONTH2, DAY2, HOUR2, MIN2, SEC2, CSEC2
      INTEGER MADFIL, MADFL1, MADFL2
C     ..
C     .. Local Arrays ..
      DOUBLE PRECISION PARM2D(500)
C     ..
C     .. External Functions ..
      INTEGER MDOPEN,MDCLOS
      INTEGER MDREAD,MDISDR
      INTEGER MDWRIT,REWIND
      INTEGER MDCOPY,GRCKEY
      DOUBLE PRECISION GETKEY,GTPMIN,GTPMAX
C     ..
C     .. External Subroutines ..
      EXTERNAL MDGERR,MDG1DP,MDG2DP,MDCREA
      EXTERNAL MDS1DP,MDS2DP,GTROOT
C
      NUMREC = 0
      CALL GTROOT(MDROOT,LEN)
      FNAME=MDROOT(:LEN)//'/experiments/1998/mlh/20jan98/mil980120g.002'
      ERROR=''
C
C
C     .....Example 1 - read a madrigal file.....
C
C
      PRINT*, "\nExample 1 - read in a madrigal file"
      MADFIL = MDOPEN(1, FNAME)
      IF (MADFIL.LT.0) THEN
         CALL MDGERR(ERROR)
         PRINT*,ERROR
         STOP
      END IF
C     loop through all the records
10    STATUS = MDREAD(MADFIL)
      IF (STATUS.NE.0) THEN
          GOTO 100
      END IF
C     count data records, ignore header or catalog
      IF (MDISDR(MADFIL).EQ.1) THEN
          NUMREC = NUMREC + 1
      END IF
C     print some data for record 5
      IF (NUMREC.EQ.5) THEN
          PRINT*,"The following is data for the 5th data record:"
          CALL MDG1DP(MADFIL,402, PARM)
```

```
            PRINT*,"   Pulse length is "
            PRINT "(F10.5)",PARM
            CALL MDG2DP(MADFIL, 550, PARM2D)
            PRINT*,"   The first 4 Ti values are:"
            PRINT "(F10.5)",PARM2D(1),PARM2D(2),PARM2D(3),PARM2D(4)
         END IF
         GOTO 10
C
100      PRINT*,"The number of data records found in file:"
         PRINT"(I6)",NUMREC
C        ..Close the file..
         STATUS = MDCLOS(MADFIL)
C
C
C        ....Example 2: create a new madrigal file ....
C
C
         PRINT*,""
         PRINT*, "Example 2 - create new file tMadrecF77.out"
         MADFIL = MDOPEN(20, 'tMadrecF77.out')
         IF (MADFIL.LT.0) THEN
            CALL MDGERR(ERROR)
            PRINT*,ERROR
            STOP
         END IF
C        create a new record
         LPROL = 16
         JPAR  = 2
         MPAR  = 1
         NROW  = 3
         KREC  = 1002
         KINST = 32
         KINDAT = 3408
         YEAR1 = 2003
         MONTH1 = 3
         DAY1   = 19
         HOUR1  = 1
         MIN1   = 0
         SEC1   = 0
         CSEC1  = 0
         YEAR2 = 2003
         MONTH2 = 3
         DAY2   = 19
         HOUR2  = 1
         MIN2   = 2
         SEC2   = 59
         CSEC2  = 99
         CALL MDCREA(MADFIL,
        *      LPROL, JPAR, MPAR, NROW,
        *      KREC, KINST, KINDAT,
        *      YEAR1, MONTH1, DAY1,
        *      HOUR1, MIN1, SEC1, CSEC1,
        *      YEAR2, MONTH2, DAY2,
        *      HOUR2, MIN2, SEC2, CSEC2)
C        set the two 1D values (pl and systmp)
         PARM = 1.28000e-03
         CALL MDS1DP(MADFIL, 402, PARM, 1)
         PARM = 151.0
         CALL MDS1DP(MADFIL, 482, PARM, 2)
C        set the 1 2D parm (range) with 3 rows
         PARM2D(1)=100.0
```

```
      PARM2D(2)=150.0
      PARM2D(3)=200.0
      CALL MDS2DP(MADFIL, 120, PARM2D, 1)
C     write the new record to file
      STATUS = MDWRIT(MADFIL)
      IF (STATUS.NE.0) THEN
         CALL MDGERR(ERROR)
         PRINT*,ERROR
         STOP
      END IF
C     ..Close the file..
      STATUS = MDCLOS(MADFIL)
C
C
C     ....Example 3: Append to an existing madrigal file ....
C
C
      PRINT*,""
      PRINT*, "Example 3 - append to existing file"
      PRINT*, "  and save as tMadrecF77_append.out"
C     read the old file into memory
      MADFL1 = MDOPEN(50, FNAME)
      IF (MADFL1.LT.0) THEN
         CALL MDGERR(ERROR)
         PRINT*,ERROR
         STOP
      END IF
C     create a new record to append to it
      CALL MDCREA(MADFL1,
     *      LPROL, JPAR, MPAR, NROW,
     *      KREC, KINST, KINDAT,
     *      YEAR1, MONTH1, DAY1,
     *      HOUR1, MIN1, SEC1, CSEC1,
     *      YEAR2, MONTH2, DAY2,
     *      HOUR2, MIN2, SEC2, CSEC2)
C     set the two 1D values (pl and systmp)
      PARM = 1.28000e-03
      CALL MDS1DP(MADFL1, 402, PARM, 1)
      PARM = 151.0
      CALL MDS1DP(MADFL1, 482, PARM, 2)
C     set the 1 2D parm (range) with 3 rows
      PARM2D(1)=100.0
      PARM2D(2)=150.0
      PARM2D(3)=200.0
      CALL MDS2DP(MADFL1, 120, PARM2D, 1)
C     append the new record to the in-memory file
      STATUS = MDWRIT(MADFL1)
      IF (STATUS.NE.0) THEN
         CALL MDGERR(ERROR)
         PRINT*,ERROR
         STOP
      END IF
C     next rewind the in-memory file
      STATUS = REWIND(MADFL1)
      IF (STATUS.NE.0) THEN
         CALL MDGERR(ERROR)
         PRINT*,ERROR
         STOP
      END IF
C     now we open another file to write the appended file to
      MADFL2 = MDOPEN(20, 'tMadrecF77_append.out')
```

```
      IF (MADFL2.LT.0) THEN
         CALL MDGERR(ERROR)
         PRINT*,ERROR
         STOP
      END IF
C     loop through all the in memory records, copy to MADFL2
20    STATUS = MDREAD(MADFL1)
      IF (STATUS.NE.0) THEN
          GOTO 200
      END IF
      STATUS = MDCOPY(MADFL1, MADFL2)
C     write the copied record to file
      STATUS = MDWRIT(MADFL2)
      GOTO 20
C
200   STATUS = MDCLOS(MADFL1)
      STATUS = MDCLOS(MADFL2)
C
C
C     ....Example 4: Manipulate an in memory file ....
C
C
      PRINT*,""
      PRINT*, "Example 4: Manipulate an in memory file"
C     read the file into memory, this time with summary info
      MADFIL = MDOPEN(30, FNAME)
      IF (MADFIL.LT.0) THEN
         CALL MDGERR(ERROR)
         PRINT*,ERROR
         STOP
      END IF
      KEY = GETKEY(1998, 1, 20, 15, 0, 0)
      STATUS = GRCKEY(MADFIL,KEY)
      IF (STATUS.NE.0) THEN
         CALL MDGERR(ERROR)
         PRINT*,ERROR
         STOP
      END IF
C     print some data from this record
      PRINT*,"The following are min and max values of Ti in file:"
      PRINT "(F10.5)",GTPMIN(MADFIL,550),GTPMAX(MADFIL,550)
      PRINT*,"The following is data for key 1/20/1998 15:00:"
      CALL MDG1DP(MADFIL,402, PARM)
      PRINT*,"   Pulse length is "
      PRINT "(F10.5)",PARM
      CALL MDG2DP(MADFIL, 550, PARM2D)
      PRINT*,"   The first 4 Ti values are:"
      PRINT "(F10.5)",PARM2D(1),PARM2D(2),PARM2D(3),PARM2D(4)
      STATUS = MDCLOS(MADFIL)
C
      PRINT*,"\nTest complete"
C
      END
```

GEOLIB Procedure Synopsis

The geolib library (geolib.a) contains the following procedures focused on time and space.

```
************************************************************************

      SUBROUTINE CARMEL(B,XI,VL)
C
C     Private/Internal subroutine. Part of Apex coordinate computation
C     package. See COORD for public API. Computes scaler VL as a
C     function of B and XI.
C
C       Input:
C           B - Scaler field strength value.
C          XI - integral invariant (see INTEG).
C
C      Output:
C           VL - McIlwain's L-shell parameter i.e. Invariant Latitude
C                = ACOS(DSQRT(1.0D0/VL))/DTR
C
C     .. Scalar Arguments ..
      DOUBLE PRECISION B,VL,XI
C     ..



************************************************************************

      SUBROUTINE CONVRT(I,GDLAT,GDALT,GCLAT,RKM)
C
C     jmh - 11/79  ans fortran 66
C
C     Converts between geodetic and geocentric coordinates. the
C     reference geoid is that adopted by the iau in 1964. a=6378.16,
C     b=6356.7746, f=1/298.25. the equations for conversion from
C     geocentric to geodetic are from astron. j., vol 66, 1961, p. 15.
C
C       Input:
C             I - 1 geodetic to geocentric, 2 geocentric to geodetic
C       Input, Output:
C         GDLAT - geodetic latitude (degrees)
C         GDALT - altitude above geoid (km)
C         GCLAT - geocentric latitude (degrees)
C           RKM - geocentric radial distance (km)
C
C     .. Scalar Arguments ..
      DOUBLE PRECISION GCLAT,GDALT,GDLAT,RKM
      INTEGER I
C     ..



************************************************************************

      SUBROUTINE COORD(SLATGD,SLON,SR,SLATGC,TM,AZ,EL,RANGE,GDLAT,GLON,
     *                 GDALT,RCOR)
C
C     Calculates the listed coordinates of a specified point. the
C     point may be specified either by slatgd, slon, sr, slatgc, tm,
C     az, el, range (range .ge. 0.) or by gdlat, glon, gdalt (range
C     .lt. 0.)
C
C     Input:
C        SLATGD - station geodetic latitude
```

```
C        SLON   - station longitude
C        SR     - radial distance of station from center of earth
C        SLATGC - station geocentric latitude
C        TM     - time in years (e.g. 1975.2)
C        AZ     - radar azimuth
C        EL     - radar elevation
C        RANGE  - range to observation point
C     Input, output:
C        GDLAT  - geodetic latitude of observation point
C        GLON   - longitude of observation point
C        GDALT  - altitude above spheroid of observation point
C     Output:
C        RCOR(7)  - b     - magnitude of geomagnetic field
C        RCOR(8)  - br    - radial component of geomagnetic field
C        RCOR(9)  - bt    - southward component of geomagnetic field
C        RCOR(10) - bp    - eastward component of geomagnetic field
C        RCOR(11) - rlatm - dip latitude
C        RCOR(12) - rlati - invariant latitude
C        RCOR(13) - rl    - magnetic l parameter
C        RCOR(14) - alat  - apex latitude
C        RCOR(15) - alon  - apex longitude
C
C        RCOR(16) - g(1,1) magnetic coordinate system metric tensor,
C                           upper half stored row-wise
C        RCOR(17) - g(2,1) "                                      "
C        RCOR(18) - g(2,1) "                                      "
C        RCOR(19) - g(2,1) "                                      "
C
C        RCOR(20) - south-direction cosine w/respect to geodetic coords.
C        RCOR(21) - east-direction cosine "                              "
C        RCOR(22) - upward-direction cosine "                            "
C
C        RCOR(23) - perpendicular to b in magnetic n - s plane
C                   (magnetic south)
C        RCOR(24) - perpendicular to b in horizontal plane
C                   (magnetic east)
C        RCOR(25) - upward along magnetic field
C
C        RCOR(26) - x-direction cosine of a vector perpendicular to
C                   l.o.s. w/respect to apex coords.
C        RCOR(27) - y-direction cosine "                          "
C        RCOR(28) - z-direction cosine "                          "
C
C        RCOR(29) - inclination of geomagnetic field
C        RCOR(30) - declination of geomagnetic field
C        RCOR(31) - gclat - geocentric latitude
C        RCOR(32) - aspct - aspect angle
C        RCOR(33) - conjugate geocentric latitude
C        RCOR(34) - conjugate geodetic latitude
C        RCOR(35) - conjugate longitude
C
C     .. Scalar Arguments ..
      DOUBLE PRECISION AZ,EL,GDALT,GDLAT,GLON,RANGE,SLATGC,SLATGD,SLON,
     *                 SR,TM
C     ..
C     .. Array Arguments ..
      DOUBLE PRECISION RCOR(38)
C     ..
```

```
************************************************************************

      SUBROUTINE CSCONV(X,Y,Z,R,THETA,PHI,IMODE)
C
C     jmh - 11/79  ans fortran 66
C
C     Converts between cartesian coordinates x,y,z and spherical
C     coordinates r,theta,phi.  theta and phi are in degrees.
C
C       Input:
C         IMODE - 1 (x,y,z) -> (r,theta,phi)
C                 2 (r,theta,phi) -> (x,y,z)
C
C       Input, Output:
C              X, Y, Z - cartesian coordinates
C        R, THETA, PHI - spherical coordinates (degrees)
C
C     .. Scalar Arguments ..
      DOUBLE PRECISION PHI,R,THETA,X,Y,Z
      INTEGER IMODE
C     ..



************************************************************************

      SUBROUTINE DIPLAT(TM,GDLAT,GLON,GDALT,AINC,DEC,RLATM)
C     .. Scalar Arguments ..
      DOUBLE PRECISION AINC,DEC,GDALT,GDLAT,GLON,RLATM,TM
C     ..



************************************************************************

      SUBROUTINE DSF(GCLAT,GLON,RKM,ALT,HALT,ISTOP,DS)
C
C     Calculates an optimum integration step size for geomagnetic
C     field line tracing routine LINTRA, as an empirical function of
C     the geomagnetic dipole coordinates of the starting point. when
C     start and end points are in the same hemisphere and
C     abs(halt-alt)
************************************************************************

      SUBROUTINE GASPCT(SLATGD,SLON,SR,SLATGC,TM,AZ,EL,RANGE,GDLAT,GLON,
     *                  GDALT,B,CASPCT,ASPCT)
C
C     jmh - 3/88
C
C     *** warning *** this routing used to be called aspect. the name
C                     was changed on 3/2/88 to avoid conflict with a
C                     new routine of the same name in the geophysics
C                     library
C
C     Calculates the aspect angle between a radar beam and the
C     geomagnetic field at a specified point. the point may be
C     specified either by slatgd, slon, sr, slatgc, tm, az, el, range
C     (range .ge. 0.) or by gdlat, glon, gdalt (range .lt. 0.)
C
C     Input:
C        SLATGD - station geodetic latitude
```

```
C        SLON   - station longitude
C        SR     - radial distance of station from center of earth
C        SLATGC - station geocentric latitude
C        TM     - time in years (e.g. 1975.2)
C        AZ     - radar azimuth
C        EL     - radar elevation
C        RANGE  - range to observation point
C     Input, output:
C         GDLAT - geodetic latitude of observation point
C         GLON  - longitude of observation point
C         GDALT - altitude above spheroid of observation point
C     Output:
C        B      - geomagnetic field magnitude
C        CASPCT - cosine of aspect angle
C        ASPCT  - aspect angle (degrees)
C
C
C
C     .....subroutine parameter specifications.....
C
C     .....local specifications.....
C
C     .. Scalar Arguments ..
      DOUBLE PRECISION ASPCT,AZ,B,CASPCT,EL,GDALT,GDLAT,GLON,RANGE,
     *                 SLATGC,SLATGD,SLON,SR,TM
C     ..



*****************************************************************

      SUBROUTINE GDMAG(TM,GDLAT,GLON,GDALT,X,Y,Z,F,H,DEC,AINC)
C
C     jmh - 1/80  ans fortran 66
C
C     Evaluates the geomagnetic field at a point specified by its
C     geodetic coordinates. the reference geoid is that adopted by the
C     iau in 1964.
C
C     input:
C           TM - time in years for desired field (e.g. 1971.25)
C        GDLAT - geodetic latitude (degrees)
C         GLON - east longitude (degrees)
C        GDALT - altitude above geoid (km)
C
C     output:
C        X,Y,Z - geodetic field components (gauss)
C            F - magnitude of field (gauss)
C            H - horizontal intensity (gauss)
C          DEC - declination (degrees)
C         AINC - inclination (degrees)
C
C     .. Scalar Arguments ..
      DOUBLE PRECISION AINC,DEC,F,GDALT,GDLAT,GLON,H,TM,X,Y,Z
C     ..



*****************************************************************

      DOUBLE PRECISION FUNCTION GDRAN(SR,SLATGC,SLON,AZ,EL,ALT)
```

```
C
C     GDRAN uses a half-interval (binary) search technique to determine
C     the geodetic range to a point of observation given in terms of
C     azimuth, elevation, and geodetic altitude.
C
C       Input:
C            SR - radial distance of station from center of earth
C         SLATGC - station geocentric latitude
C           SLON - station longitude
C             AZ - radar azimuth
C             EL - radar elevation
C            ALT - geodetic altitude
C
C     harris fortran 77
C     rgm - 8/85
C
C     .. Scalar Arguments ..
      DOUBLE PRECISION ALT,AZ,EL,SLATGC,SLON,SR
C     ..



**********************************************************************

      SUBROUTINE GDV(GDLAT,GCLAT,FR,FT,FP,FX,FY,FZ)
C
C     jmh - 11/79  ans fortran 66
C
C     GDV converts a vector field f at geodetic latitude GDLAT and
C     geocentric latitude GCLAT from a geocentric based representation
C     to a geodetic based representation. the geocentric components
C     are FR (radial outward), FT (increasing geocentric colatitude,
C     e.g. southward) and FP (increasing east longitude). the
C     geodetic components are FX (northward, parallel to surface of
C     earth), FY (eastward, parallel to surface of earth) and FZ
C     (downward, perpendicular to surface of earth). FR,FT,FP thus
C     correspond to spherical coordinates r,theta,phi, with their
C     origin at the center of the earth. x,y,z are the coordinates
C     customarily used to describe the three components of the
C     geomagnetic field. FP and FY are the same.
C
C       Input:
C         GDLAT - geodetic latitude (degrees)
C         GCLAT - geocentric latitude (degrees)
C            FR - radial outward (geocentric).
C            FT - increasing geocentric colatitude (southward).
C            FP - increasing east longitude.
C
C      Output:
C            FX - northward, parallel to surface of earth (geodetic).
C            FY - eastward, parallel to surface of earth.
C            FZ - downward, perpendicular to surface of earth.
C
C     .. Scalar Arguments ..
      DOUBLE PRECISION FP,FR,FT,FX,FY,FZ,GCLAT,GDLAT
C     ..



**********************************************************************
```

```
      SUBROUTINE GMET(DXDQ,G)
C
C     jmh - 10/79  ans fortran 66
C
C     GMET calculates the metric tensor G of a coordinate system q
C     for which dx(i)/dq(j)=dxdq(i,j).
C
C       Input:
C         DXDQ - coordinate system array.
C
C      Output:
C            G - metric tensor.
C
C     .. Array Arguments ..
      DOUBLE PRECISION DXDQ(3,3),G(3,3)
C     ..



**********************************************************************

      SUBROUTINE GTS5(IYD,SEC,ALT,GLAT,GLONG,STL,F107A,F107,AP,MASS,D,T)
C        MSIS-86/CIRA 1986 Neutral Thermosphere Model
C         A.E.Hedin 3/15/85;2/26/87 (Variable Names Shortened)
C         9/21/87 M.E. Hagan (Non-Standard Statements Changed)
C     INPUT:
C        IYD - YEAR AND DAY AS YYDDD
C        SEC - UT(SEC)
C        ALT - ALTITUDE(KM) (GREATER THAN 85 KM)
C        GLAT - GEODETIC LATITUDE(DEG)
C        GLONG - GEODETIC LONGITUDE(DEG)
C        STL - LOCAL APPARENT SOLAR TIME(HRS)
C        F107A - 3 MONTH AVERAGE OF F10.7 FLUX
C        F107 - DAILY F10.7 FLUX FOR PREVIOUS DAY
C             UNITS:1.0e-22W/m2/Hz
C        AP - MAGNETIC INDEX(DAILY) OR WHEN SW(9)=-1. :
C           - ARRAY CONTAINING:
C             (1) DAILY AP
C             (2) 3 HR AP INDEX FOR CURRENT TIME
C             (3) 3 HR AP INDEX FOR 3 HRS BEFORE CURRENT TIME
C             (4) 3 HR AP INDEX FOR 6 HRS BEFORE CURRENT TIME
C             (5) 3 HR AP INDEX FOR 9 HRS BEFORE CURRENT TIME
C             (6) AVERAGE OF EIGHT 3 HR AP INDICIES FROM 12 TO 33 HRS
C                    PRIOR TO CURRENT TIME
C             (7) AVERAGE OF EIGHT 3 HR AP INDICIES FROM 36 TO 59 HRS
C                    PRIOR TO CURRENT TIME
C        MASS - MASS NUMBER (ONLY DENSITY FOR SELECTED GAS IS
C                  CALCULATED. MASS 0 IS TEMPERATURE. MASS 48 FOR ALL.
C
C     OUTPUT:
C        D(1) - HE NUMBER DENSITY(CM-3)
C        D(2) - O NUMBER DENSITY(CM-3)
C        D(3) - N2 NUMBER DENSITY(CM-3)
C        D(4) - O2 NUMBER DENSITY(CM-3)
C        D(5) - AR NUMBER DENSITY(CM-3)
C        D(6) - TOTAL MASS DENSITY(GM/CM3)
C        D(7) - H NUMBER DENSITY(CM-3)
C        D(8) - N NUMBER DENSITY(CM-3)
C        T(1) - EXOSPHERIC TEMPERATURE
C        T(2) - TEMPERATURE AT ALT
C
```

```
C      TO GET OUTPUT IN M-3 and KG/M3:   CALL METERS(.TRUE.)
C
C          ADDITIONAL COMMENTS
C           (1) LOWER BOUND QUANTITIES IN COMMON/GTS3C/
C           (2) TO TURN ON AND OFF PARTICULAR VARIATIONS CALL TSELEC(SW)
C               WHERE SW IS A 25 ELEMENT ARRAY CONTAINING 0. FOR OFF, 1.
C               FOR ON, OR 2. FOR MAIN EFFECTS OFF BUT CROSS TERMS ON
C               FOR THE FOLLOWING VARIATIONS
C               1 - F10.7 EFFECT ON MEAN  2 - TIME INDEPENDENT
C               3 - SYMMETRICAL ANNUAL    4 - SYMMETRICAL SEMIANNUAL
C               5 - ASYMMETRICAL ANNUAL   6 - ASYMMETRICAL SEMIANNUAL
C               7 - DIURNAL               8 - SEMIDIURNAL
C               9 - DAILY AP             10 - ALL UT/LONG EFFECTS
C              11 - LONGITUDINAL         12 - UT AND MIXED UT/LONG
C              13 - MIXED AP/UT/LONG     14 - TERDIURNAL
C              15 - DEPARTURES FROM DIFFUSIVE EQUILIBRIUM
C              16 - ALL TINF VAR         17 - ALL TLB VAR
C              18 - ALL T0 VAR           19 - ALL S VAR
C              20 - ALL Z0 VAR           21 - ALL NLB VAR
C              22 - ALL TR12 VAR         23 - TURBO SCALE HEIGHT VAR
C
C             To get current values of SW: CALL TRETRV(SW)
C
C     .. Scalar Arguments ..
      DOUBLE PRECISION ALT,F107,F107A,GLAT,GLONG,SEC,STL
      INTEGER IYD,MASS
      LOGICAL METER
C     ..
C     .. Array Arguments ..
      DOUBLE PRECISION AP(*),D(8),T(2)
C     ..
C     .. Scalars in Common ..
C     ..
C     .. Arrays in Common ..
C     ..


**********************************************************************

      DOUBLE PRECISION FUNCTION DENSS(ALT,DLB,TINF,TLB,XM,ALPHA,TZ,ZLB,
     *                                S2,T0,ZA,Z0,TR12)
C       Calculate Temperature and Density Profiles for MSIS models
C     .. Scalar Arguments ..
      DOUBLE PRECISION ALPHA,ALT,DLB,S2,T0,TINF,TLB,TR12,TZ,XM,Z0,ZA,ZLB
C     ..
C     .. Scalars in Common ..
C     ..



**********************************************************************

      DOUBLE PRECISION FUNCTION GLOBE5(YRD,SEC,LAT,LONG,TLOC,F107A,F107,
     *                 AP,P)
C       CALCULATE G(L) FUNCTION FOR MSIS-86/CIRA 1986
C       Upper Thermosphere Parameters
C     .. Scalar Arguments ..
      DOUBLE PRECISION F107,F107A,LAT,LONG,SEC,TLOC,YRD
C     ..
C     .. Array Arguments ..
```

```
      DOUBLE PRECISION AP(*),P(*)
C     ..
C     .. Scalars in Common ..
C     ..
C     .. Arrays in Common ..
C     ..



*****************************************************************

      SUBROUTINE TSELEC(SV)
C        SET SWITCHES
C     .. Array Arguments ..
      DOUBLE PRECISION SV(*)
C     ..
C     .. Scalars in Common ..
C     ..
C     .. Arrays in Common ..
C     ..



*****************************************************************

      DOUBLE PRECISION FUNCTION GLOB5L(P)
C      LIMITED PARAMETER VERSION OF GLOBE 9/2/82
C       CALCULATE G(L) FUNCTION FOR MSIS-86/CIRA 1986
C       Lower Thermosphere Parameters
C     .. Array Arguments ..
      DOUBLE PRECISION P(*)
C     ..
C     .. Scalars in Common ..
C     ..
C     .. Arrays in Common ..
C     ..



*****************************************************************

      DOUBLE PRECISION FUNCTION DNET(DD,DM,ZHM,XMM,XM)
C       8/20/80
C       TURBOPAUSE CORRECTION FOR MSIS MODELS
C       Eq. A12b
C     .. Scalar Arguments ..
      DOUBLE PRECISION DD,DM,XM,XMM,ZHM
C     ..



*****************************************************************

      DOUBLE PRECISION FUNCTION CCOR(ALT,R,H1,ZH)
C        CHEMISTRY/DISSOCIATION CORRECTION FOR MSIS MODELS
C     Eq. A20a or Eq. A21
C     .. Scalar Arguments ..
      DOUBLE PRECISION ALT,H1,R,ZH
C     ..
```

```
************************************************************************

      SUBROUTINE PRMSG5
C          CIRA     11-FEB-86
C     .. Arrays in Common ..
C     ..



************************************************************************

      DOUBLE PRECISION FUNCTION HFUN(SR,EL,RANGE)
C
C     jmh - 11/79  ans fortran 66
C
C     HFUN computes the height above a sphere (radius SR) of an
C     observation point at a specified elevation (EL) and range
C     (RANGE). SR and RANGE should be positive and EL should be in the
C     range 0.0 to 90.0.
C
C       Input:
C           SR - radial distance of station from center of earth
C           EL - radar elevation
C        RANGE - range to observation point
C
C     .. Scalar Arguments ..
      DOUBLE PRECISION EL,RANGE,SR
C     ..
C     .. Scalars in Common ..
C     ..



************************************************************************

      SUBROUTINE INTEG(ARC,BEG,BEND,B,JEP,ECO,FI)
C
C     Private/Internal subroutine. Part of Apex coordinate computation
C     package. See COORD for public API. INTEG determines the value of
C     the integral invariant FI by numerically integrating along the
C     field line from the specified point of interest to its conjugate
C     point.
C
C       Input:
C         ARC - Altitudes in earth radii (array).
C         BEG - floating point array.
C        BEND - floating point array.
C           B - Magnitude of field (array)
C         ECO - floating point array.
C         JEP - floating point scaler.
C
C      Output:
C           FI - floating point scaler.
C
C     .. Scalar Arguments ..
      DOUBLE PRECISION FI
      INTEGER JEP
C     ..
C     .. Array Arguments ..
      DOUBLE PRECISION ARC(200),B(200),BEG(200),BEND(200),ECO(200)
```

```
C     ..


*****************************************************************

      SUBROUTINE INVAR(TM,FLAT,FLONG,ALT,ERR,BB,FL)
C
C     Private/Internal subroutine. Part of Apex coordinate computation
C     package. See COORD for public API. INVAR converts coordinates
C     TM, FLAT, FLON and ALT to L-shell coordinates FL and BB. The
C     uncertainty in FL is typically less than 10.*ERR*FL (percent)
C
C       Input:
C           TM - time in years for desired field (e.g. 1971.25)
C         FLAT - geocentric latitude (degrees)
C        FLONG - east longitude
C          ALT - altitude (km)
C          ERR - tolerance factor
C
C      Output:
C           BB - Magnetic Field strength at point.
C           FL - McIlwain's L-shell parameter i.e.
C                Invariant Latitude = ACOS(DSQRT(1.0D0/FL))/DTR
C
C     .. Scalar Arguments ..
      DOUBLE PRECISION ALT,BB,ERR,FL,FLAT,FLONG,TM
C     ..


*****************************************************************

      SUBROUTINE INVLAT(TM,GDLAT,GLON,GDALT,RL,RLATI)
C     .. Scalar Arguments ..
      DOUBLE PRECISION GDALT,GDLAT,GLON,RL,RLATI,TM
C     ..


*****************************************************************

      SUBROUTINE ITERAT
C
C     Private/Internal subroutine. Part of Apex coordinate computation
C     package. See COORD for public API. ITERAT integrates magnetic
C     field line using a 4-point adams formula after initialization.
C     First 7 iterations advance point by 3*DS.
C
C       Input (via common block ITER):
C               L - step count. set l=1 first time thru,
C                   set l=l+1 thereafter.
C      B,BR,BT,BP - field + components at point y
C               ST - sine of geocentric colatitude
C              SGN - sgn=+1  traces in direction of field
C                    sgn=-1  traces in negative field direction
C               DS - integration stepsize (arc increment) in km
C
C      Input, Output (via common block ITER):
C                 Y - R,DLAT,DLON: geocentric tracing point coordinates
C                     (km,deg)
```

```
C
C      Output (via common block ITER):
C            YOLD - Y at iteration L-1
C
C     .. Scalars in Common ..
C     ..
C     .. Arrays in Common ..
C     ..


*****************************************************************

      SUBROUTINE LINES(R1,R2,R3,B,ARC,ERR,J,VP,VN,TM)
C
C     Private/Internal subroutine. Part of Apex coordinate computation
C     package. See COORD for public API. Makes repeated calls to the
C     IGRF, tracing magnetic field line to minimum B.
C
C       Input:
C          ERR - tolerance factor (see INVAR)
C           TM - Time in floating point years (e.g. 1995.7)
C
C       Input, Output:
C     R1,R2,R3 - field strength in geocentric directions.
C            B - Magnitude of field (array)
C          ARC - Altitudes in earth radii (array)
C           VP - Geocentric latitude (array)
C           VN - Geocentric longitude (array)
C
C      Output:
C            J - Number of points in trace.
C
C     .. Scalar Arguments ..
      DOUBLE PRECISION ERR,TM
      INTEGER J
C     ..
C     .. Array Arguments ..
      DOUBLE PRECISION ARC(200),B(200),R1(3),R2(3),R3(3),VN(3),VP(3)
C     ..


*****************************************************************

      SUBROUTINE LINTRA(TM,GCLAT,GLON,RKM,ALT,HALT,PLAT,PLON,PRKM,ARC,
     *                  ARAD,ALAT,ALON,ISTOP,NPR,INIT,IER)
C
C     jmh - 11/79  ans fortran 66
C
C     LINTRA traces the geomagnetic field line from a specified
C     starting point to either a specified altitude in either hemisphere
C     or to the apex of the field line. in either case, if the apex
C     is passed, the apex coordinates of the field line are calculated.
C     when points are found on either side of the end point or apex,
C     the final result is calculated by quadratic interpolation.
C
C       Input:
C            TM - time for desired field (years)
C         GCLAT - start point geocentric latitude (deg)
C         GCLON - start point geocentric longitude (deg)
```

```
C           RKM - start point radial distance (km)
C           ALT - start point geodetic altitude (km)
C         ISTOP = -1 - trace to altitude halt in same hemisphere
C         ISTOP = 0 - trace to apex of field line
C         ISTOP = +1 - trace to altitude halt in opposite hemisphere....
C         NPR=1 - return to calling program after each step
C
C       Input, Output:
C          HALT - end point geodetic altitude (km)
C        INIT=1 - set by calling program when returning to lintra after
C                 receiving intermediate results
C             2 - set by lintra when trace is complete
C
C      Output:
C          PLAT - end point geocentric latitude (deg)
C          PLON - end point geocentric longitude (deg)
C          PRKM - end point radial distance (km)
C           ARC - arc length of field line traced (km)
C          ARAD - apex radius of field line (earth radii)
C          ALAT - apex latitude of field line (deg)
C          ALON - apex longitude of field line (deg)
C           IER = 1 - error, number of steps exceeds maxs
C
C     .. Scalar Arguments ..
      DOUBLE PRECISION ALAT,ALON,ALT,ARAD,ARC,GCLAT,GLON,HALT,PLAT,PLON,
     *                 PRKM,RKM,TM
      INTEGER IER,INIT,ISTOP,NPR
C     ..
C     .. Scalars in Common ..
C     ..



**********************************************************************

      SUBROUTINE LOOK(SR,SLAT,SLON,PR,GLAT,GLON,AZ,EL,RANGE)
C
C     jmh - 1/80  ans fortran 66
C
C     LOOK calculates the azimuth, elevation and range from a radar
C     of a specified point.
C
C       Input:
C        SR    - distance of station from center of earth (km)
C        SLAT  - geocentric latitude of station (deg)
C        SLON  - longitude of station (deg)
C        PR    - distance from center of earth of observation point (km)
C        GLAT  - observation point geocentric latitude (deg)
C        GLON  - observation point longitude (deg)
C
C      Output:
C        AZ    - radar azimuth (deg)
C        EL    - radar elevation (deg)
C        RANGE - radar range (km)
C
C     ...calculate "observation-point" earth centered cartesian coords..
C     .. Scalar Arguments ..
      DOUBLE PRECISION AZ,EL,GLAT,GLON,PR,RANGE,SLAT,SLON,SR
C     ..
```

```
************************************************************************

      SUBROUTINE MILMAG(TM,RKM,ST,CT,SPH,CPH,BR,BT,BP,B)
C
C     MILMAG evaluates the geomagnetic field at a point specified by
C     its geocentric coordinates.
C
C     Modified by B. Rideout - Dec. 26, 2002
C     This method is now simply a thin wrapper around geo-cgm code,
C     method igrf.  See geo-cgm.f for details

C
C       Input:
C              TM - time in years for desired field (e.g. 1971.25)
C             RKM - geocentric distance (km)
C           ST,CT - sin and cos of geocentric colatitude
C         SPH,CPH - sin and cos of east longitude
C
C      Output:
C        BR,BT,BP - geocentric field components (gauss)
C               B - magnitude of field (gauss)
C     ..
C     .. Scalar Arguments ..
      DOUBLE PRECISION B,BP,BR,BT,CPH,CT,RKM,SPH,ST,TM
C     ..



************************************************************************

      SUBROUTINE MINV(A,N,D,L,M)
C
C     Inverts general matrix A (overwrites A) using standard
C     gauss-jordan method. The determinant is also calculated. a
C     determinant of zero indicates that the matrix is singular.
C
C      Input:
C        n - order of matrix a
C        l - work vector of length n
C        m - work vector of length n
C
C      Input, Output:
C        a - input matrix, destroyed in computation and replaced by
C            resultant inverse.
C
C      Output:
C        d - resultant determinant
C
C     .. Scalar Arguments ..
      DOUBLE PRECISION D
      INTEGER N
C     ..
C     .. Array Arguments ..
      DOUBLE PRECISION A(*)
      INTEGER L(*),M(*)
C     ..



************************************************************************
```

```
      SUBROUTINE MTRAN3(A)
C
C     jmh - 1/29/80  ans fortran 66
C
C     MTRAN3 calculates the transpose of a 3 x 3 matrix a
C
C       Input, Output:
C           A - input matrix, replaced with transpose.
C
C     .. Array Arguments ..
      DOUBLE PRECISION A(3,3)
C     ..



**********************************************************************

      SUBROUTINE POINT(SR,SLAT,SLON,AZ,EL,RANGE,PR,GLAT,GLON)
C
C     jmh - 1/80  ans fortran 66
C
C     POINT calculates the position of a point defined by the radar
C     line-of sight vector to that point.
C
C     Input:
C       SR    - distance of station from center of earth (km)
C       SLAT  - geocentric latitude of station (deg)
C       SLON  - longitude of station (deg)
C       AZ    - radar azimuth (deg)
C       EL    - radar elevation (deg)
C       RANGE - radar range (km)
C
C     Output:
C       PR    - distance from center of earth of observation point (km)
C       GLAT  - observation point geocentric latitude (deg)
C       GLON  - observation point longitude (deg)
C
C     ...calculate "line-of-sight" station centered cartesian coords...
C     .. Scalar Arguments ..
      DOUBLE PRECISION AZ,EL,GLAT,GLON,PR,RANGE,SLAT,SLON,SR
C     ..



**********************************************************************

      DOUBLE PRECISION FUNCTION RFUN(SR,EL,H)
C
C     jmh - 11/79  ans fortran 66
C
C     RFUN computes the range to an observation point at a specified
C     elevation (EL) and distance (H) above a sphere of radius SR.
C     SR and H should be positive and EL should be in the range
C     0.0 to 90.0.
C
C       Input:
C         SR - radius of sphere (km)
C         EL - elevation (deg)
C          H - distance above sphere (km)
```

```
C
C     .. Scalar Arguments ..
      DOUBLE PRECISION EL,H,SR
C     ..



*****************************************************************

      SUBROUTINE RPCART(SR,SLAT,SLON,AZ,EL,RANGE,RFX,RFY,RFZ,PFX,PFY,
     *                  PFZ)
C
C     jmh - 11/79  ans fortran 66
C
C     RPCART computes the components (RFX,RFY,RFZ) relative to an earth
C     centered cartesian coordinate system of the radar line of sight
C     vector from a radar with coordinates SR (distance from center
C     of earth), SLAT (geocentric latitude) and SLON (longitude). the
C     observation point is specified by AZ (azimuth), EL (elevation) and
C     RANGE (range). the cartesian coordinates of the observation
C     point are returned in (PFX,PFY,PFZ).
C
C
C     Input:
C       SR    - distance of station from center of earth (km)
C       SLAT  - geocentric latitude of station (deg)
C       SLON  - longitude of station (deg)
C       AZ    - radar azimuth (deg)
C       EL    - radar elevation (deg)
C       RANGE - radar range (km)
C
C     Output:
C       RFX,RFY,RFZ - earth centered cartesian coordinate components
C                     of radar line of sight.
C       PFX,PFY,PFZ - earth centered cartesian coordinate components
C                     of observation point.
C
C     .. Scalar Arguments ..
      DOUBLE PRECISION AZ,EL,PFX,PFY,PFZ,RANGE,RFX,RFY,RFZ,SLAT,SLON,SR
C     ..



*****************************************************************

      DOUBLE PRECISION FUNCTION SPROD(A,B)
C
C     jmh - 11/79  ans fortran 66
C
C     SPROD calculates the scalar product of two vectors A and B.
C
C     Input:
C        A - floating point vector of dimension 3
C        B - floating point vector of dimension 3
C
C     .. Array Arguments ..
      DOUBLE PRECISION A(3),B(3)
C     ..
      SPROD = A(1)*B(1) + A(2)*B(2) + A(3)*B(3)
      RETURN
C
```

```
      END

*********************************************************************

      SUBROUTINE STARTR(R1,R2,R3,B,ARC,V,TM)
C
C     Private/Internal subroutine. Part of Apex coordinate computation
C     package. See COORD for public API.
C
C       Input:
C           TM - time in years for desired field (e.g. 1971.25)
C
C       Input, Output:
C          ARC - Altitudes in earth radii (array).
C            V - floating point array.
C
C      Output:
C     R1,R2,R3 - field strength in geocentric directions.
C            B - Magnitude of field (array)
C
C     .. Scalar Arguments ..
      DOUBLE PRECISION TM
C     ..
C     .. Array Arguments ..
      DOUBLE PRECISION ARC(200),B(200),R1(3),R2(3),R3(3),V(3,3)
C     ..



*********************************************************************

      SUBROUTINE UTHM(UTM,IUH,IUM)
C     Converts UTM from the hour and fraction to HH:MM
C     .. Scalar Arguments ..
      DOUBLE PRECISION UTM
      INTEGER IUH,IUM
C     ..
C     .. Intrinsic Functions ..
      INTRINSIC INT,NINT
C     ..
      IUH = INT(UTM)
      IF (IUH.EQ.99) THEN
         IUM = 99
      ELSE
         IUM = NINT((UTM-IUH)*60)
      END IF
      RETURN
      END
C  *********************************************************************
C  =====================================================================
      SUBROUTINE GEOCGM01(ICOR,IYEAR,HI,DAT,PLA,PLO)
C   Version 2001 for GEO-CGM.FOR                              April 2001
C     Apr 11, 2001  GEOLOW is modified to account for interpolation of
C                CGM meridians near equator across the 360/0 boundary
C     AUTHORS:
C     Natalia E. Papitashvili (WDC-B2, Moscow, Russia, now at NSSDC,
C     NASA/Goddard Space Flight Center, Greenbelt, Maryland)
C     Vladimir O. Papitashvili (IZMIRAN, Moscow, Russia, now at SPRL,
C     The University of Michigan, Ann Arbor)
C     Conributions from Boris A. Belov and Vladimir A. Popov (both at
```

```
C     IZMIRAN), as well as from Therese Moretto (DMI, DSRI, now at
C     NASA/GSFC).
C     The original version of this code is described in the brochure by
C    N.A. Tsyganenko, A.V. Usmanov, V.O. Papitashvili, N.E. Papitashvili,
C     and V.A. Popov, Software for computations of geomagnetic field and
C   related coordinate systems, Soviet Geophys. Committ., Moscow, 58 pp.,
C     1987. A number of subroutines from the revised GEOPACK-96 software
C     package developed by Nikolai A. Tsyganenko and Mauricio Peredo are
C     utilized in this code with some modifications (see full version of
C    GEOPACK-96 on http://www-spof.gsfc.nasa.gov/Modeling/geopack.html).
C     This code consists of the main subroutine GEOCGM99, five functions
C  (OVL_ANG, CGMGLA, CGMGLO, DFRIDR, and AZM_ANG), eigth new and revised
C     subroutines from the above-mentioned brochure (MLTUT, MFC, FTPRNT,
C     GEOLOW, CORGEO, GEOCOR, SHAG, and RIGHT), and 9 subroutines from
C    GEOPACK-96 (IGRF, SPHCAR, BSPCAR, GEOMAG, MAGSM, SMGSM, RECALC, SUN)
C   ======================================================================
C     Input parameters:
C     icor = +1    geo to cgm
C            -1    cgm to geo
C     iyr  = year
C     hi   = altitude
C     slar = geocentric latitude
C     slor = geocentric longitude (east +)
C     These two pairs can be either input or output parameters
C     clar = cgm latitude
C     clor = cgm longitude (east +)
C     Output parameters:
C     Array DAT(11,4) consists of 11 parameters (slar, slor, clar, clor,
C     rbm, btr, bfr, brr, ovl, azm, utm) organized for the start point
C     (*,1), its conjugate point (*,2), then for the footprints at 1-Re
C     of the start (*,3) and conjugate (*,4) points
C     Description of parameters used in the subroutine:
C     slac = conjugate geocentric latitude
C     sloc = conjugate geocentric longitude
C     slaf = footprint geocentric latitude
C     slof = footprint geocentric longitude
C     rbm  = apex of the magnetic field line in Re (Re=6371.2 km)
C            (this parameter approximately equals the McIlwain L-value)
C     btr  = IGRF Magnetic field H (nT)
C     bfr  = IGRF Magnetic field D (deg)
C     brr  = IGRF Magnetic field Z (nT)
C     ovl  = oval_angle as the azimuth to "magnetic north":
C                 + east in Northern Hemisphere
C                 + west in Southern Hemisphere
C     azm  = meridian_angle as the azimuth to the CGM pole:
C                 + east in Northern Hemisphere
C                 + west in Southern Hemisphere
C     utm  = magnetic local time (MLT) midnight in UT hours
C     pla  = array of geocentric latitude and

C     plo  = array of geocentric longitudes for the CGM poles
C            in the Northern and Southern hemispheres at a given
C            altitude (indices 1 and 2) and then at the Earth's
C            surface - 1-Re or zero altitude - (indices 3 and 4)
C     dla  = dipole latitude
C     dlo  = dipole longitude
C   ======================================================================
C     Year (for example, as for Epoch 1995.0 - no fraction of the year)
C     .. Scalar Arguments ..
      DOUBLE PRECISION HI
      INTEGER ICOR,IYEAR
```

```
C     ..
C     .. Array Arguments ..
      DOUBLE PRECISION DAT(11,4),PLA(4),PLO(4)
C     ..



*****************************************************************

      DOUBLE PRECISION FUNCTION OVL_ANG(SLA,SLO,CLA,CLO,RR)
C   This function returns an estimate at the given location of the angle
C     (oval_angle) between the directions (tangents) along the constant
C     CGM and geographic latitudes by utilizing the function DFRIDR from
C     Numerical Recipes for FORTRAN.
C   This angle can be taken as the azimuth to the local "magnetic" north
C   (south) if the eastward (westward) tangent to the local CGM latitude
C     points south (north) from the local geographic latitude.
C  Written by Therese Moretto in August 1994 (revised by V. Papitashvili
C     in January 1999).
C   Ignore points which nearly coincide with the geographic or CGM poles
C    within 0.01 degree in latitudes; this also takes care if SLA or CLA
C     are dummy values (e.g., 999.99)
C     .. Scalar Arguments ..
      DOUBLE PRECISION CLA,CLO,RR,SLA,SLO
C     ..



*****************************************************************

      DOUBLE PRECISION FUNCTION CGMGLA(CLON)
C     This function returns the geocentric latitude as a function of CGM
C     longitude with the CGM latitude held in common block CGMGEO.
C     Essentially this function just calls the subroutine CORGEO.
C     .. Scalar Arguments ..
      DOUBLE PRECISION CLON
C     ..



*****************************************************************

      DOUBLE PRECISION FUNCTION CGMGLO(CLON)
C     Same as the function CGMGLA but this returns the geocentric
C    longitude. If cr360 is true, geolon+360 deg is returned when geolon
C     is less than 90 deg. If cr0 is true, geolon-360 deg is returned
C     when geolon is larger than 270 degrees.
C     .. Scalar Arguments ..
      DOUBLE PRECISION CLON
C     ..



*****************************************************************

      DOUBLE PRECISION FUNCTION AZM_ANG(SLA,SLO,CLA,PLA,PLO)
C     Computation of an angle between the north geographic meridian and
C     direction to the North (South) CGM pole: positive azimuth is
C     measured East (West) from geographic meridian, i.e., the angle is
C     measured between the great-circle arc directions to the geographic
C     and CGM poles. In this case the geomagnetic field components in
```

```
C     XYZ (NEV) system can be converted into the CGM system in both
C     hemispheres as:
C                           XM = X COS(alf) + Y SIN(alf)
C                           YM =-X SIN(alf) + Y COS(alf)
C   Written by V. O. Papitashvili in mid-1980s; revised in February 1999
C   Ignore points which nearly coincide with the geographic or CGM poles
C    within 0.01 degree in latitudes; this also takes care if SLA or CLA
C     are dummy values (e.g., 999.99)
C     .. Scalar Arguments ..
      DOUBLE PRECISION CLA,PLA,PLO,SLA,SLO
C     ..


**********************************************************************

      SUBROUTINE MLTUT(SLA,SLO,CLA,PLA,PLO,UT)
C     Calculates the MLT midnight in UT hours
C     Definition of the MLT midnight (MLTMN) here is different from the
C     approach described elsewhere. This definition does not take into
C    account the geomagnetic meridian of the subsolar point which causes
C    seasonal variations of the MLTMN in UT time. The latter approach is
C     perfectly applicable to the dipole or eccentric dipole magnetic
C    coordinates but it fails with the CGM coordinates because there are
C     forbidden areas near the geomagnetic equator where CGM coordinates
C     cannot be calculated by definition [e.g., Gustafsson et al., JATP,
C     54, 1609, 1992].
C  In this code the MLT midnight is defined as location of a given point
C    on (or above) the Earth's surface strictly behind the North (South)
C     CGM pole in such the Sun, the pole, and the point are lined up.
C     This approach was originally proposed and coded by Boris Belov
C     sometime in the beginning of 1980s; here it is slightly edited by
C     Vladimir Papitashvili in February 1999.
C   Ignore points which nearly coincide with the geographic or CGM poles
C    within 0.01 degree in latitudes; this also takes care if SLA or CLA
C     are dummy values (e.g., 999.99)
C     .. Scalar Arguments ..
      DOUBLE PRECISION CLA,PLA,PLO,SLA,SLO,UT
C     ..


**********************************************************************

      SUBROUTINE MFC(SLA,SLO,R,H,D,Z)
C     Computation of the IGRF magnetic field components
C     Extracted as a subroutine from the earlier version of GEO-CGM.FOR
C     V. Papitashvili, February 1999
C     This takes care if SLA or CLA are dummy values (e.g., 999.99)
C     .. Scalar Arguments ..
      DOUBLE PRECISION D,H,R,SLA,SLO,Z
C     ..


**********************************************************************

      SUBROUTINE FTPRNT(RH,SLA,SLO,CLA,CLO,ACLA,ACLO,SLAF,SLOF,RF)
C     Calculation of the magnetic field line footprint at the Earth's
C     (or any higher) surface.
C   Extracted as a subroutine from the earlier version of GEO-CGM.FOR by
```

```
C   V. Papitashvili in February 1999 but then the subroutine was revised
C    to obtain the Altitude Adjusted CGM coordinates. The AACGM approach
C     is proposed by Kile Baker of the JHU/APL, see their World Wide Web
C     site http://sd-www.jhuapl.edu/RADAR/AACGM/ for details.
C     If RF = 1-Re (i.e., at the Earth's surface), then the footprint
C     location is defined as the Altitude Adjusted (AA) CGM coordinates
C     for a given point (ACLA, ACLO).
C     If RF = 1.xx Re (i.e., at any altitude above or below the starting
C     point), then the conjunction between these two points can be found
C     along the field line.
C     This takes care if SLA or CLA are dummy values (e.g., 999.99)
C     .. Scalar Arguments ..
      DOUBLE PRECISION ACLA,ACLO,CLA,CLO,RF,RH,SLA,SLAF,SLO,SLOF
C     ..



*****************************************************************

      SUBROUTINE GEOLOW(SLAR,SLOR,RH,CLAR,CLOR,RBM,SLAC,SLOC)
C     Calculates CGM coordinates from geocentric ones at low latitudes
C    where the DGRF/IGRF magnetic field lines may never cross the dipole
C     equatorial plane and, therefore, the definition of CGM coordinates
C     becomes invalid.
C     The code is written by Natalia and Vladimir Papitashvili as a part
C   of the earlier versions of GEO-CGM.FOR; extracted as a subroutine by
C     V. Papitashvili in February 1999.
C     Apr 11, 2001  GEOLOW is modified to account for interpolation of
C                CGM meridians near equator across the 360/0 boundary
C     See the paper by  Gustafsson, G., N. E. Papitashvili, and V. O.
C    Papitashvili, A revised corrected geomagnetic coordinate system for
C     Epochs 1985 and 1990 [J. Atmos. Terr. Phys., 54, 1609-1631, 1992]
C     for detailed description of the B-min approach utilized here.
C     This takes care if SLA is a dummy value (e.g., 999.99)
C     .. Scalar Arguments ..
      DOUBLE PRECISION CLAR,CLOR,RBM,RH,SLAC,SLAR,SLOC,SLOR
C     ..



*****************************************************************

      SUBROUTINE CORGEO(SLA,SLO,RH,DLA,DLO,CLA,CLO,PMI)
C     Calculates geocentric coordinates from corrected geomagnetic ones.
C     The code is written by Vladimir Popov and Vladimir Papitashvili
C     in mid-1980s; revised by V. Papitashvili in February 1999
C     This takes care if CLA is a dummy value (e.g., 999.99)
C     .. Scalar Arguments ..
      DOUBLE PRECISION CLA,CLO,DLA,DLO,PMI,RH,SLA,SLO
C     ..



*****************************************************************

      SUBROUTINE GEOCOR(SLA,SLO,RH,DLA,DLO,CLA,CLO,PMI)
C     Calculates corrected geomagnetic coordinates from geocentric ones
C     The code is written by Vladimir Popov and Vladimir Papitashvili
C     in mid-1980s; revised by V. Papitashvili in February 1999
C     This takes care if SLA is a dummy value (e.g., 999.99)
C     .. Scalar Arguments ..
```

```
      DOUBLE PRECISION CLA,CLO,DLA,DLO,PMI,RH,SLA,SLO
C     ..


*****************************************************************

      SUBROUTINE SHAG(X,Y,Z,DS)
C     Similar to SUBR STEP from GEOPACK-1996 but SHAG takes into account
C     only internal sources
C     The code is re-written from Tsyganenko's subroutine STEP by
C     Natalia and Vladimir Papitashvili in mid-1980s
C     .. Scalar Arguments ..
      DOUBLE PRECISION DS,X,Y,Z
C     ..


*****************************************************************

      SUBROUTINE RIGHT(X,Y,Z,R1,R2,R3)
C   Similar to SUBR RHAND from GEOPACK-1996 but RIGHT takes into account
C     only internal sources
C     The code is re-written from Tsyganenko's subroutine RHAND
C     by Natalia and Vladimir Papitashvili in mid-1980s
C     .. Scalar Arguments ..
      DOUBLE PRECISION R1,R2,R3,X,Y,Z
C     ..


*****************************************************************

      SUBROUTINE IGRF(IY,NM,R,T,F,BR,BT,BF)
c     Jan 20, 2001: Subroutine IGRF is modified by V. Papitashvili - SHA
c     coefficients for IGRF-2000, and SV 2000-2005 are added (note that
c     IGRF-1995 has not been changed to DGRF-1995 this time
c     (see http://www.ngdc.noaa.gov/IAGA/wg8/igrf2000.html)
c     Aug 26, 1997: Subroutine IGRF is modified by V. Papitashvili - SHA
c     coefficients for DGRF-1990, IGRF-1995, and SV 1995-2000 are added
c     (EOS, v.77, No.16, p.153, April 16, 1996)
c   Feb 03, 1995: Modified by Vladimir Papitashvili (SPRL, University of
c     Michigan) to accept dates between 1945 and 2000
C  MODIFIED TO ACCEPT DATES BETWEEN 1965 AND 2000; COEFFICIENTS FOR IGRF
C     1985 HAVE BEEN REPLACED WITH DGRF1985 COEFFICIENTS [EOS TRANS. AGU
C     APRIL 21, 1992, C  P. 182]. ALSO, THE CODE IS MODIFIED TO ACCEPT
C    DATES BEYOND 1990, AND TO USE LINEAR EXTRAPOLATION BETWEEN 1990 AND
C     2000 BASED ON THE IGRF COEFFICIENTS FROM THE SAME EOS ARTICLE
C   Modified by Mauricio Peredo, Hughes STX at NASA/GSFC, September 1992
C     CALCULATES COMPONENTS OF MAIN GEOMAGNETIC FIELD IN SPHERICAL
C     GEOCENTRIC COORDINATE SYSTEM BY USING THIRD GENERATION IGRF MODEL
C     (J. GEOMAG. GEOELECTR. V.34, P.313-315, 1982; GEOMAGNETISM AND
C     AERONOMY V.26, P.523-525, 1986).
C     UPDATING OF COEFFICIENTS TO A GIVEN EPOCH IS MADE DURING THE FIRST
C     CALL AND AFTER EVERY CHANGE OF PARAMETER IY
C     ---INPUT PARAMETERS:
C     IY - YEAR NUMBER (FROM 1945 UP TO 1990)
C  NM - MAXIMAL ORDER OF HARMONICS TAKEN INTO ACCOUNT (NOT MORE THAN 10)
C   R,T,F - SPHERICAL COORDINATES OF THE POINT (R IN UNITS RE=6371.2 KM,
C     COLATITUDE T AND LONGITUDE F IN RADIANS)
C     ---OUTPUT PARAMETERS:
```

```
C     BR,BT,BF - SPHERICAL COMPONENTS OF MAIN GEOMAGNETIC FIELD (in nT)
C    AUTHOR: NIKOLAI A. TSYGANENKO, INSTITUTE OF PHYSICS, ST.-PETERSBURG
C      STATE UNIVERSITY, STARY PETERGOF 198904, ST.-PETERSBURG, RUSSIA
C      (now the NASA Goddard Space Fligth Center, Greenbelt, Maryland)
      IMPLICIT NONE
C     G0, G1, and H1 are used in SUBROUTINE DIP to calculate geodipole's
C     moment for a given year
C     .. Scalar Arguments ..
      DOUBLE PRECISION BF,BR,BT,F,R,T
      INTEGER IY,NM
C     ..


**********************************************************************

      SUBROUTINE RECALC(IYR,IDAY,IHOUR,MIN,ISEC)
C     THIS IS A MODIFIED VERSION OF THE SUBROUTINE RECOMP WRITTEN BY
C     N. A. TSYGANENKO. SINCE I WANT TO USE IT IN PLACE OF SUBROUTINE
C     RECALC, I HAVE RENAMED THIS ROUTINE RECALC AND ELIMINATED THE
C     ORIGINAL RECALC FROM THIS VERSION OF THE  PACKAGE.
C    THIS WAY ALL ORIGINAL CALLS TO RECALC WILL CONTINUE TO WORK WITHOUT
C     HAVING TO CHANGE THEM TO CALLS TO RECOMP.
C     AN ALTERNATIVE VERSION OF THE SUBROUTINE RECALC FROM THE GEOPACK
C     PACKAGE BASED ON A DIFFERENT APPROACH TO DERIVATION OF ROTATION
C     MATRIX ELEMENTS
C     THIS SUBROUTINE WORKS BY 20% FASTER THAN RECALC AND IS EASIER TO
C     UNDERSTAND
C     ##################################################
C     #  WRITTEN BY  N.A. TSYGANENKO ON DECEMBER 1, 1991  #
C     ##################################################
C     Modified by Mauricio Peredo, Hughes STX at NASA/GSFC Code 695,
C     September 1992
c    Modified to accept years up to 2005 (V. Papitashvili, January 2001)
c  Modified to accept dates up to year 2000 and updated IGRF coeficients
c     from 1945 (updated by V. Papitashvili, February 1995)
C     OTHER SUBROUTINES CALLED BY THIS ONE: SUN
C     IYR = YEAR NUMBER (FOUR DIGITS)
C     IDAY = DAY OF YEAR (DAY 1 = JAN 1)
C     IHOUR = HOUR OF DAY (00 TO 23)
C     MIN = MINUTE OF HOUR (00 TO 59)
C     ISEC = SECONDS OF DAY(00 TO 59)
      IMPLICIT NONE
C     .. Scalar Arguments ..
      INTEGER IDAY,IHOUR,ISEC,IYR,MIN
C     ..


**********************************************************************

      SUBROUTINE SUN(IYR,IDAY,IHOUR,MIN,ISEC,GST,SLONG,SRASN,SDEC)
C    CALCULATES FOUR QUANTITIES NECESSARY FOR COORDINATE TRANSFORMATIONS
C     WHICH DEPEND ON SUN POSITION (AND, HENCE, ON UNIVERSAL TIME AND
C     SEASON)
C     ---INPUT PARAMETERS:
C     IYR,IDAY,IHOUR,MIN,ISEC - YEAR, DAY, AND UNIVERSAL TIME IN HOURS,
C     MINUTES, AND SECONDS  (IDAY=1 CORRESPONDS TO JANUARY 1).
C     ---OUTPUT PARAMETERS:
C   GST - GREENWICH MEAN SIDEREAL TIME, SLONG - LONGITUDE ALONG ECLIPTIC
C     SRASN - RIGHT ASCENSION,  SDEC - DECLINATION  OF THE SUN (RADIANS)
```

```
C     THIS SUBROUTINE HAS BEEN COMPILED FROM:
C     RUSSELL C.T., COSM.ELECTRODYN., 1971, V.2,PP.184-196.
C     AUTHOR: Gilbert D. Mead
      IMPLICIT NONE
C     .. Scalar Arguments ..
      DOUBLE PRECISION GST,SDEC,SLONG,SRASN
      INTEGER IDAY,IHOUR,ISEC,IYR,MIN
C     ..



**********************************************************************

      SUBROUTINE SPHCAR(R,TETA,PHI,X,Y,Z,J)
C     CONVERTS SPHERICAL COORDS INTO CARTESIAN ONES AND VICA VERSA
C     (TETA AND PHI IN RADIANS).
C                  J>0            J
**********************************************************************

      SUBROUTINE BSPCAR(TETA,PHI,BR,BTET,BPHI,BX,BY,BZ)
C     CALCULATES CARTESIAN FIELD COMPONENTS FROM SPHERICAL ONES
C     -----INPUT:   TETA,PHI - SPHERICAL ANGLES OF THE POINT IN RADIANS
C              BR,BTET,BPHI -  SPHERICAL COMPONENTS OF THE FIELD
C     -----OUTPUT:  BX,BY,BZ - CARTESIAN COMPONENTS OF THE FIELD
C    AUTHOR: NIKOLAI A. TSYGANENKO, INSTITUTE OF PHYSICS, ST.-PETERSBURG
C      STATE UNIVERSITY, STARY PETERGOF 198904, ST.-PETERSBURG, RUSSIA
C      (now the NASA Goddard Space Fligth Center, Greenbelt, Maryland)
      IMPLICIT NONE
C     .. Scalar Arguments ..
      DOUBLE PRECISION BPHI,BR,BTET,BX,BY,BZ,PHI,TETA
C     ..



**********************************************************************

      SUBROUTINE GEOMAG(XGEO,YGEO,ZGEO,XMAG,YMAG,ZMAG,J,IYR)
C   CONVERTS GEOCENTRIC (GEO) TO DIPOLE (MAG) COORDINATES OR VICA VERSA.
C     IYR IS YEAR NUMBER (FOUR DIGITS).
C                           J>0                J
**********************************************************************

      SUBROUTINE MAGSM(XMAG,YMAG,ZMAG,XSM,YSM,ZSM,J)
C CONVERTS DIPOLE (MAG) TO SOLAR MAGNETIC (SM) COORDINATES OR VICA VERSA
C                    J>0              J
**********************************************************************

      SUBROUTINE SMGSM(XSM,YSM,ZSM,XGSM,YGSM,ZGSM,J)
C CONVERTS SOLAR MAGNETIC (SM) TO SOLAR MAGNETOSPHERIC (GSM) COORDINATES
C     OR VICA VERSA.
C                   J>0                 J
**********************************************************************

      DOUBLE PRECISION FUNCTION TNF(TI,TE,NE,NHP,NO,NH,NN2,NO2,NHE,IER)
C
C     TNF calculates the ion temperature (tn) given the electron and
C     neutral temperatures (TE, TN) and the electron, o+, h+, o, h,
C     n2, o2 and he concentrations (NE, NOP, NHP, NO, NH, NN2, NO2,
C     NHE). o+ and h+ ions are assumed to be the only ions present.
C     only coulomb collisions and ion-neutral polarization and
C     charge-exchange interactions are considered in balancing
```

```
C     the ion gas energy source and sink terms.  the solution for
C     tn is an iterative procedure in which TI is the initial value
C     of tn for the first iteration. all concentrations are in
C     units of cm**-3.
C
C       Input:
C          TI - Ion Temperature (K)
C          TE - Electron Temperature
C          NE - Electron concentration (cm**-3)
C         NHP - H Ion concentration
C          NO - O Ion concentration
C         NN2 - N2 concentration
C         NO2 - O2 concentration
C         NHE - HE concentration
C
C      Output:
C         IER - If (IER.NE.0) an error has occurred.
C
C     .. Scalar Arguments ..
      DOUBLE PRECISION NE,NH,NHE,NHP,NN2,NO,NO2,TE,TI
      INTEGER IER
C     ..



**********************************************************************

      SUBROUTINE VADD(A,B,C)
C
C     jmh - 11/79  ans fortran 66
C
C     VADD calculates the sum of two vectors A and B, C = A + B.
C
C       Input:
C          A - floating point vector of dimension 3.
C          B - floating point vector of dimension 3.
C
C      Output:
C          C - floating point vector of dimension 3.
C
C     .. Array Arguments ..
      DOUBLE PRECISION A(3),B(3),C(3)
C     ..
      C(1) = A(1) + B(1)
      C(2) = A(2) + B(2)
      C(3) = A(3) + B(3)
      RETURN
C
      END


**********************************************************************

      SUBROUTINE VCTCNV(FX,FY,FZ,X,Y,Z,FR,FT,FP,R,THETA,PHI,IMODE)
C
C     jmh - 11/79  ans fortran 66
C
C     VCTCNV converts between the cartesian and spherical coordinate
C     representations of a vector field f. (FX,FY,FZ) are the
C     components of the field at (X,Y,Z). (FR,FT,FP) are the
C     components of the field at (R,THETA,PHI) in the directions of
```

```
C     increasing R, increasing THETA and increasing PHI. if IMODE=1,
C     (FX,FY,FZ,X,Y,Z) -> (FR,FT,FP,R,THETA,PHI). if IMODE=2,
C     (FR,FT,FP,R,THETA,PHI) -> (FX,FY,FZ,X,Y,Z). THETA and PHI are
C     in degrees.
C
C       Input:
C         IMODE - 1 cartesian to spherical
C                 2 spherical to cartesian
C
C     Input, output:
C      FX,FY,FZ - cartesian vector field components
C         X,Y,Z - cartesian vector field coordinates
C      FR,FT,FP - spherical vector field components
C   R,THETA,PHI - spherical vector field coordinates
C
C     .. Scalar Arguments ..
      DOUBLE PRECISION FP,FR,FT,FX,FY,FZ,PHI,R,THETA,X,Y,Z
      INTEGER IMODE
C     ..



**********************************************************************

      DOUBLE PRECISION FUNCTION VMAG(A)
C
C     jmh - 1/80  ans fortran 66
C
C     VMAG calculates the magnitude of a vector A
C
C     Input:
C        A - floating point vector of dimension 3
C
C     .. Array Arguments ..
      DOUBLE PRECISION A(3)
C     ..
C     .. Intrinsic Functions ..
      INTRINSIC DSQRT
C     ..
      VMAG = DSQRT(A(1)*A(1)+A(2)*A(2)+A(3)*A(3))
      RETURN
C
      END


**********************************************************************

      SUBROUTINE VSUB(A,B,C)
C
C     jmh - 11/79  ans fortran 66
C
C     VSUB calculates the difference of two vectors A and B, C = A - B.
C
C     Input:
C        A - floating point vector of dimension 3
C        B - floating point vector of dimension 3
C
C     Output:
C        C - floating point vector of dimension 3
C
C     .. Array Arguments ..
```

```
      DOUBLE PRECISION A(3),B(3),C(3)
C     ..
      C(1) = A(1) - B(1)
      C(2) = A(2) - B(2)
      C(3) = A(3) - B(3)
      RETURN
C
      END


**********************************************************************

      SUBROUTINE CALNDR(YEAR,DAYNO,DAY,MONTH,IER)
C
C     CALNDR returns DAY, MONTH and IER. THE FOLLOWING VARIABLES APPEAR
C     IN THE CALLING SEQUENCE (all INTEGERS).
C
C          YEAR - YEAR (1977)
C         DAYNO - DAY OF THE YEAR (60)
C           DAY - DAY OF THE MONTH (1)
C         MONTH - MONTH NUMBER (3)
C           IER - ERROR INDICATOR. IER IS RETURNED BY ALL ROUTINES IN
C                 THE PACKAGE. IER=0 IF NO ERRORS ARE DETECTED. IER=1
C                 IF AN ERROR IS DETECTED.
C
C     .. Scalar Arguments ..
      INTEGER DAY,DAYNO,IER,MONTH,YEAR
C     ..



**********************************************************************

      SUBROUTINE DATER(DATE,NCHAR,DAY,MONTH,YEAR,IER)
C
C     SUBROUTINES DATER, MONAME, MONUM, WKNAME, IDAY, CALNDR, JDAY,
C     JDATER, IZLR, IDMYCK AND DATES COMPRISE A COMPREHENSIVE DATE
C     MANIPULATION
C     PACKAGE. THE FOLLOWING VARIABLES APPEAR IN THE CALLING SEQUENCE
C     OF ONE OR MORE SUBROUTINES. ALL ARE TYPED INTEGER. THE VALUE
C     CORRESPONDING TO MARCH 1, 1977 IS SHOWN IN PARENTHESES.
C
C           DAY - DAY OF THE MONTH (1)
C         MONTH - MONTH NUMBER (3)
C          YEAR - YEAR (1977)
C         DAYNO - DAY OF THE YEAR (60)
C        JDAYNO - JULIAN DAY NUMBER (2443204)
C          WDAY - WEEKDAY NUMBER (3)
C          DATE - DATE AS A STRING OF ALPHANUMERIC CHARS, 3 CHARS/WORD.
C                 WHEN AN INPUT VARIABLE, DATE MAY BE IN ANY REASONABLE
C                 FORMAT, E.G. MARCH 1 1977, 3/1/77, ETC. IF EXPRESSED
C                 AS THREE NUMERIC FIELDS, ORDER IS PRESUMED TO BE
C                 MONTH, DAY, YEAR. WHEN AN OUTPUT VARIABLE, THE FORMAT
C                 IS DETERMINED BY IOPT, AND SIX WORDS SHOULD BE
C                 RESERVED IN THE CALLING PROGRAM.
C
C          MSTR - MONTH, AS A STRING OF UPPER CASE ALPHABETIC
C                 CHARACTERS, 3 CHARACTERS/WORD. THREE WORDS SHOULD BE
C                 RESERVED IN THE CALLING PROGRAM. (MARCH)
C          WSTR - DAY OF THE WEEK, AS A STRING OF UPPER CASE ALPHABETIC
C                 CHARACTERS, 3 CHARACTERS/WORD. THREE WORDS SHOULD BE
```

```
C                 RESERVED IN THE CALLING PROGRAM. (TUESDAY)
C          IOPT - FORMAT INDICATOR WHEN DATE IS AN OUTPUT VARIABLE
C                  1 - 03/01/77
C                  3 - 01,MAR,1977
C                  4 - 1 MARCH, 1977
C                  5 - MARCH 1, 1977
C         NCHAR - NUMBER OF CHARACTERS TO BE SCANNED IN AN INPUT STRING.
C           IER - ERROR INDICATOR. IER IS RETURNED BY ALL ROUTINES IN
C                 THE PACKAGE. IER=0 IF NO ERRORS ARE DETECTED. IER=1
C                 IF AN ERROR IS DETECTED.
C
C     .. Scalar Arguments ..
      INTEGER DAY,IER,MONTH,NCHAR,YEAR
      CHARACTER*(*) DATE
C     ..


**********************************************************************

      SUBROUTINE DATES(DAY,MONTH,YEAR,IOPT,IER,DATE)
C
C     Returns DATE String in various formats given DAY, MONTH, YEAR
C     and IOPT (which specifies the desired format).
C
C     The following variables appear in the calling sequence
C
C       Input:
C           DAY - DAY OF THE MONTH (1)
C         MONTH - MONTH NUMBER (3)
C          YEAR - YEAR (1977)
C          IOPT - FORMAT INDICATOR WHEN DATE IS AN OUTPUT VARIABLE
C                  1 - 01/01/97
C                  2 - 01,JAN,1997
C                  3 - 1 JANUARY, 1997
C                  4 - JANUARY 1 1997
C                  5 - 01JAN97
C
C       Output:
C           IER - ERROR INDICATOR. IER IS RETURNED BY ALL ROUTINES IN
C                 THE PACKAGE. IER=0 IF NO ERRORS ARE DETECTED. IER=1
C                 IF AN ERROR IS DETECTED.
C          DATE - DATE AS A STRING OF ALPHANUMERIC CHARACTERS, 3
C                 CHARACTERS/WORD. WHEN AN INPUT VARIABLE, DATE MAY BE
C                 IN ANY REASONABLE FORMAT, E.G. MARCH 1 1977, 3/1/77,
C                 ETC. IF EXPRESSED AS THREE NUMERIC FIELDS, ORDER IS
C                 PRESUMED TO BE MONTH, DAY, YEAR. WHEN AN OUTPUT
C                 VARIABLE, THE FORMAT IS DETERMINED BY IOPT, AND SIX
C                 WORDS SHOULD BE RESERVED IN THE CALLING PROGRAM.
C
C     .. Scalar Arguments ..
      INTEGER DAY,IER,IOPT,MONTH,YEAR
      CHARACTER*(*) DATE
C     ..


**********************************************************************

      SUBROUTINE IDAY(DAY,MONTH,YEAR,DAYNO,IER)
C
```

```
C     Returns Day-of-year from DAY, MONTH, YEAR.
C
C      Input:
C        IDBFIL - Madrigal file number (see EXDCON)
C           DAY - Day of month (1-31)
C         MONTH - Month of year (1-12)
C          YEAR - Year (e.g. 1977)
C
C      Output:
C         DAYNO - Day-of-year (1-356)
C           IER - If (IER.NE.0) an error has occurred.
C
C     .. Scalar Arguments ..
      INTEGER DAY,DAYNO,IER,MONTH,YEAR
C     ..



**********************************************************************

      INTEGER FUNCTION IDMYCK(DAY,MONTH,YEAR)
C
C     Returns 1 if DAY, MONTH and YEAR are legal values, 0 otherwise.
C
C      Input:
C           DAY - Day of month (1-31)
C         MONTH - Month of year (1-12)
C          YEAR - Year (e.g. 1977)
C
C     .. Scalar Arguments ..
      INTEGER DAY,MONTH,YEAR
C     ..



**********************************************************************

      SUBROUTINE IZLR(DAY,MONTH,YEAR,WDAY,IER)
C
C     Returns day-of-the-week value from DAY, MONTH and YEAR.
C
C      Input:
C           DAY - Day of month (1-31)
C         MONTH - Month of year (1-12)
C          YEAR - Year (e.g. 1977)
C
C      Output:
C          WDAY - Day of week (1-7)
C           IER - If (IER.NE.0) an error has occurred.
C
C     .. Scalar Arguments ..
      INTEGER DAY,IER,MONTH,WDAY,YEAR
C     ..



**********************************************************************

      SUBROUTINE JDATER(JDAYNO,DAY,MONTH,YEAR,IER)
C
C     Returns DAY, MONTH, YEAR from Julian day (See JDAY for inverse).
```

```
C
C      Input:
C        JDAYNO - Julian day (e.g. 2447892)
C
C      Output:
C           DAY - Day of month (1-31)
C         MONTH - Month of year (1-12)
C          YEAR - Year (e.g. 1977)
C           IER - If (IER.NE.0) an error has occurred.
C
C     .. Scalar Arguments ..
      INTEGER DAY,IER,JDAYNO,MONTH,YEAR
C     ..


*****************************************************************

      SUBROUTINE JDAY(DAY,MONTH,YEAR,JDAYNO,IER)
C
C     Returns Julian day from DAY, MONTH, YEAR (See JDATER for
C     inverse).
C
C      Input:
C           DAY - Day of month (1-31)
C         MONTH - Month of year (1-12)
C          YEAR - Year (e.g. 1977)
C
C      Output:
C        JDAYNO - Julian day (e.g. 2447892)
C           IER - If (IER.NE.0) an error has occurred.
C
C     .. Scalar Arguments ..
      INTEGER DAY,IER,JDAYNO,MONTH,YEAR
C     ..


*****************************************************************

      SUBROUTINE MONAME(MONTH,MSTR,IER)
C
C     Returns Month string from Month integer.
C
C      Input:
C        MONTH - Month of year (1-12)
C
C      Output:
C         MSTR - Month string (e.g. 'JANUARY')
C          IER - If (IER.NE.0) an error has occurred.
C
C     .. Scalar Arguments ..
      INTEGER IER,MONTH
      CHARACTER*(*) MSTR
C     ..


*****************************************************************

      SUBROUTINE MONUM(MSTR,MONTH,IER)
```

```
C
C     Returns Month integer from Month string (case insignificant).
C
C      Input:
C         MSTR - Month string (e.g. 'January')
C
C      Output:
C        MONTH - Month of year (1-12)
C          IER - If (IER.NE.0) an error has occurred.
C
C     .. Scalar Arguments ..
      INTEGER IER,MONTH
      CHARACTER*(*) MSTR
C     ..



**********************************************************************

      SUBROUTINE WKNAME(WDAY,IER,WSTR)
C
C     Returns day of the week string from day of the week number.
C
C       Input:
C         WDAY - week day number (1-7)
C
C      Output:
C         IER  - If (IER.NE.0) an error has occurred.
C         WSTR - day of the week string (e.g. 'SUNDAY').
C
C     .. Scalar Arguments ..
      INTEGER IER,WDAY
      CHARACTER*(*) WSTR
C     ..
```


	Madrigal Fortran API	Doc home	Madrigal home

Previous: C API **Up:** Madrigal developer's guide **Next:** Tcl API

	Madrigal Tcl API	Doc home	Madrigal home

Previous: Fortran API **Up:** Madrigal developer's guide **Next:** Madrigal file format

Madrigal Tcl API

Madrigal extensions to Tcl (madtclsh)

Madtclsh is an extended tcl interpreter which adds direct support for manipulating Cedar database files and their contents. This executable is located in *madroot*/bin. There are eight new commands:

- mad
- cedarCode
- getKey
- isr_point
- isr_look
- isr_geodetic2geocentric
- isr_geocentric2geodetic
- isr_dircos

The first two commands create new tcl objects which can be used to manipulate Cedar files (mad) and Cedar metacode information (cedarCode). The remaining commands determine the Madrigal key given a time and date (getKey) and provide basic geometry and geographic coordinate functions (isr_point, isr_look, isr_geodetic2geocentric, isr_geocentric2geodetic and isr_dircos).

The script shown below, summarizeCedarFile, illustrates the use of madtclsh. It begins by creating a new madrec object (mad madin). More than one madrec object can be created. For example, a script to translate a Cedar file from Madrigal to Cedar Ascii format would require two. The mad command creates a new tcl command - $madin in the example. Typically the first thing done with the new command is to associate it with a Cedar file ($madin open 1 $infile). The file can then be read using the getNextRecord subcommand ($madin getNextRecord). Once a record has read, numerous subcommands are available to extract information from the record (e.g. $madin get startTime). When the file is no longer needed, it can be disassociated from the madrec object ($madin close). Finally, when the madrec object is no longer needed, it can be destroyed ($madin destroy).

```
#!/bin/sh
# The madtclsh path is longer than 32 characters. So, we take advantage
# of the fact that a backslash continues a comment line in tcl \
exec /opt/madrigal/bin/madtclsh "$0" ${1+"$@"}

# summarizeCedarFile prints a one line per record summary of a CEDAR
# file.  The file may be any of the 5 supported CEDAR formats (Madrigal,
# Blocked Binary, Cbf, Unblocked Binary or ASCII"), and may include any
# mixture of prologue, header and data records. The format of the file is
# determined automatically.

# Usage: printCedarRecords filename firstRecord lastRecord

# Get parameter codes
cedarCode cedarCode

# Get number of parameters
set nargs $argc
if {$nargs != 1} {
    puts {Usage: summarizeCedarFile filename}
    exit
}

# Get file name from the argument list
set infile [lindex $argv 0]
```

```
# Create madrec object for the input file. Specify file type 1 for automatic
# determination of the CEDAR file type
mad madin
catch [$madin open 1 $infile]

# Print a one line per record summary of the file
puts "  rec     Start Time            End Time      kinst krec kindat"
set rec 0
while {[set status [$madin getNextRecord]] == 0} {
    incr rec
    set recno [format %4d $rec]
    set startTime [$madin get startTime]
    set yr1 [lindex $startTime 0]
    set mo1 [lindex $startTime 1]
    set dy1 [lindex $startTime 2]
    set hr1 [lindex $startTime 3]
    set mn1 [lindex $startTime 4]
    set sc1 [lindex $startTime 5]
    set zero 0
    if {[string length $hr1] == 1} {
        set hr1 $zero$hr1
    }
    if {[string length $mn1] == 1} {
        set mn1 $zero$mn1
    }
    if {[string length $sc1] == 1} {
        set sc1 $zero$sc1
    }
    set endTime [$madin get endTime]
    set yr2 [lindex $endTime 0]
    set mo2 [lindex $endTime 1]
    set dy2 [lindex $endTime 2]
    set hr2 [lindex $endTime 3]
    set mn2 [lindex $endTime 4]
    set sc2 [lindex $endTime 5]
    set zero 0
    if {[string length $hr2] == 1} {
        set hr2 $zero$hr2
    }
    if {[string length $mn2] == 1} {
        set mn2 $zero$mn2
    }
    if {[string length $sc2] == 1} {
        set sc2 $zero$sc2
    }
    puts "$recno  $mo1/$dy1/$yr1 $hr1:$mn1:$sc1  $mo2/$dy2/$yr2 $hr2:$mn2:$sc2 \
         [$madin get kinst] \
         [$madin get krec] \
         [$madin get kindat]"
}

# Close file and delete madrec object
$madin close
$madin destroy
```

madtclsh Commands

- ***mad***

  mad *madObj*: Creates a madrec object, i.e. a new Tcl command named madObj.

  - **destroy**

    madObj destroy - destroys madObj.
  - **open**

    madObj open *fileType fileName* - opens Cedar file fileName and associates it with madObj. The following file types are supported:

    - Open Cedar file for sequential and random reading:
      - 1 - Determine file type automatically
      - 10 - Madrigal file
      - 11 - Blocked Binary file
      - 12 - Cbf file
      - 13 - Unblocked Binary file
      - 14 - Ascii file

      Create Cedar file for update; discard previous contents if any:
      - 2 - Madrigal file
      - 20 - Madrigal file
      - 21 - Blocked Binary file
      - 22 - Cbf file
      - 23 - Unblocked Binary file
      - 24 - Ascii file
      - 25 - Unblocked Binary file image in memory. Supports sequential read and write, sequential and random read.
      - 26 - Unblocked Binary file image in memory. Supports sequential read and write, sequential and random read and in place edits.
  - **close**

    madObj close - closes the file associated with madObj and disassociates it from madObj.
  - **getNextRecord**

    madObj getNextRecord - reads the next Cedar record and fills a Madrec structure with the information in the record.
  - **putNextRecord**

    madObj putNextRecord - writes (appends) the current Cedar record to a file.
  - **getPreviousRecord**

    madObj getPreviousRecord - reads the previous Cedar record and fills a Madrec structure with the information in the record.
  - **getRecordByRecno**

    madObj getRecordByRecno *recordNumber* - reads the specified Cedar record and fills a Madrec structure with the information in the record.
  - **getRecordByKey**

    madObj getRecordByKey *key* - reads the specified Cedar record and fills a Madrec structure with the information in the record.
  - **rewind**

    madObj rewind - positions the file at the first record.
  - **copy**

    madObj copy madObjOut - copies the currect CEDAR record from madObj to madObjOut.
  - **checkRecord**

    madObj checkRecord - checks the current CEDAR record for compliance with the standard. Returns 0 if the record is compliant.

- **parmCodeArray**
  madObj parmCodeArray - Under development.
- **parmArray**
  madObj parmArray - Under development.
- **printProlog**
  madObj printProlog - Prints the prolog of the current CEDAR record.
- **createRecord**
  madObj createRecord *parmList* - Creates a new CEDAR record using the parameters in parmList.
- **printRecord**
  madObj printRecord *options* - Prints an ASCII version of the current CEDAR record. Options: -d - decimal -h - hex
- **get**
  madObj get *parameter* - Gets the specified parameter from the current CEDAR record.
  - *numBlocks*
  - *fileType*
  - *missing*
  - *error*
  - *numParms*
  - *parmsList*
  - *parmLoc*
  - *parmMin*
  - *parmMax*
  - *startTime*
  - *endTime*
  - *startIndex*
  - *endIndex*
  - *ltot*
  - *krec*
  - *kinst*
  - *kindat*
  - *ibyr*
  - *ibdt*
  - *ibhm*
  - *ibcs*
  - *ieyr*
  - *iedt*
  - *iehm*
  - *iecs*
  - *lprol*
  - *jpar*
  - *mpar*
  - *nrow*
  - *kpar*
  - *word*
  - *startJday*
  - *endJday*
  - *header*
  - *parcodes1d*
  - *parcodes2d*
  - *parm1d*

        - ◊ *parm2d*
        - ◊ *1dInt*
        - ◊ *2dInt*
    - **set**
      madObj set *parameter value* - Sets the specified parameter in the current CEDAR record.
        - ◊ *krec*
        - ◊ *kinst*
        - ◊ *kindat*
        - ◊ *startTime*
        - ◊ *endTime*
        - ◊ *1dParm*
        - ◊ *2dParm*
        - ◊ *1dInt*
        - ◊ *2dInt*
- ***cedarCode***
    - ♦ *destroy*
    - ♦ *numCodes*
    - ♦ *code*
    - ♦ *codeIndex*
    - ♦ *type*
    - ♦ *scaleFactor*
    - ♦ *units*
    - ♦ *description*
    - ♦ *int16Description*
    - ♦ *mnemonic*
    - ♦ *format*
    - ♦ *width*
- ***getKey***
- ***isr_point***
- ***isr_look***
- ***isr_geodetic2geocentric***
- ***isr_geocentric2geodetic***
- ***isr_dircos***

Further details of the Madrigal Tcl are found by examining the tcl source code from OpenMadrigal.

Madrigal Tcl API	Doc home	Madrigal home

Previous: Fortran API **Up:** Madrigal developer's guide **Next:** Madrigal file format

Madrigal file format	Doc home	Madrigal home

Previous: Tcl API **Up:** Madrigal developer's guide **Next:** Cedar file format

Madrigal file format

Physical records:

```
At MHO physical records are 6720 16-bit words long

WORD(0)            = Total number of significant words in a physical record,
                     including the Checksum.

WORD(1)            = Pointer to the first word of the first logical
                     record contained in physical record.
                       - set to zero if the physical record doesn't contain
                         any complete logical records i.e. it just contains
                         the last part of a logical record.)

WORD(2)            = Pointer to the first word of the last logical
                     record contained in physical record.

WORD(3->WORD(0)-2) = Logical records

WORD(WORD(0)-1)    = Checksum.
```

Logical Records:

```
WORD(0->15)        = Same as NCAR binary logical records.

WORD(16)           = Pointer to word 1 of previous logical record.
                      - could be contained in previous physical record.
                      - set to zero in the first logical record of the file.

WORD(17->20)       = 32-bit Start and end times of the logical record.
```