Version 2.8 May 2, 2005

CARMA Auxiliary Services (WP#163)

Peter Teuben, Chul Gwon, Marc Pound

1. FUNCTIONALITY

Auxiliary Services is intended to provide routines commonly needed for array control and observational planning. These routines are mostly, but not exclusively, astronomical in nature. Auxiliary Services will contain a mixture of new code, reused code, and wrappers for external libraries. It will involve both library routines and standalone programs (depending on the nature of the service). We describe the services below. Observational planning tools that use auxiliary services must work off-site (e.g., no RDBMS) and ideally without a network connection.

1.1. Tables and Catalogs

Table representation and I/O is needed by control procedures and for observational planning. Examples are source catalogs (a catalog is just a kind of table), spectral line catalogs, phase switch tables, and flux calibrator history. This package will provide support for reading and writing of tables, including both public and private catalogs.

Catalog support is needed both on-site and off-site. On-site, source catalogs provide the connection between a source name and its coordinates (including, e.g. VLSR velocity, but also proper motion) which ultimately get passed to the observing system. Similarly, line catalogs provide the connection between a spectral line name and its rest frequency. Off-site, these catalogs are needed by the user planning observations perhaps disconnected from internet (laptop on the airplane paradigm). Flux calibrator history is needed for (obviously) flux calibration, planning observations, and is also useful for system checks.

Source catalogs should contain catalog name and source specifications, where the specification would consist of source name(s), coordinates and equinox, velocity, velocity reference frame and definition, proper motion, source size and flux estimates as a function of frequency (Requirement 2.6.14 R1-3). The requirement that source size and flux be part of the source catalog (2.6.14-R2) will make a non-RDBMS implementation more difficult, as it makes the size of a single catalog entry variable. It may also put an unnecessary burden on observers when creating their observing program. Its intention is essentially covered by the calibrator flux catalog; we suggest it be dropped.

Line Catalogs in general are obtained externally, e.g., the Lovas and Poynter-Pickett catalogs. However, if we want to create a separate site catalog (say, frequently observed lines), then the minimum columns needed are line identification and rest frequency.

It is a requirement (2.7.5-R1) that the “system automatically harvest calibrator visibilities and make them available as a function of frequency and time.”[1] Most user access to the flux measurements will be the measurements themselves rather than the visibilities, so a separate flux calibrator catalog seems appropriate where some quality control is possible during the transfer. Flux calibrator history should be available to data calibration software. For MIRIAD, this is accomplished by the ASCII file `$MIRCAT/cals.fluxes`, see Section 5.2 for an example.

Previous standard source and flux catalogs from BIMA and OVRO should be transferred for testing and first light.

1.2. Ephemerides and Planetary Brightness

Ephemerides are needed by the on-site observing system to get RA, DEC for moving objects like planets and comets. Off-site, again they are needed for observational planning. This package will provide object interfaces to read astronomical ephemerides for control and observational planning tools. It will also provide a program to periodically download updated ephemerides from JPL. In addition to apparent size, which is obtained via the ephemeris routines, we need planet brightness calculations for flux calibration.

1.3. Astronomical Timekeeping

There are two widely used time standards. One is the rotation of the earth, and the other is the frequency of atomic oscillations. The earth’s rotation is not uniform; its rate exhibits both periodic changes and long term drifts on the order of a second per year. Atomic standards are the closest approximations currently available to a uniform time with accuracies on the order of microseconds per year.

For internal time representation, CARMA uses Modified Julian Day (Scott & Woody 2002; Requirement 2.6.5-R5). However, for source tracking and observational planning, universal and/or sidereal time are needed. Auxiliary services will provide appropriate conversion between the following timescales as needed[2]

---

[1] We believe this requirement is poorly specified. We suggest it be re-written as “make their fluxes available as a function of frequency and time.”

[2] A good review of astronomical timekeeping, from which much of this discussion is copped, can be found at `http://www.cv.nrao.edu/r̃fisher/Ephemerides/times.html`

1.3.1. Atomic Times

International Atomic Time: TAI

The SI second is defined as "the duration of 9,192,631,770 cycles of radiation corresponding to the transition between two hyperfine levels of the ground state of Cesium 133." The International Atomic Time scale (Temps Atomique Internationale, TAI) is the primary world time standard. It is a statistical timescale based on a large number of atomic clocks.

Coordinated Universal Time: UTC

Coordinated Universal Time (UTC) is the basis of civil timekeeping. By definition, UTC and TAI have the same rate, but UTC stays close to Mean Solar Time (and thus slows down) by adding integer numbers of seconds, called leap seconds, from time to time.

$$UTC = TAI - N_{leap} \tag{1}$$

where N_{leap} is the number of leap seconds for the date in question. The leap second table is available from the International Earth Rotation Service (IERS). The CARMA master clock is synched to UTC (and thus TAI) via GPS. The number of leap seconds (now at 32) has not changed since January 1, 1999, and there is discussion of abandoning it.

Julian Day

The Julian Day (JD) of a particular instant of time is the number of days and fractions of a day since 12 hours UTC on January 1 of the year -4712, where the year is given in the Julian proleptic calendar. The Modified Julian Day (MJD) was introduced by space scientists in the late 1950's. It is defined as $MJD = JD - 2400000.5$. The half day is subtracted so that the day starts at midnight in conformance with civil time reckoning in Greenwich. As mentioned above, CARMA expresses UTC as MJD (see carma::util::Time).

Dynamical Time: TT, TDB

Dynamical time is the independent variable in the theories which describe the motions of bodies in the solar system. When looking up the position of a solar system body in a precomputed ephemeris, the date and time used must be in terms of one of the dynamical timescales.

Terrestrial Time (TT)[3], with unit of duration 86400 SI seconds on the geoid, is the independent

[3]You might also find references to this as ET

argument of apparent geocentric ephemerides. It is linked to TAI through the exact formula

$$TT = TAI + 32.184\ s \tag{2}$$
$$= UTC + N_{leap} + 32.184\ s \tag{3}$$

Barycentric Dynamical Time (TDB) is the independent argument of ephemerides and dynamical theories that are referred to the solar system barycenter. TDB differs from TT by an amount which cycles back and forth by between 1 and 2 milliseconds due to relativistic effects. Planetary motions are computed using TDB.

1.3.2. Earth Rotation Times

Universal Time: UT1

Universal Time (UT1 = UT) measures the actual rotation of the earth, essentially mean solar time. It is the observed rotation of the earth with respect to the mean sun corrected for the observer's longitude with respect to the Greenwich Meridian and for the observer's small shift in longitude due to polar motion. Since the earth's rotation is not uniform, the rate of UT1 is not constant, and its offset from atomic time is continually changing in a not completely predictable way.

UT1 is needed for computing the sidereal time, an essential part of pointing a telescope at a celestial source. To obtain UT1, one must look up the value of UT1-UTC (aka DUT1) for the date concerned in tables published weekly by the IERS.

$$UT1 = UTC + DUT1 \tag{4}$$

The quantity UT1-UTC typically changes by 1 or 2 ms per day, and can only be obtained by observation, though seasonal trends are known and the IERS listings are able to predict some way into the future with adequate accuracy for pointing telescopes. The value as of this writing is about -600 ms.

Sidereal Time: GMST, GAST, LMST, LAST

Greenwich Mean Sidereal Time (GMST) is linked to UT1 by a numerical formula described in *The Astronomical Almanac*. There are no leap seconds in GMST, but the second changes in length along with the UT1 second, and also varies over long periods of time because of slow changes in the Earth's orbit. This makes the timescale unsuitable for everything except predicting the apparent directions of celestial sources.

Greenwich Apparent Sidereal Time is Greenwich Mean Sidereal Time (GMST) corrected for the shift in the position of the vernal equinox due to nutation. The right ascension component of nutation is called the "equation of the equinoxes":

$$GAST = GMST + EE. \tag{5}$$

The Local Mean Sidereal Time (LMST) is just GMST plus the observer's east longitude, where longitude is measured positive to the east of Greenwich (so LMST is GMST *minus* what we traditionally think of as our longitude).

Local Apparent Sidereal Time (LAST or simply LST) is the apparent right ascension of the local meridian, from which the hour angle of any celestial source can be determined knowing its RA. It is obtained from GMST by adding the east longitude (corrected for polar motion, $\Delta\lambda$) and the equation of the equinoxes.

$$\begin{aligned} LAST &= GMST + \mathscr{L} + \Delta\lambda + EE \qquad (6)\\ &= GAST + \mathscr{L} + \Delta\lambda \qquad (7)\end{aligned}$$

1.4. Source Coordinates and Velocities

Sky coordinates are used for source tracking and observational planning. CARMA will support Equatorial J2000, Galactic, and AzEl sky coordinate systems (Requirement 2.6.4-R2). Full conversion from equinox coordinates to current apparent coordinates is, of course, a requirement for tracking a source. Source apparent coordinates must include a correction for diurnal aberration due to the displacement of the observer from the center of the Earth. Table 1 contains the velocity reference frames that will be supported by CARMA.

Table 1: CARMA Supported Velocity Reference Frames

Frame	Description	Definition
Topocentric	The LO is fixed, no tracking	Antenna station coordinates
Barycentric	Solar System Barycenter	JPL Ephemeris DE405
Heliocentric	Center of Sun	JPL Ephemeris DE405
Kinematic LSR	Conventional Local Standard of Rest	Solar motion towards (18h +30) at epoch 1900.0

A full source specification must include a radial velocity to allow LO tracking. As with sky coordinates, there are a number of ways of specifying velocity. One must specify a *velocity reference frame* and a *velocity definition*.[4] Table 2 contains common radial velocity definitions. The radio

[4]Requirements 2.6.2-R3 states that CARMA will support the "LSR, Heliocentric, Planetary, Topocentric and CZ velocity reference frames", confusing velocity frame and velocity definition. This needs to be corrected.

and optical (equivalent to redshift) definitions are both approximations to the relativistic formula valid for $V << c$.

Table 2: Radial Velocity Definitions

Definition	Formula	Conversion to Frequency
Radio	$V = c(\nu_0 - \nu)/\nu_0$	$\nu(V) = \nu_0(1 - V/c)$
Optical	$V = c(\nu_0 - \nu)/\nu$	$\nu(V) = \nu_0(1 + V/c)^{-1}$
Redshift	$z = (\nu_0 - \nu)/\nu$	$\nu(V) = \nu_0(1 + z)^{-1}$
Relativistic	$V = c(\nu_0^2 - \nu^2)/(\nu_0^2 + \nu^2)$	$\nu(V) = \nu_0\sqrt{1 - (V/c)^2}\ (1 + V/c)^{-1}$

To calculate Doppler corrections, we need to place the source at rest with respect to the observer at the time of interest. We can write the Doppler velocity as

$$V_{Doppler} = \vec{V}_{\oplus} \cdot \vec{r_s} + V_{rad,frame} - \vec{V}_{frame} \cdot \vec{r_s}, \tag{8}$$

where the first term is the observer's motion toward the source, the second term is the radial velocity in the relevant frame, and the last term is the motion of the frame toward the source. For Doppler tracking of the LO, the Doppler velocity must be converted to observed frequency ν, given the desired rest frequency ν_0.

1.5. Unit Conversion

The CARMA Standard Units have been defined for all public APIs[5]. Internal representation of quantities is often in units convenient for the programmer or the associated hardware. Auxiliary Services provides a comprehensive unit conversion class.

insert results of discussion on the use of units during R3 integration week here

1.6. Antenna Location and Pointing Solutions

There are a number of ways to represent the antenna locations. It has been decided that CARMA will present antenna coordinates to the observer in the convenient (East, North, Up) system, which is also used in the array configuration memo. However, other processes, such as the Delay Engine and baseline determination require a different antenna coordinate system (e.g. equatorial XYZ, which can be either geocentric or topocentric). Therefore conversion utilities are needed and will provided by this package.

After array reconfiguration, antenna positions must be determined to $\sim \lambda/10$. A baseline error $\Delta b = (\Delta X, \Delta Y, \Delta Z)$ will translate into a phase error $\Delta\phi = 2\pi/\lambda\Delta\vec{b}\cdot\vec{s}$, where s is the unit vector in

[5]`http://www.mmarray.org/workinggroups/computing/CarmaStdUnits.html`

the source direction. Observations of quasars with well-known positions are used to determine Δb. (Wright 1990; Thompson, Moran & Swenson 1986 [TSW]). Routines will be provided to determine antenna position (baseline) from such observations. (The Program to acquire such observations is part of Atomic Commands).

The pointing model is used to remove effects such as encoder zero offsets, axis misalignments, tilts, etc. The azimuth and elevation correction terms, ΔAZ and ΔEL, are described by equations with 7 to 9 terms (Wright 1990; Mangum 2001). The specification of the pointing model is covered in the Antenna API. This package will provide the program(s) to reduce pointing data and derive the pointing model coefficients. It also will provide the atmospheric refraction correction term needed for source tracking.

1.7. Observing Modes

Observing Modes are described in HLCR&S Section 2.4 and in the Singledish work package. A recap: Interferometric modes are Single pointing, Multiple pointing, and Mosaic. Singledish modes are Spectral line Position-switch, Spectral line On-The-Fly, and Continuum beam-switch. In discussions about the Singledish package, Pound and Welch have ruled out spectral line frequency-switching and continuum OTF.

These modes are specified by the observer, so must be passed to the Control system by the Observing Blocks, for instance in a get/setObsMode() in SubarrayControl. We can create a class in Auxiliary Services that has enums/string representations of the available modes.

2. IMPLEMENTATION

The solutions used by OVRO and BIMA to the above aspects of observing and control are described in the Work Package analysis [6] and will not be repeated here, except where this package is reusing previous code or algorithms.

2.1. Tables and Catalogs

Intelligent ASCII Tables will be used for a number of tables/catalogs described here. ASCII makes the most sense as data format for the end-user. It is portable and easy to edit without special tools. (It also opens up the possibility of using CVS to keep on-site and off-site catalogs in sync). By intelligent we mean it has a header that would appear to most table readers and

[6] `http://www.mmarray.org/project/WP/AuxiliaryServices/auxservices2.pdf`

plotters as comment lines, but define the names of the columns, their data types, their units etc. An example of this could look as follows[7]:

```
#|name|ra       | dec    | flux | error | date|
#| s  | r       |  r     |  r   |  r    |  r  |
#|    | deg     |deg     | Jy   | Jy    | MJD |
#
3c555 6.234366 -12.2244   3.2 0.1  342.45623349    # 2004-02-03T12:33:22.2
...
```

Although in some sense these tables perform a function that could also be take up by our RDBMS (e.g. mySQL), it is not meant to approach that level, and the simple API we are proposing will reflect that. It is meant to make reading and writing such (ASCII) tables quick and easy. For very large tables, there may be performance hits for searching. Lovas 3 mm and 1 mm catalogs together are 3000 lines; BIMA calibrator fluxes has 4800 entries; Poynter-Pickett is huge but unlikely to be used much.

A well-defined programmatic interface for this to read and write such tables is provided by carma::services::Table API[8] A Table contains an ASCII “spreadsheet” row-based representation of the file, and member functions exist to extract usefully typed (double, HMS, DMS, ...) columns or rows from this table.

```
Table t("flux.tab");
vector<double> flux, mjd;

mjd  = t.getDoubleColumn(5);       // 6th column (columns are 0-based)
flux = t.getDoubleColumn("flux");  // column named ‘‘flux’’

// getHmsColumn is overloaded to allow sexagesimal or
// space-separated H,M,S.  Ditto for DMS.
ra   = t.getHmsColumn(2);           // H:M:S.S in column 2
ra   = t.getHmsColumn(2,3,4);       // H M S.S in columns 2,3,4 resp.
```

A pure virtual class Catalog is implemented, from which various specific catalogs can be derived.

```
class Catalog
        // All Catalogs have a method to lookup an entry (row)
virtual CatalogEntry& lookup(string entryName);

class CatalogEntry
```

[7]units and types are generally not used or enforced in our table I/O

[8]see also `carma::dbms::Table` for a simular interface to our RDBMS

```
        virtual string getName();

class Source : public CatalogEntry
class SpectralLine : public CatalogEntry
class Flux: public CatalogEntry

class SourceCatalog : public Catalog
class SpectralLineCatalog : public Catalog
class FluxCatalog : public Catalog
```

- *Source Catalogs*

  Initially, ASCII source catalogs created by observers will be used directly by Control via the SourceCatalog API. Later, uploading or downloading of ASCII tables to/from the system RDBMS could be implemented. An example catalog use would be as follows:

```
// open the  CARMA source catalog on disk
SourceCatalog cat(Program::getConfFile("catalogs/SystemSource.cat");
// find a source, case insensitive
Source source = cat.lookup("3c273");
// Point the antenna at it.
drive.setRaDec(mjd, source.getXCoordinate(), source.getYCoordinate(), true);
```

  with the following an example of a source entry:

```
#| Source|    RA        |    DEC       | Parallax| Velocity | VelFrame | VelDef |  PMRA | PMDEC | ID |Comments|
#|  s    |   hms        |    dms       |    r    |    r     |    r     |   r    |   r   |   r   |  i |   s    |
#|       |              |              |         |          |          |        |       |       |    |        |
0005+544  00:05:04.363 54:28:24.938     0.0        0.0         LSR      RADIO    0.000  0.000   1     -rla
```

- *Line Catalogs*

  These are provided in ASCII by catalog authors. It is rare for a user to add a new line, but some mechanism could be put in place for user line catalogs (low priority). Automatic line frequency lookup as at OVRO is a nice feature and should be kept. This can be provided with aliases in the system spectral line catalog for the most commonly observed lines or with static methods of SpectralLineCatalog.

```
// open and read the Lovas line catalog from disk
SpectralLineCatalog cat(Program::getConfFile("catalogs/SpectralLine.cat");

// find the CO J=1-0 line
SpectralLine line = cat.lookup("CO1-0");

// get its rest frequency in GHz
```

```
Units u;
double frequency = line.getFrequency().gigahertz();

// possible shortcut (currently unimplemented)
double frequency = cat.getFrequency("CO1-0").gigahertz();

// get all CO lines in a given frequency range
vector<Spectraline> lines = cat.lookup("CO",
                                       Frequency(100,"GHz"),
                                       Frequency(231,"GHz")
                                      );
```

- *Flux Calibrator History*

  A Flux contains the source name, date of observation, flux value in Janskys, error on the flux, frequency of the measurement, and a descriptive string (which may be empty). An example of how it might work:

```
class Flux (name,date,value,error,frequency,description);
// read in the flux catalog from disk
FluxCatalag fc("carma.fluxes");

// Get a vector containing all measurements of 3c84
vector<Flux> fluxlist = fc.lookup("3c84");

// Get the measurement closest to a specific data
Flux flux = fc.lookup("3c84",mjd);

// Get only the most recent measurement,
// alias for fc.lookup("3c84",Time::MJD());
Flux flux = fc.lookupLatest("3c84");
```

2.2. Ephemerides and Planetary Brightness

The JPL Solar System Ephemeris specifies the past and future positions of the Sun, Moon, and nine planets in three-dimensional space. Many versions of this ephemeris have been produced to include improved measurements of the positions of the Moon and planets and to conform to new and improved coordinate system definitions. The current state of the art is JPL DE405 (Standish 1998, see also
`http://ssd.jpl.nasa.gov/iau-comm4`). Typically a binary file (about 4.5MB) containing the tables from which the 3D positions of solar system objects are calculated is spans 50 years. Querying a solar system object can be done many times per ms CPU. The majority of CPU is related to opening the binary file, and/or post-processing the 3D positions via the NOVAS routines.

Ephemerides could also be queried/downloaded at the JPL Horizons website; a simple scripting interface to this (e.g. via email) should be sufficient. NOVAS requires JPL fortran code though recently several C versions have also become available. We are using the `ephcom` code from ephemeris.com to access the JPL binary tables. BIMA has lots of code for planet brightness calculations (Fortran+C), including the most recent Mars model by Mark Gurwell. These will be wrapped/translated.

```
Ephemeris e;
double mjd = Time::MJD();

// set time
e.setMJD(mjd);
// set a souorce
e.setSource("mars");

// obtain current sky positions
double ra  = e.getRA();
double dec = e.getDEC();

double az = e.getAz();
double el = e.getEl);
```

2.3. Astronomical Timekeeping

We will use the Naval Observatory Vector Astrometry Subroutines (NOVAS; Kaplan et al. 1989) library where appropriate. NOVAS supplies routines to compute sidereal times and the equation of the equinoxes given a Julian Day and to convert TDB to TT. For conversion to other timescales, we will write our own routines (probably cut and paste from SLALIB). A crontab script will fetch tables of UT1-UTC from IERS at regular intervals (once a week should suffice); an API to read these will be provided by using Table. A class IERSTable is available for this:

```
IERSTable iers(Program::getConfFile("catalogs/IERS.tab"));

double dut1 =  iers.dut1(mjd);
double xpol =  iers.xpolar(mjd);
double ypol =  iers.ypolar(mjd);
```

2.4. Source Coordinates and Velocities

Conversion of J2000 sky coordinates to current epoch coordinates is required including all corrections (diurnal aberration, polar motion, etc.). NOVAS provides this functionality (*topo_star,* *topo_planet*). Full conversion of (HA, DEC) to (AZ,EL) is provided by NOVAS (*equ2hor*), but not the reverse. The Delay Engine has a method to do the reverse, which will be migrated to this package.

We will allow users to specify any of the four radial velocity definitions, Redshift is essentially the same as Optical and would be stored internally as Optical. We will provide a method to convert to frequency for a fully-specified velocity, always using the full relativistic conversion to frequency.

We need the components of the observer's motion toward the source in the relevant velocity frame. NOVAS has routines to compute observer velocities (e.g., *get_earth* returns the earth's motion vectors in heliocentric and barycentric coordinates). We will need to supply the velocity of the standard frames in an earth- or sun-centered coordinate system. For instance, the components of the LSR solar motion in J2000.0 in km/s are (–.29000685, 17.31726682, –10.00140835).

2.5. Unit Conversion and Conformable Quantities

The GNU Units library has been adopted for unit conversion and has been integrated into CARMA via carma::services::Units. The file `$CARMA_TOOLS/share/units.dat` contains the possible conversions and is fairly comprehensive; for instance the default installation already has Janskys. Any additional conversions needed are added by editting units.dat, no recompilation is necessary.

Two quantities are said to be conformable if they can be converted between each other by a scale change. We have implemented the ConformableQuantity class and derived classes (e.g., Angle, HourAngle, Length, Distance, Temperature, Velocity) for use in interfaces to reduce the chance of errors due to units (see Mars Observer).

2.6. Antenna Location and Pointing Solutions

Antenna locations will be determined from interferometric data acquired by Atomic Commands. Since we have plenty of antennas for self-calibration, we will adopt the BIMA method of antenna-based solutions. The quickest way to accomplish this is to export as MIRIAD and use the MIRIAD task BEE. BEE is a sophisticated tool for fitting antenna positions on baselines to 2 km. Shell scripts and WIP graphical plotting package have helped automate the process. *This requires the MIRIAD filler be available before well before first light.*

This package will provide routines to convert between commonly used antenna coordinate systems. The class `sza::util::Coordinates` was migrated to `carma::services::AntennaCoordinates`.

The OVRO, BIMA, and SZA antennas will retain their current pointing models. The Antenna API specifies a PointingModel that encompasses all three. The coefficients for each pointing model equation will then be determined with observations as described above. It will not be necessary to create new fitting software; the fitting software of each will be reused. For BIMA radio pointing, this is the MIRIAD task PNT and some associated shell scripts. The input to PNT is an ASCII file containing results of gaussian fits to 5-point interferometric observations of planets and point sources (the output of XPNT). PNT will be modified to remove the fitted refraction term, since that will be determined separately. At OVRO, radio and optical data were reduced by a Fortran program that runs on the VaxStation; this code must be ported to Linux. An alternative is to modify PNT to handle both BIMA and OVRO pointing models. The SZA has written its own fitting program.

The radio and optical refraction corrections will be computed using the ALMA prescription (Mangum 2001). These formulae have subarcsecond accuracy down to 2 degrees elevation. They have not yet been coded up by ALMA (Mangum, private communication), so no reuse is possible here. The class `carma::environment::Atmosphere` already contains the refractivity calculation, since it was needed for the Delay Engine. Code has been implemented in Atmosphere to calculate the refraction corrections (refractivity times the elevation mapping function), with the mapping function as described by Yan (1996) and Mangum (2001). See the Interferometry design document for detailed discussion of refraction.

3. MAJOR CLASSES

The following classes exist in the `carma::services` namespace:

- `Table` - Base class for all ASCII table operations; already implemented.
- Catalogs - `Catalog, SourceCatalog, FluxCatalog, SpectralLineCatalog`, as described above.
- `AntennaCoordinates` - Converts between various antenna location coordinate systems; actual migrated from of `sza::util::Coordinates`.
- `Ephemeris` - load the JPL ephemeris tables for selected sources and calls NOVAS routines directly.
- `AstroTime` - Astronomical Timekeeping. The most important methods are those to compute sidereal times from UTC. The inputs are the MJD on the UTC timescale; UT1-UTC will be internally supplied via IERS tables. Example usage:

```
// instantiate a location from Observatory.cat
Location carma("CARMA");
// instantiate an AstroTime for this location
AstroTime time(carma);

// get the current LST
```

```
double lst = time.localSiderealTime();

// get the current hour angle for 3c273
SourceCatalog cat;
cat.open(SourceCatalog::defaultCatalog());
Source source = cat.lookup("3c273");
HourAngle ha  = time.hourAngle(Time::MJD(),source.getXCoordinate())
```

- `carma::environment::Atmosphere` – The Atmosphere class is already well-developed in support of the Delay Engine. We refer readers to the CARMA API web page for details. Here we highlight the suggested public method for the refraction correction,

```
double computeRefractionCorrection(double elevation,
                                   double airTemp,
                                   double atmPressure,
                                   double relHumid,
                                   double frequency);
```

  This method will return the refraction pointing correction in radians, to be added to the computed *in vacuo* elevation. The correction will be appropriate for optical or radio depending on the value of frequency ($\nu > 3$ THz $= 100\mu$m is considered "optical"). Additional private methods that compute various portions of the refraction correction are also implemented.

  Atmosphere also implements the conversion routines between Ambient Temperature, Dewpoint Temperature and Humidity:

```
double computeHumidity(double airTemp, double dewTemp, int method=0);

double computeDewpoint(double airTemp, double Humidity, int method=0);
```

- `Source` The Source class is derived from CatalogEntry and represents a source from the CARMA catalog. Units of Source member variables will follow the FK5 conventions. It is instantiated primarily using Conformable Quantities.

```
class Source (
            const std::string&  name,
    const Angle&        xCoord,
    const Angle&        yCoord,
    const Velocity&     velocity,
    const Distance&     distance,
    coordSysType        coordSys = RADEC,
    double              xProperMotion = 0.0,
    double              yProperMotion = 0.0,
    const std::string&  catalogFormat = "CARMA",
```

```
unsigned long   idNo = 0,
const std::string&  comments = "This Space For Rent"
 )
```

An alternative constructor substitutes the Conformable Quantity `carma::services::Parallax` for Distance. To support velocity specifications a separate Velocity class has been created. It is intended that this class will handle conversions between various velocity reference frames and definitions.

- `Units` – The Units class is simple and easy to use. The method `double convert(value, char *convertFrom, char *convertTo)` returns a converted value. For instance:

```
Units u ;
double valDegrees = 123.4;
double valRadians = u.convert(valDegress,"degrees","radians");
```

  There are also methods for standard scale factors e.g., `milli(), micro(), giga()`.

- `Conformable Quantities` Angle, Delay, Distance, Frequency, HourAngle, Length, Temperature, [Pressure], Velocity. These wrapper classes handle commonly used quantities. They are handy in constructors and at top-level interfaces, but should be used with caution if nested deeply in your classes if performance is an issue. Best is to advertise the units to use and let the clients deal with it.

- `Vector, Matrix` – mathematical vector and matrix operations.

4. PROGRAMS

Here we briefly list the programs that Auxilliary Services provides. Some of the `CARMA/carma/services/Test/` programs are also worth mentioning as they are able to help with debugging or performance issues.

4.1. checksource

This program simply shows where the source is in topocentric Ra/Dec and Az/El, it can produce this for a number of specific times. Example:

```
  % checksource source=mars mjd=+0.5 observatory=ovro nsteps=10 step=0.1
    Time              Ra (topo)    Dec        Az        El
2005-05-03T05:36:08.6 22:16:36.33 -12:28:28.20 58:29:08.24 -51:37:35.63
2005-05-03T08:00:08.6 22:16:53.25 -12:27:01.03 87:23:19.04 -24:15:56.71
2005-05-03T10:24:08.6 22:17:10.04 -12:25:33.99 108:53:38.78 04:09:29.18
```

```
2005-05-03T12:48:08.6 22:17:26.71 -12:24:06.80 135:38:00.79 28:17:37.26
2005-05-03T15:12:08.6 22:17:43.27 -12:22:39.24 176:04:30.41 40:19:23.28
2005-05-03T17:36:08.6 22:17:59.82 -12:21:11.21 218:35:04.82 31:33:47.32
2005-05-03T20:00:08.6 22:18:16.44 -12:19:42.77 247:16:49.21 08:41:52.28
2005-05-03T22:24:08.6 22:18:33.17 -12:18:14.11 269:00:12.10 -19:18:16.36
2005-05-04T00:48:08.6 22:18:50.03 -12:16:45.49 295:18:36.94 -47:11:58.12
2005-05-04T03:12:08.6 22:19:06.98 -12:15:17.12 353:15:57.31 -64:53:05.00
```

4.2. distance

Useful to compute the angular distance between two sources, of which the default reference source is the sun, therefore giving us the phase of a planet

```
  % distance source=mars nsteps=10 step=7
# Time                   sun :  az/el           mars :  az/el           angular distance
2005-05-02T17:39:56.3 118.2070044 54.00587420 219.0904067 31.02387906 71.22194157
2005-05-09T17:39:56.3 115.8390133 55.41543736 222.1412729 31.43829553 72.93436185
2005-05-16T17:39:56.3 113.5080883 56.53793565 225.3370556 31.76112806 74.64436707
2005-05-23T17:39:56.3 111.3156542 57.36960494 228.6558067 31.97323827 76.35719420
2005-05-30T17:39:56.3 109.3636946 57.91589638 232.0718082 32.05976503 78.08348411
2005-06-06T17:39:56.3 107.7531986 58.19142634 235.5591583 32.00345956 79.84070620
2005-06-13T17:39:56.3 106.5787506 58.22334772 239.0908102 31.78852910 81.64249957
2005-06-20T17:39:56.3 105.9096966 58.04375769 242.6375573 31.40593144 83.49912267
2005-06-27T17:39:56.3 105.7793693 57.68134499 246.1716099 30.85125322 85.42517596
2005-07-04T17:39:56.3 106.1936268 57.15913529 249.6699786 30.11521849 87.44527736
```

4.3. Test/tEphemeris

This programs does much like `checksource` except much more verbatim. It also have a silent mode, e.g. to check the influence of I/O accross NFS if "large" ephemeris need to be read.

```
% carma/services/Test/tEphemeris
```

4.4. Test/tIERSTable

Helpful to debug the current IERS table. Prints min and max valid times, and check interpolation into the table. This program needs a direct link to the IERS table file, not relative to some CARMA, e.g.

```
% carma/services/Test/tIERSTable in=$CARMA/conf/catalogs/IERS.tab mjd=53491
```

```
53491 -0.59399 -0.0695 0.2727
```

prints out the MJD in days, dut1 in seconds, X-polar and Y-polar motion both in arcsec.

5. EXAMPLE TABLES

In this section we show some example tables that have already been in use within BIMA/OVRO/SZA. The equivalent CARMA versions have been placed in `CARMA/conf/catalog`, where `CARMA` is the top of the source, build or install tree. The `Program::getConfFile(const string& name)` provides a convenient interface to get access to these tables.

Table 3: CARMA catalogs in CARMA/conf/catalog

Filename	Function	Interface
IERS.tab	IERS table of dut1, and polar motion	IERSTable
Observatory.cat	list of stations and their lat,lon,elev	Location
SpectralLine.cat	list of spectial lines and their frequency	SpectralLineCatalog
SystemSource.cat	list of sources, sky positions and slow motions	SourceCatalog

5.1. Lovas line catalog

The Lovas spectral line catalog[9] comes with MIRIAD. It has a somewhat complicated structure and contains a list of all identified and un-identified interstellar molecular spectral lines between about 10 MHz and 800 GHz.

```
 Frequency    Unc. Formula      Quantum                          Tr(K)  Source          Telescope    Astr.  Lab.
 (MHz)                                                           /Ta(K)                              ref.   ref.


  200809.32 *( 3) SO2          16(1,15)-16(0,16)                 4.87  OriMC-1         NRAO      12m Jew89
  200888.30 *(10) SO2          13(5,9)-14(4,10) v2=1             0.28  OriMC-1         NRAO      12m Jew89
  200913.79 *( 4) HC3N         22-21 v7=1 l= 1f                  0.73  OriMC-1         NRAO      12m Jew89
U 200936.         unidentified    (U204070?)                     0.50  OriMC-1         NRAO      12m Jew89

   68972.154*( 4) SO2          6(1,5)-6(0,6)                     0.8   OriMC-1         NRAO      11m Joh76
U  69460.         unidentified                                   0.18  OriMC-1         NRAO      11m Tur89
   69464.094*( 9) SO2          14(4,10)-15(3,13)                 0.70  OriMC-1         OSO       20m Sch83
```

[9]See also `http://physics.nist.gov/cgi-bin/micro/table5/start.pl`

The table has been reformmatted slightly and made available as `conf/catalogs/SpectralLine.cat` for automated access using `carma::services::SpectralLine` and `carma::services:SpectralLineCatalog`. *This is still the 3mm catalog, we don't have the 1mm catalog in here yet*

5.2. BIMA flux catalog

The BIMA flux catalog, also delivered via MIRIAD, was maintained by astronomers at Hat Creek (mostly Mel Wright and Rick Forster) from carefully controlled observations.

```
! SOURCES
! sorted by time and RA
!
!Source          Day.UT          Freq (GHz)       Flux(Jy)         Rms(Jy) Ref.
'nmaobs'         92MAY20.0       87.0              1.0             0.2
!
!! 0005+544
'0005+544'       98APR27.0       86.2              0.9             0.4    9
'0005+544'       98OCT22.0       86.2              0.9             0.2    9
'0005+544'       99MAR14.0       86.2              0.4             0.2    9
..
'0005+544'       03JUL01.0       86.2              0.3             0.1    9
'0005+544'       03SEP01.0       86.2              0.4             0.3    9
!
!! 0006-063
'0006-063'       95JUN18.0       86.2              1.4             0.9    7
...
```

5.3. sza ephem files

SZA ephem files (only shown here for illustrative purposes) are ASCII ephemeris, one per source, obtained via the JPL/HORIZON procedure, and look roughly as follows (only the first 4 columns count)

```
#
# MJD (TT)   Right Ascen    Declination   Distance (au)      Calendar (TT)
#---------  -------------  -------------  ------------     -----------------
53005.0000  00:33:35.2643  +03:41:19.071  1.1112017627  #  2004-Jan-01 00:00
53006.0000  00:35:47.3868  +03:56:55.925  1.1200841033  #  2004-Jan-02 00:00
53007.0000  00:37:59.9236  +04:12:32.155  1.1289840401  #  2004-Jan-03 00:00
...
53369.0000  16:03:50.9833  -20:34:42.059  2.2622996871  #  2004-Dec-30 00:00
53370.0000  16:06:43.6583  -20:43:12.186  2.2558736835  #  2004-Dec-31 00:00
53371.0000  16:09:36.7800  -20:51:32.042  2.2494140942  #  2005-Jan-01 00:00
```

Again, this table format is not supported in CARMA.

5.4. sza source catalog files

SZA source catalogs (only shown here for illustrative purposes) contain a frame of reference (e.g. `FIXED` (meaning AZEL) and `J2000`), source name, ra, dec and optional proper motions in ra and dec;

```
#--------------------------------------------------------------------------------
#     Name       RA          DEC         PMRA (hrs)   PMDEC (deg)   Magnitude
#--------------------------------------------------------------------------------
J2000 TheOct    00:01:35.71 -77:03:56.9 -0:0:00.0042 -0:0:00.177 # 4.78
J2000 30Psc     00:01:57.60 -06:00:51.1 +0:0:00.0034 -0:0:00.041 # 4.41
J2000 2Cet      00:03:44.40 -17:20:10.0 +0:0:00.0017 -0:0:00.009 # 4.55
...
alias sirius   AlpCMa
...
FIXED zenith   *  90.0 *
FIXED nadir    * -90.0 *
FIXED horizon  *   0.0 *
J2000 npole2000   00:00:00.0     90:00:00
...
EPHEM sun      $SZA_DIR/ephem/sun.ephem
```

Again, these conventions are not supported in CARMA.

6. NOTES

- procedures for new solar system objects (comets etc.). We can create them as DE405-compatible binary files, or through their orbital elements. Either way, a new service routine `solsysN` in NOVAS (the way this is normally done) will be needed for this.
- the transmitter should be a special source, it has (E,N,U) antenna coordinates and on a per-antenna basis will have the proper local AZEL computed.
- NOVAS 3.0 coming out "soon", we use the C library (2.0), the current Fortran library is 2.9. So there is always a lag between Fortran (which is what Kaplan writes) and and C (written by Bangart). Bangert reported the C 3.0 version to be available early 2005.
- IERS may discontinue use of leap seconds in the future.

7. FTE ESTIMATE AND MILESTONES

The functionality of this package has expanded since the conceptual design review (addition of antenna coordinates, sky coordinates, time, and units). The schedule for completion and FTE estimate is as follows:

Stage		Months	Responsible	Release (Completion Date)
Work Package Analysis				08/27/2002
Conceptual Design Review				09/06/2003
Preliminary Design Review				09/02/2004
Critical Design Review				05/05/2005
Design		2.0	all	
Implementation	Antenna Coordinates	0.5	Teuben	R1
	Intelligent Tables	0.5	Teuben	R1
	Refraction	0.25	Pound	R1
	Time Conversion	0.5	Teuben	R1
	Units	0.5	Gwon	R1
	Ephemerides	0.5	Teuben	R2
	Source Catalog	0.25	Gwon	R3
	Line Catalog	0.25	Gwon	R3
	Doppler Correction	0.25	Pound	R4
	Baselines	0.5	Teuben	R4
	Pointing	0.5	Pound	R4
	Planet Brightness	0.25	Teuben	R4
	Flux History	0.25	Gwon	R5
	Observing Modes	0.25	Pound	R5
	Total	7.5		
Testing		1.0	all	
Integration		1.0	all	
Package Total		9.5	FTE months	

REFERENCES

Astronomical Almanac - USNO. See URL=

Kaplan, G.H. et al. 1989, AJ, 97, 4

Lovas, F.J. - `http://physics.nist.gov/cgi-bin/micro/table5/start.pl`

Mangum, J. 2001, ALMA Memo 366

Newhall, X. X.; Standish, E. M.; Williams, J. G. 1983, Astron. Astrophys. 125, 150-167. (DE102)

Scott, S. & Woody, D. 2002, *CARMA Time Distribution*, `http://www.mmarray.org/project/WP/MasterClock/hw/TimeDistribution.pdf`

Standish, E.M..: 1998, "JPL Planetary and Lunar Ephemerides, DE405/DE406", JPL IOM 312.F-98-048.

Thompson, Moran, & Swenson, 1986 (1st edition), publ. Wiley

Wright, M.C.H.W 1990, BIMA Memo 2

Yan, H. 1996, AJ, 112, 1312

This preprint was prepared with the AAS LATEX macros v5.0.