The National Virtual Observatory Book
ASP Conference Series, Vol. 382, © 2008
M. J. Graham, M. J. Fitzpatrick, and T. A. McGlynn, eds.

Chapter 37: Space-Time Coordinate Metadata for the Virtual Observatory

Arnold H. Rots

Introduction

The full characterization of astronomical data in a semantically meaningful way requires a precise description of the properties of those data along all relevant axes. This description needs to be complete, internally consistent, and unambiguous. In this chapter we will describe the design of a metadata model that satisfies these requirements, distinguishing three broad categories of coordinate axes: generic coordinates (typically independent parameters, such as flux density, brightness, etc.), astronomical coordinates (time, space, spectral, and redshift or Doppler shift), and pixel coordinates. In all categories we distinguish three classes of metadata: the coordinate frame, the properties along the coordinates (coordinate values, errors, resolutions), and the coordinate area (the volume in coordinate space occupied by the data). The completeness of the coordinate metadata, and the underlying data model, is crucial for the Virtual Observatory since it is the part that ultimately enables the exchange of data, and their meaning, between different sub-disciplines. A more extensive discussion may be found in the formal STC standards document (see *Useful Links*). The STC standard is contained in that document, together with the STC XML Schema.

1. What Do We Need and Why Do We Need It?

Coordinate information has always been crucial to astronomical data, but many of the more subtle details were often not explicitly spelled out because they were understood in the sub-disciplines in which the data were circulated and used. Unfortunately, these sub-disciplines adopted different tacitly understood defaults, some for good practical reasons, others that were rather arbitrary. Obvious examples are the spatial coordinate systems used in Galactic, extragalactic, solar, planetary, and theoretical research. More subtle (and therefore more deadly) is the difference between the optical and the radio definition of Doppler velocities.

If the Virtual Observatory is to allow exchange of data between sub-disciplines, we will need to be very particular about the metadata that we provide, to the point of being pedantic, and not assume anything to be "obvious". We should also not be so arrogant as to pretend that we know what the users of our data need and what they do not need. We may not care when exactly our timeless image of M81 was taken, but it may be critical for someone years from now – and that means that additional information is needed: not only *when*, but also *where*, and what timescale was used.

In the following we shall first enumerate the basic requirements for all coordinate metadata and then describe what specific requirements there are for the three

main categories of coordinates. We note that there are some special provisions to accommodate simulation data which, by their nature, lack an absolute place and time.

Working one's way through the design may make working with the coordinate metadata look like a daunting task, both for the server and for the client, but this is not necessarily the case. Admittedly, the full system is fairly complicated and needs to be that way in order to enable us, eventually, to perform any and all transformations transparently. However, in practice servers will be able to compose coordinate metadata largely from boilerplate templates; and it is perfectly acceptable for clients to implement only a subset of all coordinate system options and to simply say: "Sorry, I can't handle this".

Among the existing applications that use STC's schema, VOEvent limits itself to about a dozen coordinate systems and to the specification of times and positions; areas and pixel spaces are not considered. The *Coverage* component in the *Registry*, on the other hand, relies heavily on *Regions*, rather than single positions. One of the most sophisticated applications in existence at the present time is the *Footprint Service* described by Budavari et al. in this volume (see Chapter 9), which has implemented the full complexity of the *Region* syntax to allow simple answers to simple questions, like: what is the area of the intersection of these two regions? Or: is this point contained in this region?

1.1. Basic Requirements

The basic requirements for coordinate metadata are threefold.

Complete Completeness refers to the necessity to provide all metadata details – not only the relevant information, but also the *potentially* relevant information and, as argued before, "*potentially*" should be interpreted very generously. Consequently, any piece that is missing should be considered UNKNOWN. And the rule is that it is up to the client to decide whether to accept an UNKNOWN quantity and to assign it a default value.

Unambiguous Nothing should be left ambiguous: a good-faith effort should be made to be as clear as possible in the description of the data.

Internally consistent Special care should be taken that there are no inconsistencies between different parts of the metadata which would lead, to put it mildly, to ambiguities. For example, if the same position is provided in two coordinate systems, the transformation should have been carried out correctly.

1.2. Top-level Components

There are three top-level components that make up the coordinate metadata.

Coordinate System The Coordinate System contains a *Coordinate Frame* for each coordinate. A Coordinate Frame consists of a *Reference Frame* (either a standard frame that is universally understood, such as "ICRS" or "UTC", or defined through a transformation from a known system), a *Reference Position* (the origin: either a

known place, such as “BARYCENTER” or “LSR”, or a specific position), and a “Flavor” (Cartesian, spherical, string coordinate; 1-D, 2-D, 3-D). See Fig. 1 for a basic diagram of the structure.

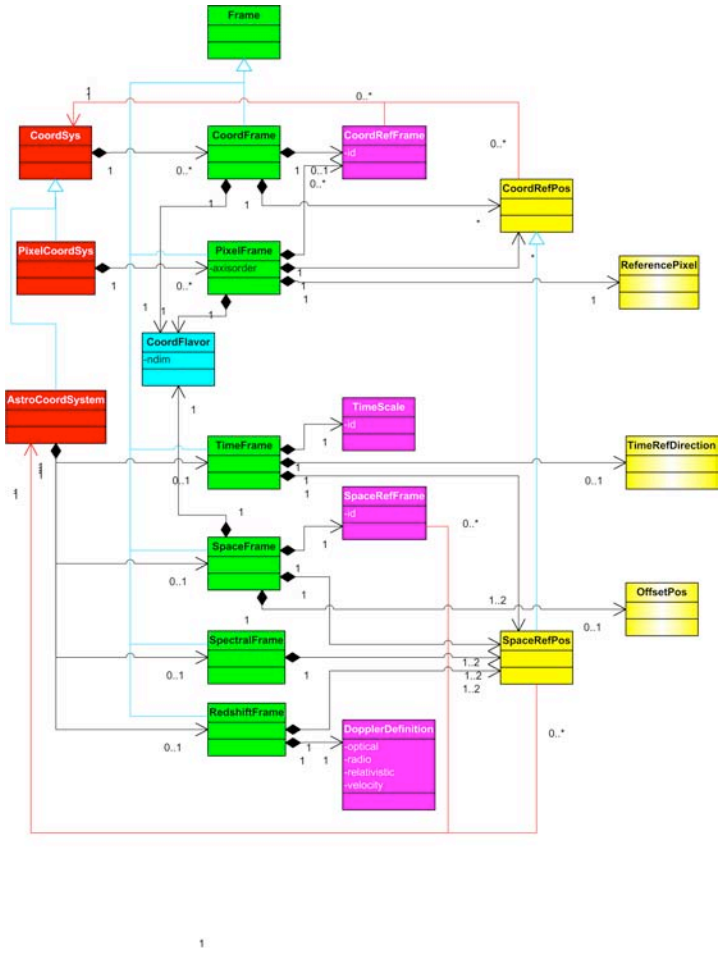

Figure 1. The basic design structure of Coordinate Systems, for the three categories: Generic, Astronomical, and Pixel. This UML diagram shows, from left to right: coordinate systems, coordinate frame flavor, the coordinate frames that form a coordinate system, and the frames’ components: reference frames, reference positions, three special components

Coordinates This contains the data’s properties along the coordinate axes: value, error, resolution, size, pixelsize. It requires a reference to a Coordinate System. More than one Coordinates object may reference the same Coordinate System.

Coordinate Area This area describes the volume in coordinate space that is occupied by the data it refers to. Typically, it consists of one or more intervals along a

coordinate axis, but there are special areas defined for 2-dimensional coordinates: *Regions*.

1.3. Coordinate Categories

Not all coordinates are equal – some have special needs. We distinguish three broad categories. Some of the differences are illustrated in Figure 1.

Generic Coordinates These are the coordinates without special needs and they have the properties outlined in the previous section. Coordinates in this category are typically the independent parameters in the data, such as flux density, brightness, temperature. They are often 1-dimensional, but there are also 2-dimensional examples, e.g. polarization intensity and angle, complex visibility (real-imaginary, amplitude-phase).

Astronomical Coordinates Specifically, this includes Time, Space (position and velocity), Spectrum, and Redshift (Doppler shift). These provided the original impetus for this project (the others are generalizations) as it was recognized that they need to be considered together since they are intertwined: spatial position has no meaning without the time when it was measured, and time has no meaning if we do not know where it was measured; spectral and redshift data require a reference position in phase space in order to be properly interpreted. "Redshift" should be taken as a generic term, referring to a spectral shift that is interpreted as a Doppler shift. Note that it is neither considered a spatial velocity coordinate, nor a spectral coordinate. *Doppler velocities* are defined as a formalism based on spectral information that may or may not have a direct association with spatial velocities; for one thing, the derived Doppler velocities depend on the definition used ("optical" *versus* "radio"), but in the relativistic regime any physical relation breaks down. Neither is it a spectral coordinate as it is very different from the spectral coordinate in Spectral Energy Distributions; moreover, one can well imagine having datasets containing Doppler velocities based on multiple spectral lines. "Real" velocities are found in solar system ephemerides, orbit ephemerides, simulations, proper motions, etc.

The Astronomical Frames have an elaborate set of Standard Reference Positions in common: GEOCENTER, BARYCENTER, LSR, planetary centers, etc.; but also TOPOCENTER: the place from which the observation was made – which leads to the requirement to include the observer's position in the metadata. More details on these frames are provided in a later section.

Pixel Coordinates Pixel coordinates differ from generic coordinates in that, on the one hand, they require the concept of reference pixels, while, on the other hand, they do not need units and only contain values (i.e. no other characterizing quantities, such as errors, etc.).

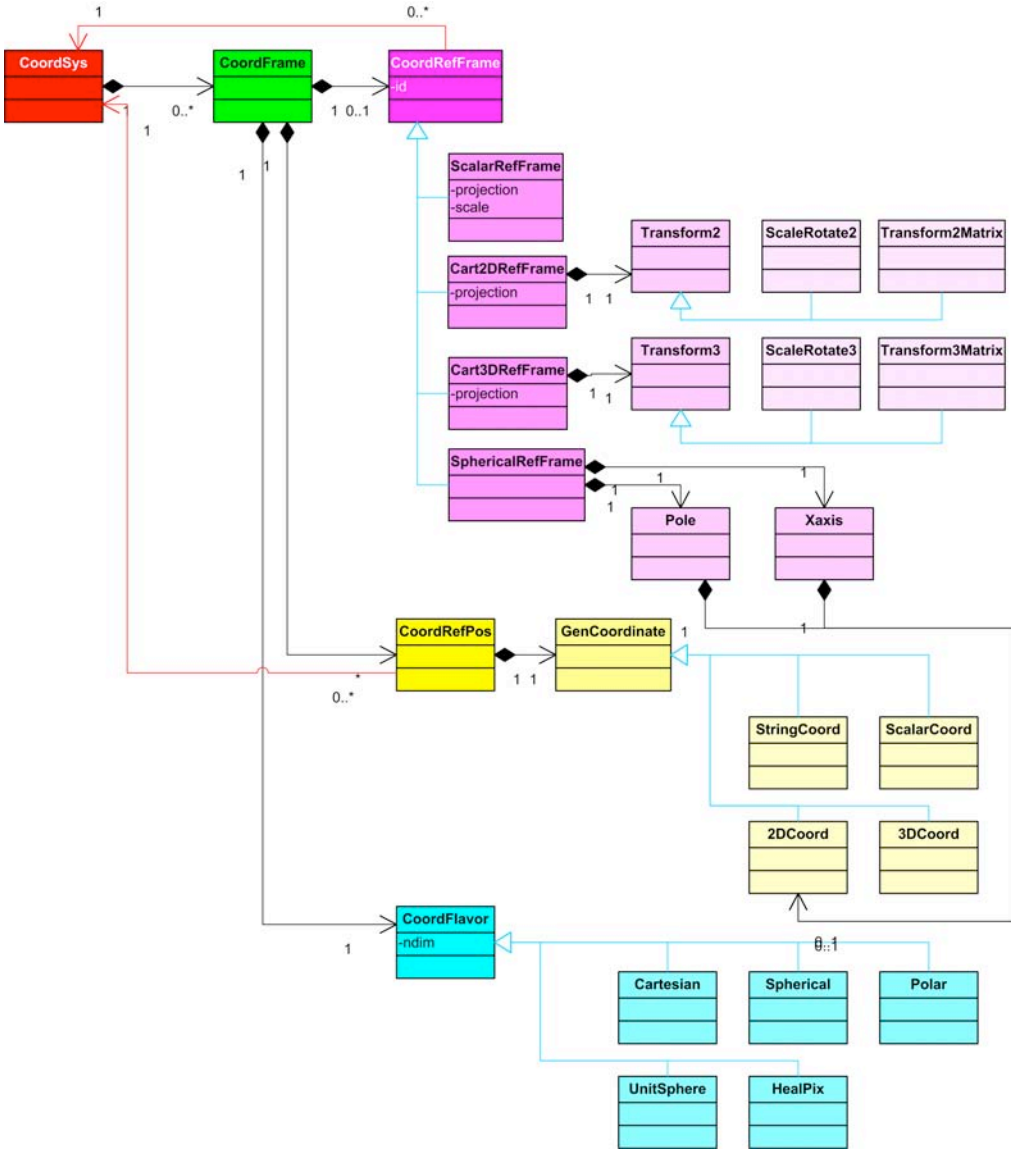

Figure 2. The detailed design structure of the Generic Coordinate System, elaborating on the design of the Frames' components: Reference Frame, Reference Position, and Flavor

2. Coordinate Systems

This section describes the details of the various types of coordinate systems, first the generic systems, then the astronomical and the pixel systems.

2.1. Generic Coordinate System

As explained before, a Coordinate System is a conglomerate of Coordinate Frames. Each Coordinate Frame consists of three parts: a Flavor, a Reference Frame, and a Reference Position. It goes without saying that these three parts need to be consistent in, for instance, dimensionality. Figure 2 illustrates the following explanations.

Coordinate Flavor The Coordinate Flavor provides two pieces of information: the dimensionality of the Coordinate Frame (1, 2, 3, or "string" – e.g. Stokes) and the configuration (Cartesian, spherical, polar, unit-sphere, or HEALPix).

Coordinate Reference Frame The Reference Frame defines the Coordinate Frame in terms of a projection type and the scale-and-rotate elements of a coordinate transformation (for Cartesian frames) or the pole and X-axis (for spherical systems). For 1-dimensional frames this reduces simply to a scale factor and a limited set of projections (linear and logarithmic, for now). The more elaborate set of projections is targeted specifically at (2-D) spherical-to-Cartesian transformations.

Coordinate Reference Position The Reference Position is the final component of the transformation: the translation. It specifies the origin of the frame. Needless to say, it should be consistent with the Reference Frame and drawn from the same coordinate frame on which the rotation and scaling are defined.

2.2. Astronomical Coordinate System

The design of the Astronomical Coordinate System is illustrated in Figure 3. In addition to the astronomical frames, the Astronomical Coordinate System is also allowed to include one or more Generic Coordinate Frames. The properties of the components of the astronomical frames are essentially the same as those of the generic frames, with some individual additions or restrictions. The main overall extra feature is a list of Standard Reference Positions which are commonly shared: GEOCENTER, BARYCENTER, HELIOCENTER, TOPOCENTER, LSR(K/D) (specific to Spectral and Redshift), GALACTIC_CENTER, LOCAL_GROUP_CENTER, EMBARYCENTER, MOON, MERCURY, VENUS, MARS, JUPITER, SATURN, URANUS, NEPTUNE, PLUTO (yes ☺), RELOCATABLE (specific to simulations), UNKNOWNRefPos. For reference positions in the solar system, other than the GEOCENTER, one should also indicate which planetary ephemeris was used (DE-200, DE-405). The intrinsically 1-dimensional frames (time, spectral, redshift) do not require a Flavor. Frames may contain *Units* that are valid for all Coordinates and Coordinate Area objects that refer to it, except when overridden by Unit specifications in the individual Coordinate elements.

Time Frame The equivalent of a Reference Frame for Time is the *Time Scale*. Recognized values are: TT, TDT, ET, TAI, IAT, UTC, TDB, TEB, TCG, TCB, LST, LOCAL (only to be used in conjunction with RELOCATABLE spatial coordinates, in simulations). An added item is the *Time Reference Direction:* for Reference Posi-

tions other than TOPOCENTER one has to apply a pathlength correction which involves an assumption as to the direction of the path.

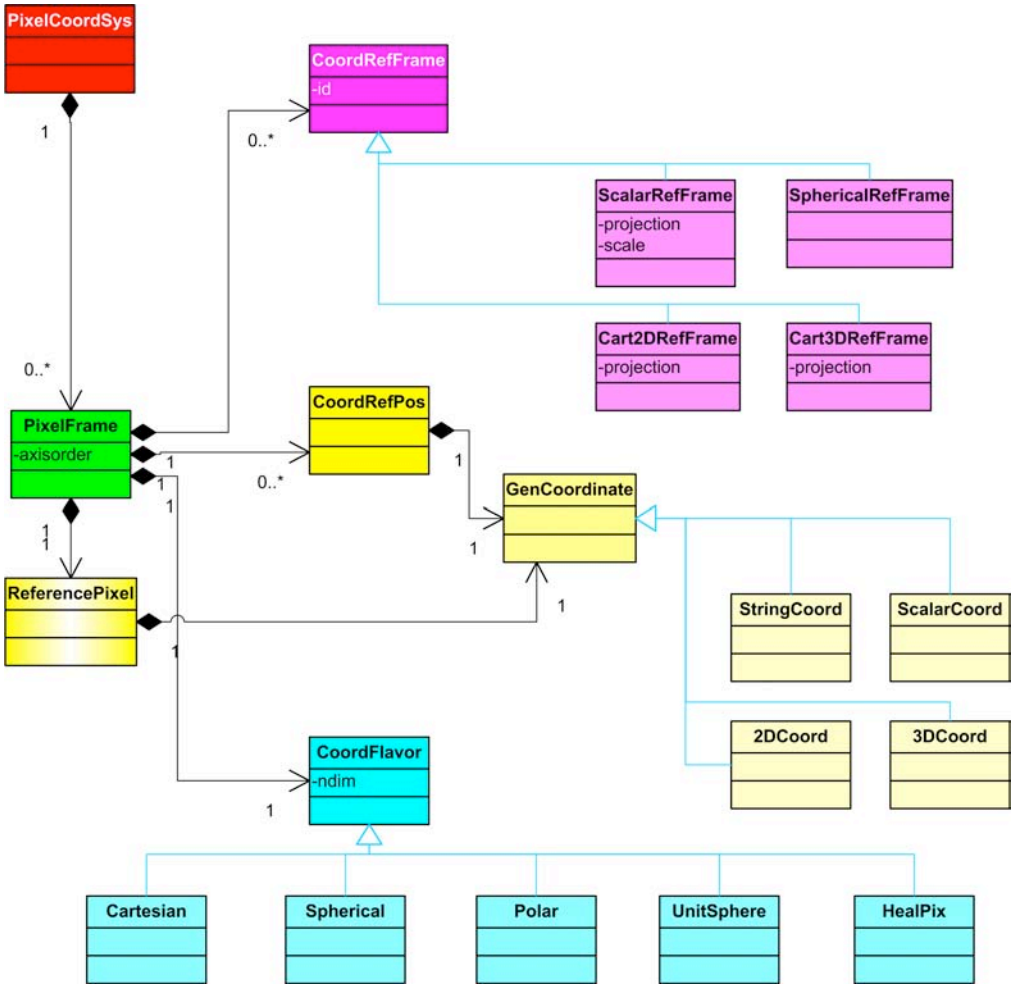

Figure 3. The Astronomical Coordinate System design, an elaboration of the components in the frames that are typically astronomical. In addition to the Frames shown in this figure, the Astronomical Coordinate System may include Generic Frames.

Spatial Frame This is where the largest number of standard frames has been defined, including FK4, FK5, ICRS, ecliptic, galactic, super-galactic, geodetic, solar, and planetary (-centric, as well as -graphic). Some of these require an equinox and one may optionally provide an epoch. Note that, although we generally think of most of these as spherical reference frames, their Flavor may be 2-D spherical, Healpix, 3-D unitsphere, or 3-D Cartesian. The Space Frame controls positions as well as (true) velocities. Spatial Frames will presumably be the most active users of 2-D and 3-D

coordinate transformations and of spherical to 2-D Cartesian projections. For convenience, one may specify an *Offset Position* to enter relative coordinate values.

Spectral Frame This is a 1-dimensional Cartesian frame with just a reference position in phase space. However, if a Redshift coordinate is present, the Spectral coordinate represents the rest frequency of the line used.

Redshift Frame The Redshift (or Doppler shift) Frame is very much like the Spectral Frame. In addition, it has a *Doppler Definition* element that indicates the Doppler definition used (optical, radio, or relativistic) and whether the coordinates are expressed as true redshifts or as Doppler velocities. Don't ever use any value for the Doppler Definition other than "optical" in combination with "redshift".

2.3. Pixel Coordinate System

It is important to realize that the intent is that one Pixel Coordinate System provides the pixel information for a pixel array, whatever its dimensionality may be. The Pixel Coordinate System will, in principle, contain one frame per pixel array axis, except where it concerns an intrinsically multi-dimensional coordinate, such as spatial position. In general, the sum of the dimensionalities of the frames should equal the dimensionality of the pixel array. Each frame ought to have an Id, so that the Coordinate and Coordinate Area components can refer to individual frames. The "extras" in Pixel Frames are axis order attributes (cross-referencing the Pixel Frames with the pixel array axes) and Reference Pixels to specify the location of the Reference Position in pixel coordinates. For an illustration of the design, see Figure 4.

The Pixel Frames will almost always contain coordinate transformation elements that specify scaling, projection, and orientation, as needed.

3. Coordinates

A *Coordinates* object is a collection of individual Coordinate objects. Coordinate objects may be 1-, 2-, or 3-dimensional, or string. They may contain one or more of the following components: Name, Value, Error, Resolution, Size, Pixel Size; string and pixel coordinates may only have a Name and Value. They may all have Units and they should have (except for Astronomical Coordinates, since the linking is implied) a reference to a *frame Id*. For multi-dimensional Coordinate objects, Error, Resolution, etc., may take on various appearances: a single radius, a box, and ellipsoid, or a matrix.

3.1. Generic Coordinates

Generic Coordinates objects are completely described by the above.

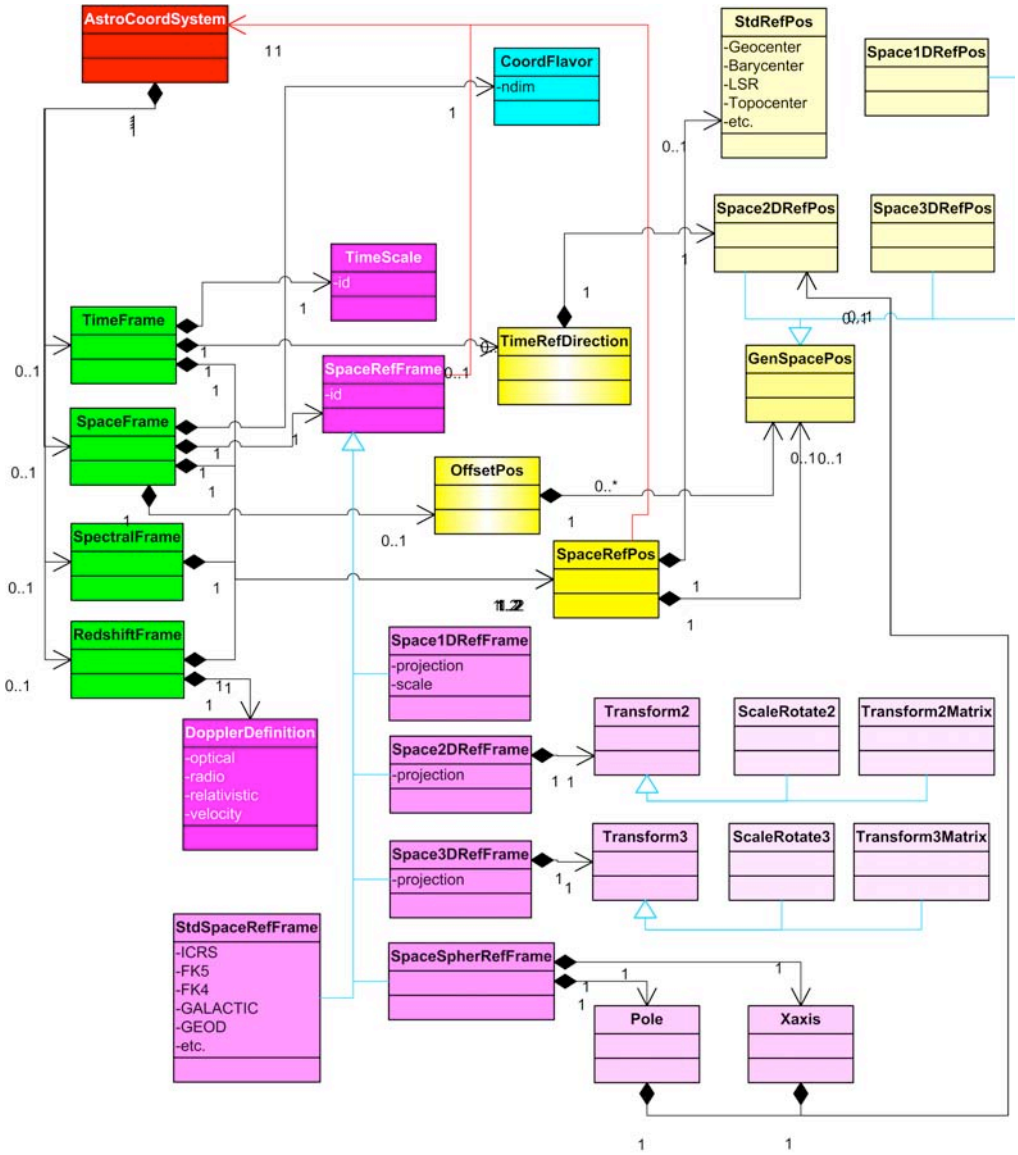

Figure 4. The design of the Pixel Coordinate System: Pixel Frame and its components. Note the addition of axis order and Reference Pixel.

3.2. Astronomical Coordinates

Astronomical Coordinates do not need a reference to a frame Id since the reference is implicit. In addition, there are a few extra comments to be made with respect to time and space. Instead of providing direct values for astronomical coordinates, one may also refer to values stored in a FITS binary table. This is particularly useful for catalog tables and orbit ephemeris files.

Time The *AstronTime* object contains an absolute time, in ISO-8601 format (with 0001-01-01T00:00:00 being the earliest time that can be represented), Julian Day, Modified Julian Day (MJD = JD – 2400000.5), or RELOCATABLE (for simulations only), an optional Time Scale (so that the class can be used independently of a Coordinate System – though this is not to be encouraged), and an optional time offset (from the absolute time). Time should be expressed with appropriate precision; double precision variables are not necessarily sufficient. In the XML schema we use the data type *decimal*.

Space The spatial position may be 1-, 2-, or 3-dimensional. It is possible to provide coordinates with lower dimensionality than the associated frame. One may, for instance, just list Right Ascension in an equatorial frame. In such a case the name of the coordinate should be provided to avoid ambiguity. The spatial coordinates may consist of a position and a velocity. Both have the same structure, properties, and restrictions, with the exception of an additional time unit for velocity. Velocity units are separated into a spatial unit (numerator) and a time unit (denominator) to provide more flexibility (and simpler enumeration). These velocities should be true space velocities, including proper motions, but not Doppler velocities.

There are some special cases defined, as well. A Keplerian orbit (i.e. position and velocity, as a function of time) may be specified through a set of orbital elements. And there are hooks to define curves – currently only implemented as straight lines or great circles.

Spectral The spectral coordinate is a simple 1-dimensional one, with appropriate units (if not defined in the frame).

Redshift The redshift coordinate is also 1-dimensional and should have a spatial and a time unit if not defined in the frame and if expressed as Doppler velocity.

3.3. Pixel Coordinates

Pixel coordinates do not have units or error, resolution, or size components – only value and, optionally, a name. They may be 1-, 2-, or 3-dimensional and should have a frame Id to identify which axis they refer to.

4. Coordinate Area

The Coordinate Area specifies the volume in coordinate space that is occupied by the object that the metadata refer to. It requires a reference to the Coordinate System that is being used. The basic building block is the coordinate interval which may be 1-D (range), 2-D (rectangle), or 3-D (block), but the area along any coordinate axis may consist of any number of such intervals which are logically OR-ed together. In three dimensions we also allow spheres to be specified and in two dimensions a variety of region shapes (see below).

4.1. Intervals

Intervals consist of a lower and an upper bound. Both are optional, in order to allow the specification of only upper or lower limits. There are two Boolean attributes indicating whether the bounds are part of the interval; the default is *true*. The optional fill factor attribute (a value between 0.0 and 1.0; default: 1.0) specifies what fraction of the interval is actually covered; it is a mechanism to indicate sparse coverage without having to provide full detail.

4.2. Regions

The 2-dimensional case is special, whether on the celestial sphere or in a simple plane. The ability to define complicated areas in two dimensions, whether it be outlining source shapes or describing instrumental footprints, is absolutely crucial and for this purpose we have introduced the concept of *Regions*. A Region is either a *Shape* or the result of an operation on one or more Regions. The operations are:

- **Intersection** (of two or more Regions)
- **Union** (of two or more Regions)
- **Negation** (of one Region)
- **Difference** (of two Regions): this operation is defined as a convenience since it can also be written as a combination of intersection and negation; note that this operation, unlike the others, is not commutative

There are nine Shapes defined; the first six apply to Cartesian or spherical frames:

- **AllSky**: just a simple way of specifying all of the 2-D space
- **Circle**: through center and radius
- **Ellipse**: through center, semi major and minor axes, and position angle; beware that definition of an ellipse in spherical coordinates is somewhat tricky
- **Polygon**: through a list of vertices; in spherical coordinates we assume that the sides are great circles, but they may be specified as small circle segments
- **Box**: through center and length of the sides; again, this is a convenience since the box can also be specified as a polygon
- **Sector**: through an apex position and the position angles of the two bounding lines

The remaining three are defined on the unit sphere:

- **Convex**: through an unordered list of positions; it is the smallest convex polygon that contains all positions on the list
- **ConvexHull**: the intersection of two or more HalfSpaces (this can also be specified as a polygon); a HalfSpace is a circular area on the Unit Sphere, defined through a normal vector and an intersection distance along that vector: it is the halfspace on the sphere that contains the vector location and is sliced

off the sphere by the plane intersecting the vector at the specified distance from the center, normal to the vector

- **SkyIndex**: currently just a placeholder allowing Regions in the future to be specified in terms of various sky indexing schemes

Like other areas, Regions may have a fill factor; note that calculating fill factors becomes uncertain when operations are performed on Regions. Regions may have an associated surface area object that includes a Boolean indicating whether its value has been validated.

5. XML Implementation

A schema that allows STC metadata to be encoded in XML is provided as part of the STC standard (see *Useful Links* below). We will not discuss that schema here in detail but rather refer the reader to the STC standards document.

However, we will explain here its referencing mechanism. Xlink has been adopted as a general-purpose IVOA referencing mechanism. Xlink allows an element to refer to any object anywhere. In the context of STC we restrict this to actual XML elements available on the web, using XPath syntax. We have therefore defined an STC basetype from which many types are derived that contains the following attributes:

- **xlink:type**We only use the value “simple”
- **xlink:href**The URI where an element of the correct type is to be found
- **idref**A reference to an existing element in the document
- **id**An ID allowing the element to be referenced
- **ucd**An optional UCD string

Consequently, the contents of an element may be specified in three ways:

1. Through the actual contents in its body
2. Through a reference to another element in the document (idref)
3. Through a reference to an element defined elsewhere (Xlink)

This referencing model has some consequences that one should be aware of. First, there may be multiple (and possibly contradictory) specifications of the contents of such an element; if this is the case, then the order of precedence is the one provided in the above ordered list. Second, since these attributes are necessarily optional and any element containing them, by necessity nillable, such an element may not have any contents at all; if this is the case, the contents should be considered UNKNOWN. Finally, there is no obligation for the client to go and substitute an element that is referenced through Xlink (though it should always be permissible). We will provide some standard libraries (of common coordinate systems and observatory locations; see examples and the STC document) with a standard naming convention; a client that is familiar with such libraries may use its inside knowledge instead.

6. Examples

We will show one major example, two fragments, and a small example. For more examples, see the STC standards document.

6.1. Image Observed from Moving Spacecraft

The *ObsDataLocation* element is the container for the complete metadata description of an observational dataset, in this case an image. It specifies the various required namespaces (for STC: *STC* and *Xlink* via the `xmlns` attributes, and the location of the STC schema is the `xsi:schemaLocation` attribute). *ObsDataLocation* contains two main components: *ObservatoryLocation* and *ObservationLocation*.

This (rather extensive) example shows how to do this for a spacecraft observatory, using an orbit ephemeris file[1].

```
<?xml version="1.0" encoding="UTF-8"?>
<ObsDataLocation xmlns="http://www.ivoa.net/xml/STC/stc-v1.30.xsd"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ivoa.net/xml/STC/stc-v1.30.xsd
http://www.ivoa.net/xml/STC/stc-v1.30.xsd">
```

The location of the observatory is specified in *ObservatoryLocation* by a Chandra orbit ephemeris file and an associated coordinate system – geocentric 3-D Cartesian aligned with FK5. Identifier strings are in principle arbitrary, but should be helpful.

```
   <ObservatoryLocation id="CXO">
```

First the coordinate system; spatial coordinates are geodetic, referenced to the center of the earth (`GEOCENTER`), but times are measured locally (`TOPOCENTER`):

```
      <AstroCoordSystem id="TT-TOPO-FK5-GEO">
         <TimeFrame>
             <TimeScale>TT</TimeScale>
             <TOPOCENTER/>
         </TimeFrame>
         <SpaceFrame>
             <FK5>
                <Equinox>J2000.0</Equinox>
             </FK5>
             <GEOCENTER/>
```

---

[1] The data file may be found at:
`ftp://cdaftp.cfa.harvard.edu/pub/science/ao02/cat5/1952/primary/acisf01952N002_cntr_img2.fits.gz`

```
            <CARTESIAN coord_naxes="3"/>
         </SpaceFrame>
      </AstroCoordSystem>
```

Then the coordinates, for which one is referred to columns `Time`, `X`, `Y`, `Z`, `Vx`, `Vy`, `Vz` in HDU 1 of a FITS orbit ephemeris file:

```
      <AstroCoords coord_system_id="TT-TOPO-FK5-GEO">
         <CoordFile>
            <FITSFile hdu_name="ORBITEPHEM" hdu_num="1">
   ftp://cda.cfa.harvard.edu/pub/bary/orbitf129297900N001_eph0.fits
            </FITSFile>
            <FITSTime>
               <Value>Time</Value>
            </FITSTime>
            <FITSPosition>
               <Value>X,Y,Z</Value>
            </FITSPosition>
            <FITSVelocity>
               <Value>Vx,Vy,Vz</Value>
            </FITSVelocity>
         </CoordFile>
      </AstroCoords>
   </ObservatoryLocation>
```

The location represented by the observation's data and the volume it occupies in coordinate space is specified in *ObservationLocation* by a set of coordinates and their properties (errors, resolutions, etc., defined using the `AstroCoords` element) and the area or interval covered in each coordinate (defined using the `AstroCoordArea` element) with the associated coordinate system (defined using the `AstroCoordSystem` element). Note that this coordinate system includes a generic coordinate for brightness (the `PhotonCounts` frame).

```
   <ObservationLocation id="M81">
```

Again, first the coordinate system:

```
      <AstroCoordSystem id="TT-ICRS-ENERGY-TOPO">
         <CoordFrame id="counts">
            <Name>PhotonCounts</Name>
            <ScalarRefFrame>
               <Scale>1.0</Scale>
            </ScalarRefFrame>
            <CARTESIAN coord_naxes="1"/>
         </CoordFrame>
         <TimeFrame>
            <TimeScale>TT</TimeScale>
            <TOPOCENTER/>
```

```
        </TimeFrame>
        <SpaceFrame id="spaceFrame">
            <ICRS/>
            <TOPOCENTER/>
            <SPHERICAL coord_naxes="2"/>
        </SpaceFrame>
        <SpectralFrame>
            <TOPOCENTER/>
        </SpectralFrame>
      </AstroCoordSystem>
```

Next are the coordinates and their properties:

```
      <AstroCoords coord_system_id="TT-ICRS-ENERGY-TOPO">
```

Rather trivial for the photon count independent variable:

```
        <ScalarCoordinate unit="count/s" frame_id="counts">
            <Resolution>1</Resolution>
        </ScalarCoordinate>
```

The time of observing is specified in MJD, the exposure was 49664 seconds long:

```
        <Time unit="s">
            <TimeInstant>
               <MJDTime>52311.266</MJDTime>
            </TimeInstant>
            <Resolution>49664</Resolution>
            <PixSize>49664</PixSize>
        </Time>
```

Here are the center position, the astrometric errors, resolution, and pixel sizes (in degrees) in ICRS positions as seen from the spacecraft:

```
        <Position2D unit="deg">
            <Value2 id="Center">
               <C1>350.9171576</C1>
               <C2>58.79377795</C2>
            </Value2>
            <Error2Radius>0.0003 </Error2Radius>
            <Resolution2Radius>0.00015</Resolution2Radius>
            <PixSize2>
               <C1>0.0001367</C1>
               <C2>0.0001367</C2>
            </PixSize2>
        </Position2D>
```

The spectral coordinate, in keV:

```
        <Spectral unit="keV">
            <Value>2.5</Value>
            <Resolution>7</Resolution>
            <PixSize>7</PixSize>
        </Spectral>
    </AstroCoords>
```

And the volume in coordinate space taken up by the image:

```
    <AstroCoordArea id="Cas-A"
        coord_system_id="TT-ICRS-ENERGY-TOPO">
```

The photon counts cannot be negative; note the *frame_id* attribute:

```
        <CoordScalarInterval frame_id="counts">
            <LoLimit>0.0</LoLimit>
        </CoordScalarInterval>
```

The precise time interval during which the observation was made; the precise exposure time might be extracted by multiplying the difference between stop and start time by the fill factor (if present; default=1) – this would be easier if the times were expressed in MJD, rather than ISO-8601:

```
        <TimeInterval>
            <StartTime>
                <ISOTime>2002-02-06T06:23:17</ISOTime>
            </StartTime>
            <StopTime>
                <ISOTime>2002-02-06T20:39:48</ISOTime>
            </StopTime>
        </TimeInterval>
```

The spatial area on the sky is expressed as a polygon:

```
        <Polygon unit="deg">
            <Vertex>
                <Position>
                    <C1>350.99584</C1>
                    <C2>58.74772</C2>
                </Position>
            </Vertex>
            <Vertex>
                <Position>
                    <C1>350.72609</C1>
                    <C2>58.74760</C2>
                </Position>
```

```
            </Vertex>
            <Vertex>
               <Position>
                  <C1>350.72532</C1>
                  <C2>58.88741</C2>
               </Position>
            </Vertex>
            <Vertex>
               <Position>
                  <C1>350.99616</C1>
                  <C2>58.88753</C2>
               </Position>
            </Vertex>
         </Polygon>
```

And the exact spectral band; in this example it just provides the bounds, but in principle, through the use of multiple intervals with specific fill factors, the filter shape could be encoded:

```
         <SpectralInterval unit="keV">
            <LoLimit>0.5</LoLimit>
            <HiLimit>7.0</HiLimit>
         </SpectralInterval>
      </AstroCoordArea>
   </ObservationLocation>
```

The image is represented as a pixel array. The *PixelSpace* (below) specifies the properties of this pixel array and the way it is related to the world coordinates specified above. It consists of a pixel coordinate system that defines the pixel axes and their transformation on the sky (WCS) and a pixel coordinate area that defines the array's dimensions. Note the use of *ref_frame_id* to tie pixel interval to the correct frame (via the `<SpaceFrame id="spaceFrame">` in the `AstroCoordSystem` above); in this case it is not strictly required since there is only one frame, but in the general case this is the mechanism used to avoid ambiguity.

The pixel coordinate system consists of one 2-dimensional frame and specifies the axis order:

```
   <PixelSpace>
      <PixelCoordSystem id="Cas-APix">
         <PixelCoordFrame id="spacepix" axis1_order="1"
            axis2_order="2" ref_frame_id="spaceFrame">
```

The reference frame specifies a tangent projection with 0.0001367 degrees per pixel:

```
            <Cart2DRefFrame projection="TAN">
               <Transform2 unit="deg">
                  <C1>0.0001367</C1>
```

```
                <C2>0.0001367</C2>
             </Transform2>
          </Cart2DRefFrame>
```

The WCS values for the reference pixel are provided about two pages ago and referenced through `idref="Center"`.

```
          <CoordRefPos>
             <Vector2DCoordinate unit="deg">
                <Value2 idref="Center" xsi:nil="true"/>
             </Vector2DCoordinate>
          </CoordRefPos>
```

The *Flavor* of the frame, 2-dimensional Cartesian:

```
          <CARTESIAN coord_naxes="2"/>
```

And the reference pixel:

```
          <ReferencePixel>
             <Pixel2D>
                <Name1>RA</Name1>
                <Name2>Dec</Name2>
                <Value2>
                   <C1>299.69</C1>
                   <C2>337.84</C2>
                </Value2>
             </Pixel2D>
          </ReferencePixel>
       </PixelCoordFrame>
    </PixelCoordSystem>
```

The pixel area defines a 1025 by 1024 array; it points back to PixelCoordFrame "`spacepix`" – although there is no confusion in this case since there is only one pixel coordinate frame, this cannot be assumed to be the case in general:

```
    <PixelCoordArea coord_system_id="Cas-APix" id="Cas-APixImage">
       <PixelCoord2VecInterval frame_id="spacepix">
          <LoLimit2Vec>
             <C1>1</C1>
             <C2>1</C2>
          </LoLimit2Vec>
          <HiLimit2Vec>
             <C1>1025</C1>
             <C2>1024</C2>
          </HiLimit2Vec>
       </PixelCoord2VecInterval>
    </PixelCoordArea>
```

```
   </PixelSpace>
</ObsDataLocation>
```

6.2. A Ground-based Observatory Location

For a ground-based observatory one can specify the *ObservatoryLocation* directly:

```
   <ObservatoryLocation id="Arecibo">
      <AstroCoordSystem id="TT-GEOD-TOPO">
         <TimeFrame>
             <TimeScale>TT</TimeScale>
             <TOPOCENTER/>
         </TimeFrame>
         <SpaceFrame>
             <GEO_D/>
             <TOPOCENTER/>
             <SPHERICAL coord_naxes="3"/>
         </SpaceFrame>
      </AstroCoordSystem>
      <AstroCoords coord_system_id="TT-GEOD-TOPO">
         <Position3D>
             <Value3>
                <C1 pos_unit="deg">293.2469</C1>
                <C2 pos_unit="deg">18.3435</C2>
                <C3 pos_unit="m">497</C3>
             </Value3>
         </Position3D>
      </AstroCoords>
   </ObservatoryLocation>
```

6.3. Using the Libraries through Xlink

We are providing two libraries of standard elements. References to these standard objects are made through Xlinks which leave it up to the client to (a) look it up and do a substitution, (b) know what the contents of the standard object are and not bother looking it up, or (c) ignore the object altogether. The first library contains standard coordinate systems. Using that library, the example above could have been written thus:

```
   <ObservatoryLocation id="Arecibo">
      <AstroCoordSystem id="TT-GEOD-TOPO" xlink:type="simple"
         xlink:href="ivo://STClib/CoordSys#TT-GEOD-TOPO"/>
      <AstroCoords coord_system_id="TT-GEOD-TOPO">
         <Position3D>
             <Value3>
                <C1 pos_unit="deg">293.2469</C1>
                <C2 pos_unit="deg">18.3435</C2>
                <C3 pos_unit="m">497</C3>
```

```
            </Value3>
         </Position3D>
      </AstroCoords>
   </ObservatoryLocation>
```

The second library can be referenced using an Xlink attribute and contains known observatory locations and allows us to write the same example simply as:

```
   <ObservatoryLocation id="Arecibo" xlink:type="simple"
      xlink:href="ivo://STClib/Observatories#Arecibo"/>
```

6.4. A Simple Orbit Representation

Finally, a simple example of how to specify a (parabolic, in this case) orbit for C/2006 A1, taken from IAU Circular 8653, using Keplerian orbital elements:

```
<?xml version="1.0" encoding="UTF-8"?>
<CatalogEntryLocation
xmlns="http://www.ivoa.net/xml/STC/stc-v1.30.xsd"
xsi:schemaLocation="http://www.ivoa.net/xml/STC/stc-v1.30.xsd
http://www.ivoa.net/xml/STC/stc-v1.30.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xlink="http://www.w3.org/1999/xlink">
   <AstroCoordSystem xsi:nil="true" xlink:type="simple"
      xlink:href="ivo://STClib/CoordSys#TDB-ECLIPTIC-BARY"
      id="TDB-ECLIPTIC-BARY"/>
   <AstroCoords coord_system_id="TDB-ECLIPTIC-BARY" >
      <Time>
         <TimeInstant>
            <ISOTime>2006-01-05T00:00:00</ISOTime>
         </TimeInstant>
      </Time>

      <Orbit>
         <q unit="AU">0.56748</q>
         <e>1.0</e>
         <i unit="deg">93.230</i>
         <Node unit="deg">212.275</Node>
         <Aop unit="deg">350.499</Aop>
         <T><ISOTime>2006-02-22T16:06:14</ISOTime></T>
      </Orbit>
   </AstroCoords>
</CatalogEntryLocation>
```

Acknowledgments

Many people have contributed to the shaping of this standard. In particular, I would like to mention David Berry, Ray Plante, Jonathan McDowell, Ed Shaya, Gerard Lemson, Alex Szalay, Tamas Budavari, François Bonnarel, and Michael Fitzpatrick. I am grateful for the support of the Chandra X-ray Center which is operated by the SAO for NASA under contract NAS 8-03060 and for the support, at the early stages of this project, of the Université Louis Pasteur and the Centre de Données astronomiques de Strasbourg.

Useful Links

The STC standards document (latest public version): http://www.ivoa.net/Documents/latest/STC.html [Accessed 19 Dec 2006]
The STC schema: http://www.ivoa.net/xml/STC/stc-v1.30.xsd [Accessed 19 Dec 2006]
The author maintains a current development site with more documentation and examples: http://hea-www.cfa.harvard.edu/~arots/nvometa/STC/ [Accessed 19 Dec 2006]
The Xlink standard: http://www.w3.org/TR/xlink [Accessed 19 Dec 2006]
The Xlink schema: http://www.ivoa.net/xml/Xlink/xlink.xsd [Accessed 19 Dec 2006]
Information on Healpix coordinates and FITS WCS in general: http://fits.gsfc.nasa.gov/fits_wcs.html
A general reference on coordinate systems: Explanatory Supplement to the Astronomical Almanac (University Science Books 1992, Ed. P. K. Seidelmann)
Timescales: http://tycho.usno.navy.mil/systime.html, http://cxc.harvard.edu/contrib/arots/time/time_tutorial.html