



*International*  
*Virtual*  
*Observatory*  
*Alliance*

## Coordinates Data Model

IVOA DM WG Internal Note  
2006-09-14 (Rev 1)

**Working Group:** <http://www.ivoa.net/twiki/bin/view/IVOA/IvoaDataModel>  
**Authors:**  
Jonathan McDowell

### Abstract

The STC Model is an XML-specific implementation model. I present an abstract data model which can be mapped to STC. This model will serve as a guide to non-XML serializations of STC. It is based on STC V1.30 (Aug 2006).

### Status of this document

This is a Working Group Internal Note. It's a somewhat incomplete draft. I acknowledge helpful discussions with Arnold Rots, the creator of STC.

# Contents

<b>1</b>	<b>The STC model</b>	<b>3</b>
1.1	Existing STC documentation . . . . .	3
1.2	Summary of the model . . . . .	3
1.3	Differences from the XML version . . . . .	3
<b>2</b>	<b>CoordSys and CoordFrame</b>	<b>9</b>
2.1	CoordSys . . . . .	9
2.2	CoordFrame objects . . . . .	9
2.3	CoordSys and CoordFrame data elements . . . . .	9
<b>3</b>	<b>Coordinates</b>	<b>15</b>
3.1	Coordinate . . . . .	15
3.2	Basic Coordinates: SpectralCoord and Redshift . . . . .	15
3.3	Beyond Basic Coordinates . . . . .	16
3.3.1	CoordValue . . . . .	16
3.3.2	Error, Resolution, Size, PixSize . . . . .	17
3.4	Specialized Coordinates: Position and Velocity . . . . .	18
3.5	Specialized Coordinates: AstronTime . . . . .	21
3.6	Coordinates in FITS files . . . . .	22
3.7	Keplerian Coordinates . . . . .	23
3.8	Coordinate Sets . . . . .	23
<b>4</b>	<b>Auxiliary types</b>	<b>25</b>
4.1	Units . . . . .	25
4.2	PositionAngles . . . . .	25
4.3	Intervals . . . . .	25
4.4	Regions . . . . .	27
4.5	Areas . . . . .	29
<b>5</b>	<b>High level STC objects</b>	<b>31</b>
<b>6</b>	<b>Frequently Asked Questions</b>	<b>33</b>
6.1	Why do you need a SpectralFrame in STC? . . . . .	33
6.2	Why do you need separate SpectralFrame and RedshiftFrame in STC? . . . . .	33
6.3	Barycenter corrections: why might you need a TimeRefDirection? . . . . .	33

# 1 The STC model

## 1.1 Existing STC documentation

## 1.2 Summary of the model

The STC model is intended as a description of the coverage of a dataset, survey or observation. It includes three main objects:

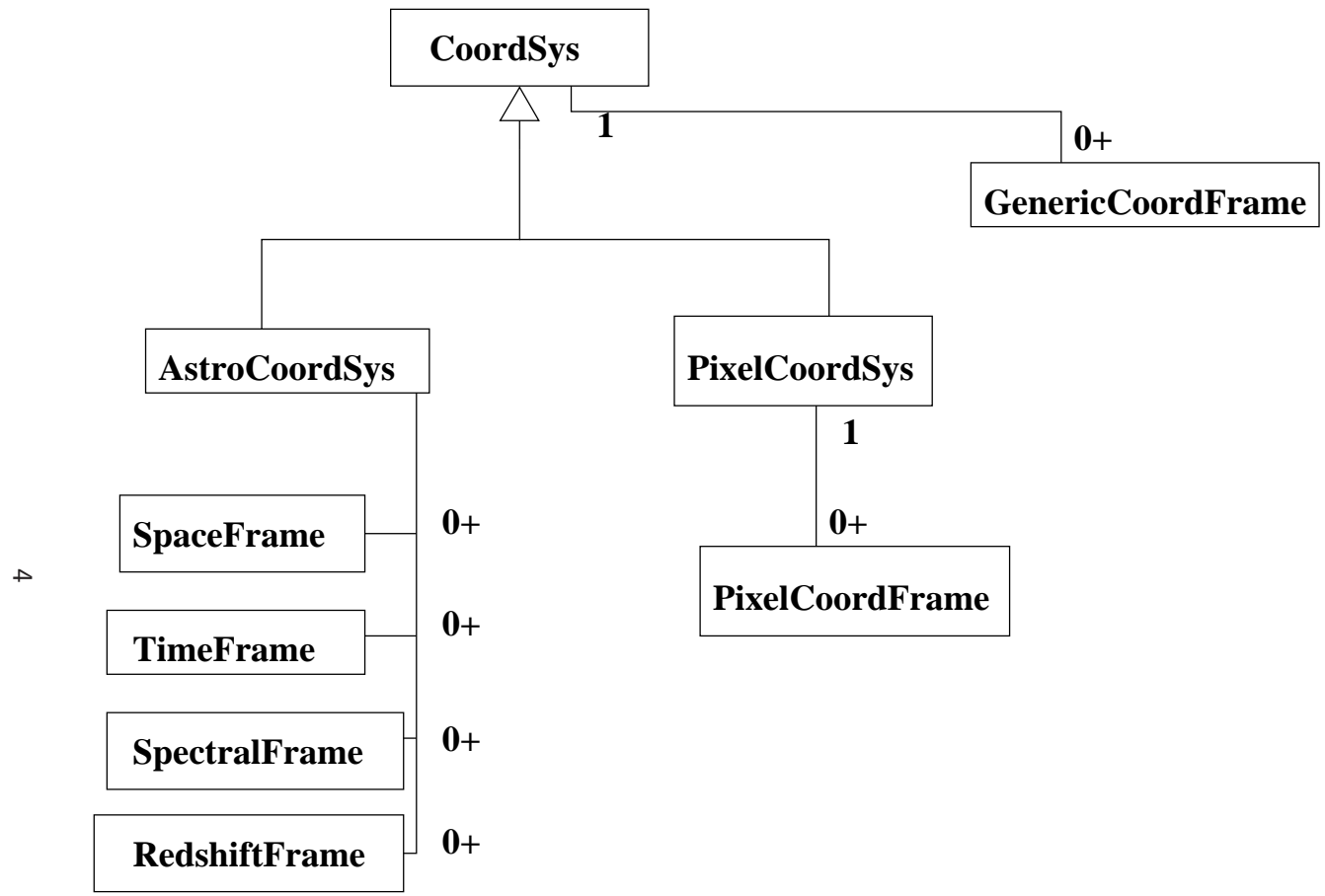
- The coordinate system object **CoordSys**, identifying the parameter space of the coverage
- The coordinate area object **CoordArea**, defining the region within that space which contains valid data.
- The coordinate location object **Coords**, defining a reference point within that space that may be used to approximately characterize the data.

The STC model draws particular attention to the connection between time, spatial position, spatial velocity, redshift and spectral coordinates, whose definitions are intertwined.

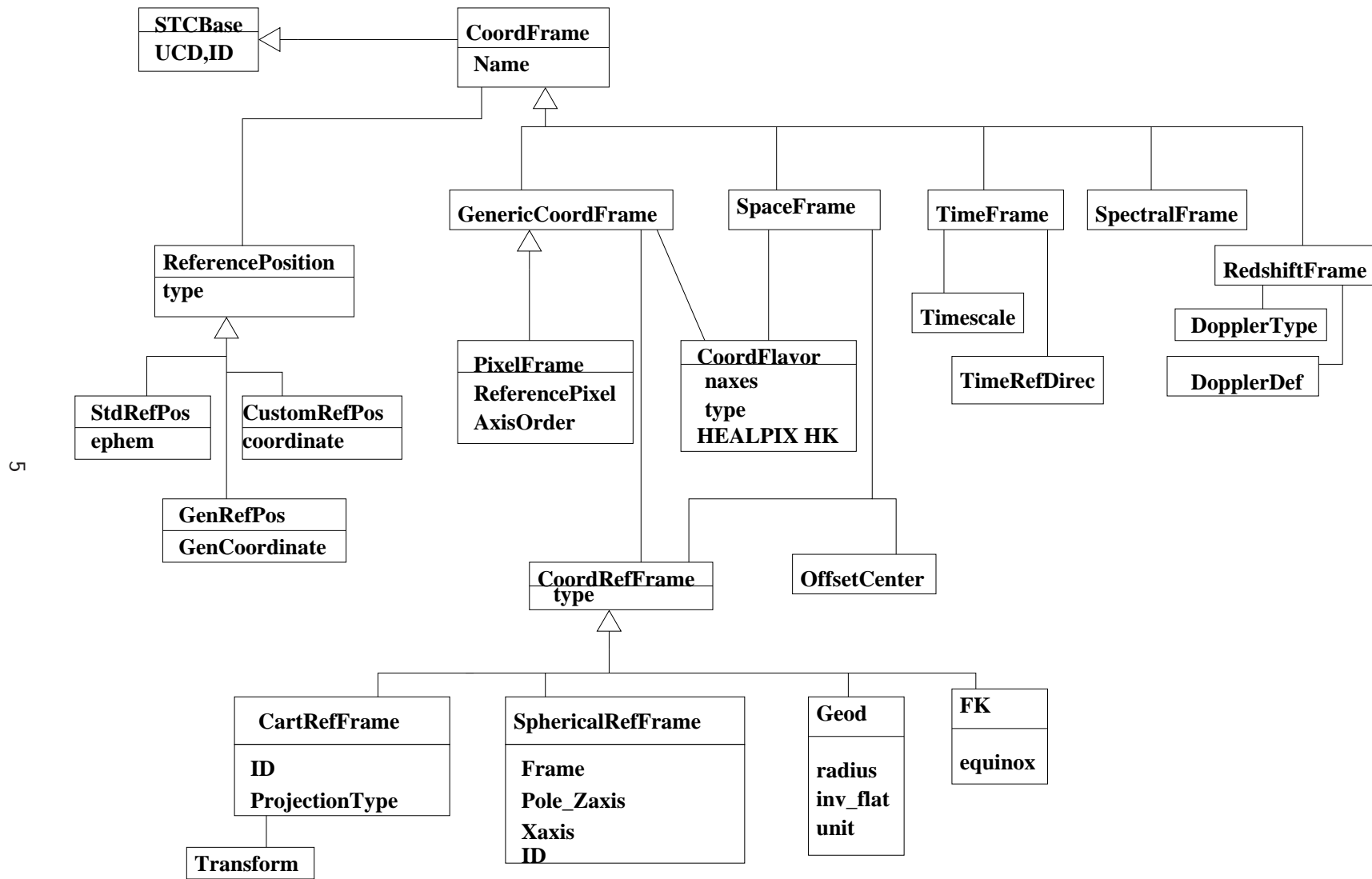
The model also defines an STC object which aggregates a CoordSys, a CoordArea and a Coords into a single thing, but I expect they will often be used separately. In addition, the STC model envisages a separate instance of the STC object to describe the observation coverage (on the sky) and the observatory location.

## 1.3 Differences from the XML version

One general comment I have is that any place where an STC element must have one of a predefined set of enumerated values (e.g. the StdPole which can be one of EQUATORIAL, ECLIPTIC, etc.), we are defining a controlled vocabulary of astronomical concepts. I would argue that any such controlled vocabulary should be included by the IVOA Semantics group as part of the official data dictionary for astronomical concepts in the VO. I will include these in a separate document. At the abstract data model level, these enumerated types are replaced by a 'type' field in the model: we do not specify whether the field maps to a simple string or to an enumerated type, that's for the specific implementation formats to define. If the type is empty, the value 'UNKNOWN' is assumed.



4



Field		Meaning	Req	Default
Coordinate system metadata fields				
<b>CoordSys.ID</b>		ID for this CoordSys	R	None
<b>CoordFrame.Name</b>		Name of CoordFrame	R	None
<b>CoordFrame.ID</b>		ID for this CoordFrame	R	None
CoordFrame.UCD		UCD for this CoordFrame	O	None
<b>CoordFrame.ReferencePosition.Type</b>		Origin of CoordFrame	R	None
CoordFrame.StdRefPos.Ephem		Planetary ephemeris used	O	N/A
CoordFrame.GenRefPos.GenCoordinate		Origin for custom frame	O	N/A
CoordFrame.CustomRefPos.Coordinate		Origin for custom frame	O	N/A
<b>CoordFrame.CoordFlavor.Type</b>		Cartesian, polar, etc.	R	
CoordFrame.CoordFlavor.Naxes		Number of axes	O	2
CoordFrame.CoordFlavor.healpix_H		HEALPIX H	O	4
CoordFrame.CoordFlavor.healpix_K		HEALPIX K	O	3
<b>CoordFrame.CoordRefFrame.Type</b>		RefFrame type	R	
CoordFrame.GeodRefFrame.radius		Geodetic radius	O	
CoordFrame.GeodRefFrame.inv_flat		Inverse flattening	O	
CoordFrame.GeodRefFrame.radius_unit	Unit for radius	O	m	
CoordFrame.FKRefFrame.equinox	B1950 or J2000, etc	O		
CoordFrame.SphericalRefFrame.ID	ID for reference frame	O		
CoordFrame.SphericalRefFrame.Frame	Same as CoordFrame.Name?	O		
CoordFrame.SphericalRefFrame.Pole_Zaxis	Pole coords in ref system	O		
CoordFrame.SphericalRefFrame.Xaxis		O		
CoordFrame.CartRefFrame.ID		Pole coords in ref system	O	
CoordFrame.CartRefFrame.ProjectionType		Projection type	O	
CoordFrame.CartRefFrame.Transform		(transform parameters)		
PixelFrame.ReferencePixel		(pixel coords)		
PixelFrame.AxisOrder		Axis number of this frame		

<b>SpaceFrame.Name</b>		Name of SpaceFrame	R	None
<b>SpaceFrame.ID</b>		ID for this SpaceFrame	R	None
SpaceFrame.UCD		UCD for this SpaceFrame	O	None
<b>SpaceFrame.ReferencePosition.Type</b>		Origin of SpaceFrame	R	None
SpaceFrame.StdRefPos.Ephem		Planetary ephemeris used	O	N/A
SpaceFrame.GenRefPos.GenCoordinate		Origin for custom frame	O	N/A
SpaceFrame.CustomRefPos.Coordinate		Origin for custom frame	O	N/A
SpaceFrame.OffsetCenter		Offset center	O	
<b>SpaceFrame.CoordFlavor.Type</b>		Cartesian, polar, etc.	R	
SpaceFrame.CoordFlavor.Naxes		Number of axes	O	2
SpaceFrame.CoordFlavor.healpix_H		HEALPIX H	O	4
SpaceFrame.CoordFlavor.healpix_K		HEALPIX K	O	3
SpaceFrame.GeodRefFrame.radius		Geodetic radius	O	
SpaceFrame.GeodRefFrame.inv_flat		Inverse flattening	O	
SpaceFrame.GeodRefFrame.radius_unit	Unit for radius	O		m
SpaceFrame.FKRefFrame.equinox	B1950 or J2000, etc	O		
SpaceFrame.SphericalRefFrame.ID	ID for reference frame	O		
SpaceFrame.SphericalRefFrame.Frame	Same as SpaceFrame.Name?	O		
SpaceFrame.SphericalRefFrame.Pole_Zaxis	Pole coords in ref system	O		
SpaceFrame.SphericalRefFrame.Xaxis		O		
SpaceFrame.CartRefFrame.ID		Pole coords in ref system	O	
SpaceFrame.CartRefFrame.ProjectionType		Projection type	O	
SpaceFrame.CartRefFrame.Transform		(transform parameters)		

<b>TimeFrame.Name</b>	Name of TimeFrame	R	None
<b>TimeFrame.ID</b>	ID for this TimeFrame	R	None
TimeFrame.UCD	UCD for this TimeFrame	O	None
<b>TimeFrame.ReferencePosition.Type</b>	Origin of TimeFrame	R	None
TimeFrame.StdRefPos.Ephem	Planetary ephemeris used	O	N/A
TimeFrame.GenRefPos.GenCoordinate	Origin for custom frame	O	N/A
TimeFrame.CustomRefPos.Coordinate	Origin for custom frame	O	N/A
TimeFrame.Timescale	Timescale	O	TT
TimeFrame.TimeRefDirection	Reference direction	O	
<b>SpectralFrame.Name</b>	Name of SpectralFrame	R	None
<b>SpectralFrame.ID</b>	ID for this SpectralFrame	R	None
SpectralFrame.UCD	UCD for this SpectralFrame	O	None
<b>SpectralFrame.ReferencePosition.Type</b>	Origin of SpectralFrame	R	None
SpectralFrame.StdRefPos.Ephem	Planetary ephemeris used	O	N/A
SpectralFrame.GenRefPos.GenCoordinate	Origin for custom frame	O	N/A
SpectralFrame.CustomRefPos.Coordinate	Origin for custom frame	O	N/A
<b>RedshiftFrame.Name</b>	Name of RedshiftFrame	R	None
<b>RedshiftFrame.ID</b>	ID for this RedshiftFrame	R	None
RedshiftFrame.UCD	UCD for this RedshiftFrame	O	None
<b>RedshiftFrame.ReferencePosition.Type</b>	Origin of RedshiftFrame	R	None
RedshiftFrame.StdRefPos.Ephem	Planetary ephemeris used	O	N/A
RedshiftFrame.GenRefPos.GenCoordinate	Origin for custom frame	O	N/A
RedshiftFrame.CustomRefPos.Coordinate	Origin for custom frame	O	N/A
RedshiftFrame.DopplerValueType	Velocity or redshift	O	VELOCITY
RedshiftFrame.DopplerDefinition	Optical, radio, rel.	O	

## 2 CoordSys and CoordFrame

### 2.1 CoordSys

The CoordSys object consists of the aggregation of several CoordFrame objects, each for a separate set of axes. CoordFrame is subclassed for particular kinds of axis, such as Space, Time etc.

The CoordSys object specializes to an AstroCoordSys subclass which consists of at least one CoordFrame object for time, space, spectral, and redshift (doppler) coordinates, as well as optional generic axes.

### 2.2 CoordFrame objects

The CoordFlavor specifies the type of the coordinates (2D spherical, 3D cartesian, etc.). The HEALPIX system has parameters for the coordinate type which are given in this object.

### 2.3 CoordSys and CoordFrame data elements

Several of these elements are of type AstroCoords, which is described later in the document. Here I give generic CoordFrame utypes, which can be specialized to TimeFrame, SpectralFrame etc. as needed; see the list of utypes above.

- **CoordSys.ID** A token assigning an ID to the entire coordinate system. This ID will be used by other data objects to refer to this coordinate frame instance. REQUIRED.
- **CoordFrame.Name** A token assigning a name to the coordinate frame. REQUIRED.
- **CoordFrame.ID** A token assigning an ID to the coordinate frame in the current document and those including it. This ID will be used by other data objects to refer to this coordinate frame instance. REQUIRED.
- **CoordFrame.ReferencePosition.Type**: the token identifying the space-time origin and rest frame of the coordinate frame. REQUIRED.

Note the special token values UNKNOWN, RELOCATABLE (a relative origin for simulations), TOPOCENTER (referring to the ObservatoryLocation element), and CUSTOM (referring to the CoordFrame.ReferencePosition.CoordOrigin which gives the frame origin wrt another system). See the table.

- **CoordFrame.RefFrame.Type**: the token identifying the coordinate system and frame. REQUIRED.

The usual value is ICRS, which is the standard post-Hipparcos-era RA and Dec; it is within 12 millarcseconds of J2000/FK5. The other systems have different orientation and rotation:

- FK4 and FK5 have a pole whose position depends on time with respect to the ICRS system. This pole also depends on a parameter called the equinox; by convention FK4 and FK5 use Besselian and Julian epochs (defaults B1950.0 and J2000.0) respectively to label the equinox. The ecliptic system also has an equinox parameter.
- Planetary coordinate systems have a pole fixed in the planet, and so require an ephemeris to transform them to ICRS.

- the galactic systems are considered to be a fixed (time-independent) rotation wrt ICRS.
  - The generic BODY system is for 2D coordinates on any body and does not have a defined transform to ICRS.
  - The AZEL system is based at the topocenter and needs the location of the topocenter to define a transform to ICRS.
- **CoordFrame.ReferencePosition.Ephem**: optional token identifying the planetary ephemeris used in tying the time/position data to the coordinate frame. You need this if your data have been transformed to a position other than the one they've been observed in, e.g. by applying a barycenter timing correction.  
Values are "JPL-DE200", "JPL-DE405", or absent. OPTIONAL when not relevant.
  - **CoordFrame.ReferencePosition.CoordOrigin**: astroCoordsType variable containing reference origin with respect to another coordinate system. Used if Type is CUSTOM.
  - **CoordFrame.CoordFlavor.Type**. Specifies whether the coordinates in the frame are cartesian, polar, etc. See table. REQUIRED.
  - **CoordFrame.CoordFlavor.Naxes**. Number of axes in the coordinate frame. Only values 1, 2 and 3 are supported; default is 2.
  - **CoordFrame.CoordFlavor.healpix\_H** Used if type is HEALPIX; value of HEALPIX H, default is 4.
  - **CoordFrame.CoordFlavor.healpix\_K** Used if type is HEALPIX; value of HEALPIX K, default is 3.
  - **CoordFrame.RefFrame.Radius**: Radius parameter for geodetic spheroid; this is a coordinate frame parameter for the geodetic systems. Default unit is meters. Default value is IAU 1976 spheroid value.
  - **CoordFrame.RefFrame.Inv\_Flattening**: Dimensionless inverse flattening parameter for geodetic system spheroid. Default is IAU 1976 spheroid value.
  - **CoordFrame.RefFrame.Unit**: : Unit for CoordFrame.GeodRefFrame.Radius. Default is "m" (meters).
  - **CoordFrame.RefFrame.Equinox**: Equinox string for FK frame. Used when SpaceRefFrame is FK4, FK5 or ECLIPTIC. Usually "B1950.0" or "J2000.0".
  - **CoordFrame.Epoch**: Epoch string, used for spatial frame. OPTIONAL, no default. Implies that positions given in this coordinate frame have a particular epoch.
  - **CoordFrame.Frame**: String defining custom frame. Provides a name for the custom space frame; probably superfluous given the CoordFrame.Name field.
  - **CoordFrame.Pole\_Zaxis**: AstroCoords instance giving coordinates of pole of custom system in terms of reference system. Used for custom space frames.
  - **CoordFrame.Xaxis**: AstroCoords instance giving coordinates of X-axis direction of custom system in terms of reference system. Used for custom space frames.

- **CoordFrame.OffsetCenter:** CoordValue instance. Optional. Defines a frame in which the coordinates are offsets from this reference direction. In some data, positions are quoted as offsets from a given reference location: for example, positions relative to the center of a galaxy. Intended to apply to RA and Dec positions; coordinate tuples will be interpreted as arithmetic differences. Example: if the OffsetCenter is (RA= 200.0 deg, Dec = -80.0 deg) and the coordinate tuple is (0.02, 0.01) then the position is (RA = 200.02, Dec = -79.99); no ' $\cos \delta$ ' term is applied to the RA difference.

Note: the XML implementation uses a container type holding a CoordValue, rather than just using a CoordValue itself, due to the limitations of XML Schema. See later for the definition of the CoordValue type.

- **CoordFrame.TimeScale:** string defining timescale. Default is "TT"; see table.
- **CoordFrame.TimeRefDirection:** AstroCoords instance. See FAQ.
- **CoordFrame.DopplerValueType:** Values are "VELOCITY" or "REDSHIFT". Default is VELOCITY. Specifies whether redshift is quoted in velocity or dimensionless units.
- **CoordFrame.DopplerDefinition:** Values are OPTICAL, RADIO or RELATIVISTIC. Specifies the relationship between the observed redshift and a quoted velocity. Only relevant for redshift frames quoted in velocity.
- There is also support for transforms and projections, not covered in this rev of this document.

Token	Meaning	Note
UNKNOWN	Unknown origin	
RELOCATABLE	Relative origin	Suitable for simulations
CUSTOM	Origin specified wrt another system	
TOPOCENTER	Location of the observing device	(telescope)
BARYCENTER	Solar system barycenter	
HELIOCENTER	Center of the Sun	
GEOCENTER	Center of the Earth	
EMBARYCENTER	Earth-Moon barycenter	
MOON	Center of the Moon	
MERCURY	Center of Mercury	
VENUS	Center of Venus	
MARS	Center of Mars	
JUPITER	Center of Jupiter	
SATURN	Center of Saturn	
URANUS	Center of Uranus	
NEPTUNE	Center of Neptune	
PLUTO	Center of Pluto	
LSRK	Kinematic local standard of rest	Redshift frame only
LSRD	Dynamic local standard of rest	Redshift frame only
GALACTIC_CENTER	Center of the Galaxy	
LOCAL_GROUP_CENTER	Barycenter of the Local Group	

Table 1: Allowed values for CoordFrame.ReferencePosition.Type

Token	Meaning	Note
SPHERICAL	Spherical coords longitude, latitude, radius	Also 2D lon,lat
CARTESIAN	1,2,or3D cartesian tuple	
UNITSPHERE	Direction cosine 3-tuple, norm is 1	
POLAR	2D polar coords radius, angle	
HEALPIX	2D Healpix coordinates	
STRING	String coords (e.g. Stokes)	

Table 2: Allowed values for CoordFrame.CoordFlavor.Type

Token	Meaning	Parameter(s)
UNKNOWN	Unknown frame	
CUSTOM	Custom frame	Pole, axis
AZ.EL	Azimuth and elevation	
BODY	Generic body (eg planet)	
ICRS	The ICRS frame	
FK4	FK4	Equinox
FK5	FK5	Equinox
ECLIPTIC	Ecliptic l,b	Equinox
GALACTIC_I	Old galactic LI,BI	
GALACTIC_II	Galactic LII,BII	
GALACTIC	Same as GALACTIC_II	
SUPER.GALACTIC	SGL, SGB	
MAG	Geomagnetic ref frame	
GSE	Geocentric Solar Ecliptic	
GSM	Geocentric Solar Magnetic	
SM	Solar Magnetic	
HGC	Heliographic	
HEE	Heliocentric Earth Ecliptic	
HEEQ	Heliocentric Earth Equatorial	
HCI	Heliocentric Inertial	
HGS		
HPC		
HPR		
HGI		
HRTM		
HCD	Heliocentric of Date	
GEO_C	Geocentric corotating	
GEO_D	Geodetic ref frame	Spheroid
MERCURY_C	Corotating planetocentric	
VENUS_C	Corotating planetocentric	
LUNA_C	Corotating planetocentric	
MARS_C	Corotating planetocentric	
JUPITER_C.III	Corotating planetocentric	
SATURN_C.III	Corotating planetocentric	
URANUS_C.III	Corotating planetocentric	
NEPTUNE_C.III	Corotating planetocentric	
PLUTO_C	Corotating planetocentric	
MERCURY_G	Corotating planetographic	
VENUS_G	Corotating planetographic	
LUNA_G	Corotating planetographic	
MARS_G	Corotating planetographic	
JUPITER_G.III	Corotating planetographic	
SATURN_G.III	Corotating planetographic	
URANUS_G.III	Corotating planetographic	
NEPTUNE_G.III	Corotating planetographic	
PLUTO_G	Corotating planetographic	

Table 3: Allowed values for CoordFrame.RefFrame.Type

Token	Meaning	Note
LOCAL	Relocatable (simulation) time	
TT	Terrestrial Time	
UTC	Coordinated Universal Time	
ET	Ephemeris Time	
TDB	Barycentric dynamical time	
TCG	Terrestrial Coordinate Time	
TCB	Barycentric Coordinate Time	
TAI	International Atomic Time	
LST	Local Sidereal Time	
IAT	Same as TAI	
TEB	Same as TDB	
TDT	Same as TT	

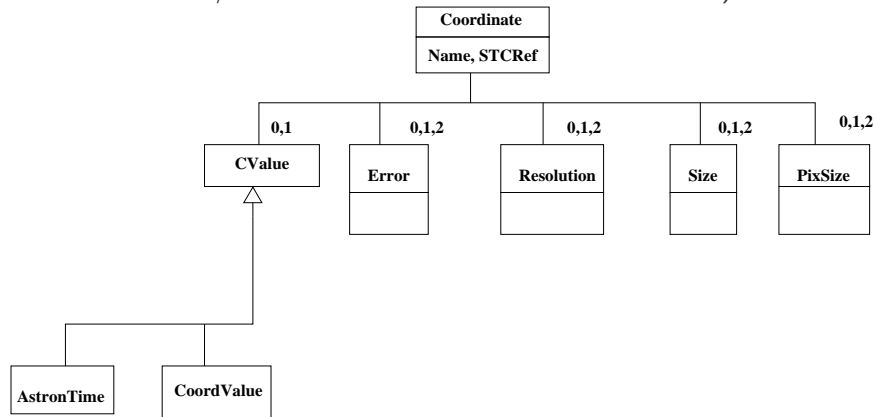
Table 4: Allowed values for CoordFrame.TimeScale

### 3 Coordinates

#### 3.1 Coordinate

The STC Coordinate is a container that in its simplest form just contains a CValue. The CValue can be a CoordValue (see below) or an AstronTime.

The Coordinate can also contain an Error (the uncertainty on the CValue), a Resolution (the (instrumental or effective) resolution on the coordinate in the current context), a PixSize (the size of the local instrumental pixel in coordinate units) or a Size (the observational field of view, or duration/extent of observation, in that coordinate).



Each of these additional entries can have its own unit, which is optional if the CValue is present; the unit defaults to that of the CValue. You can have a CValue and also have an Error with a different unit; this supports representing catalogs which have positions in degrees and errors quoted in arcseconds, without requiring conversions.

You don't have to have a CValue. For example, the Size would often be used on its own (a Coordinate containing only a Size).

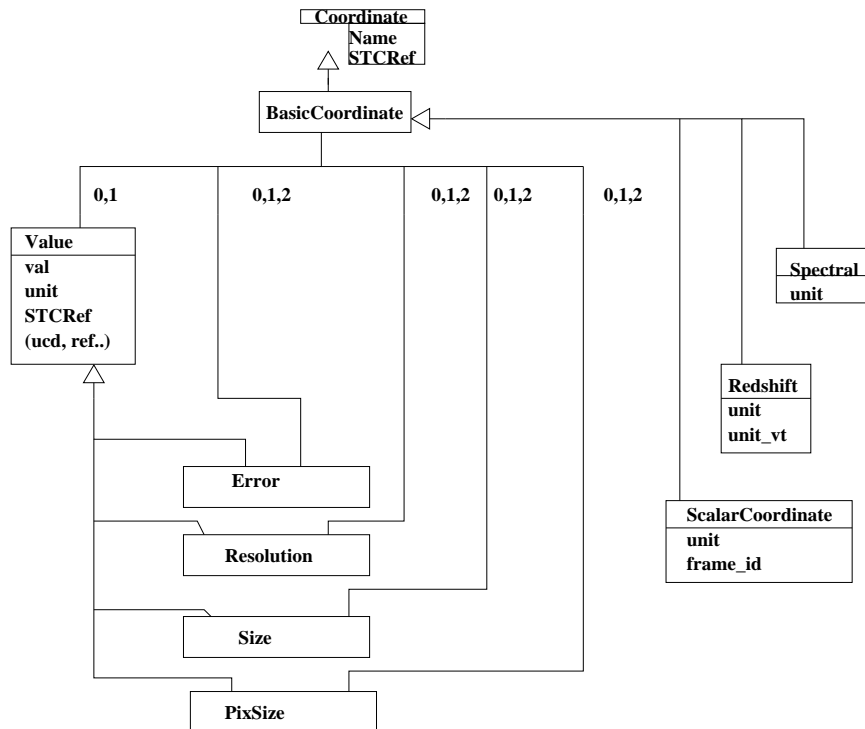
Pairs of Error, Resolution, PixSize or Size entries are also allowed and indicate the range of values - this is useful in entries summarizing a collection of observations.

While the CValue can hold a single number - see Basic Coordinates immediately below - it can also hold more complicated types, discussed later.

The STCRef object here is a shorthand for a set of attributes including a UCD, a string to identify the object in cross-references, and a cross-reference (either internal or external).

#### 3.2 Basic Coordinates: SpectralCoord and Redshift

STC defines a BasicCoordinate which restricts the elements to be 1-dimensional (scalar) values.



STC further defines a Redshift coordinate, a Spectral coordinate, and a generic ScalarCoordinate, which each have units in addition to the other BasicCoordinate properties. They differ only in the type of units permitted; note that even the BasicCoordinate can specify units in each of its components (Value, Error etc.).

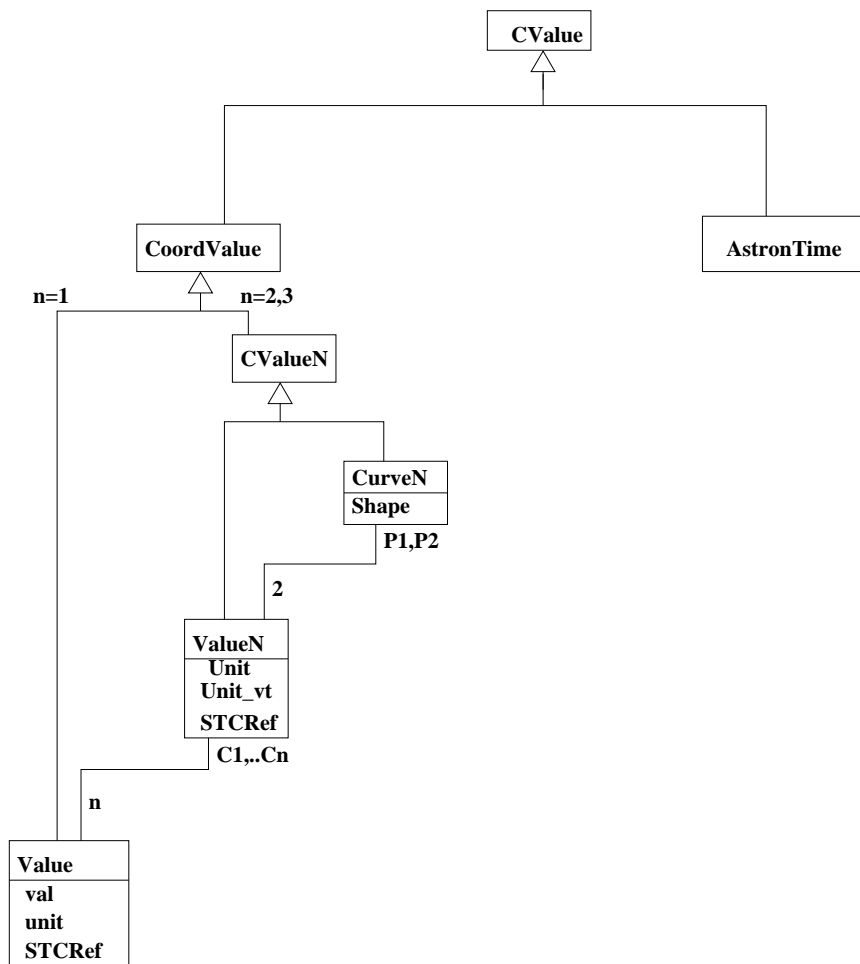
### 3.3 Beyond Basic Coordinates

#### 3.3.1 CoordValue

The CoordValue class represents either a 1,2 or 3 element tuple with a unit, or a curve in 2 or 3 dimensions. The curve support is currently a place holder and may be supported more fully in future versions. Each curve has a shape, and two end points of the appropriate dimension. The 2 and 3 dimensional tuples carry an optional unit and a time-derivative unit at the tuple level, while the 1D elements of the tuple themselves have a unit (and an STCRef).

- **...Coord.Value** a 1D value
- **...Coord.Value.Unit** The value unit
- **...Coord.Value.UCD** The value UCD (part of STCRef)
- **...Coord.Value.Ref** The reference link (part of STCRef)
- **...Coord.Value.C1** X coord of 2D or 3D value
- **...Coord.Value.C2** Y coord of 2D or 3D value
- **...Coord.Value.C3** Z coord of 3D value
- **...Coord.Value.C\*.Unit** Unit of X,Y,Z coord of 2D or 3D value

- ...**Coord.Value.C\*.TimeUnit** Unit of X,Y,Z coord of 2D or 3D value
- ...**Coord.Value.C\*.STCRef** Reference link for X,Y,Z coord of 2D or 3D value
- ...**Coord.Curve.Shape** Type of curve. Values allowed: "line" (default) or "great circle".
- ...**Coord.Curve.Unit** Unit for points in curve
- ...**Coord.Curve.P1.C1** X coord of point 1 in 2D or 3D curve
- ...**Coord.Curve.P1.C2** Y coord of point 1 in 2D or 3D curve
- ...**Coord.Curve.P1.C3** Z coord of point 1 in 3D curve
- ...**Coord.Curve.P2.C1** X coord of point 2 in 2D or 3D curve
- ...**Coord.Curve.P2.C2** Y coord of point 2 in 2D or 3D curve
- ...**Coord.Curve.P2.C3** Z coord of point 2 in 3D curve



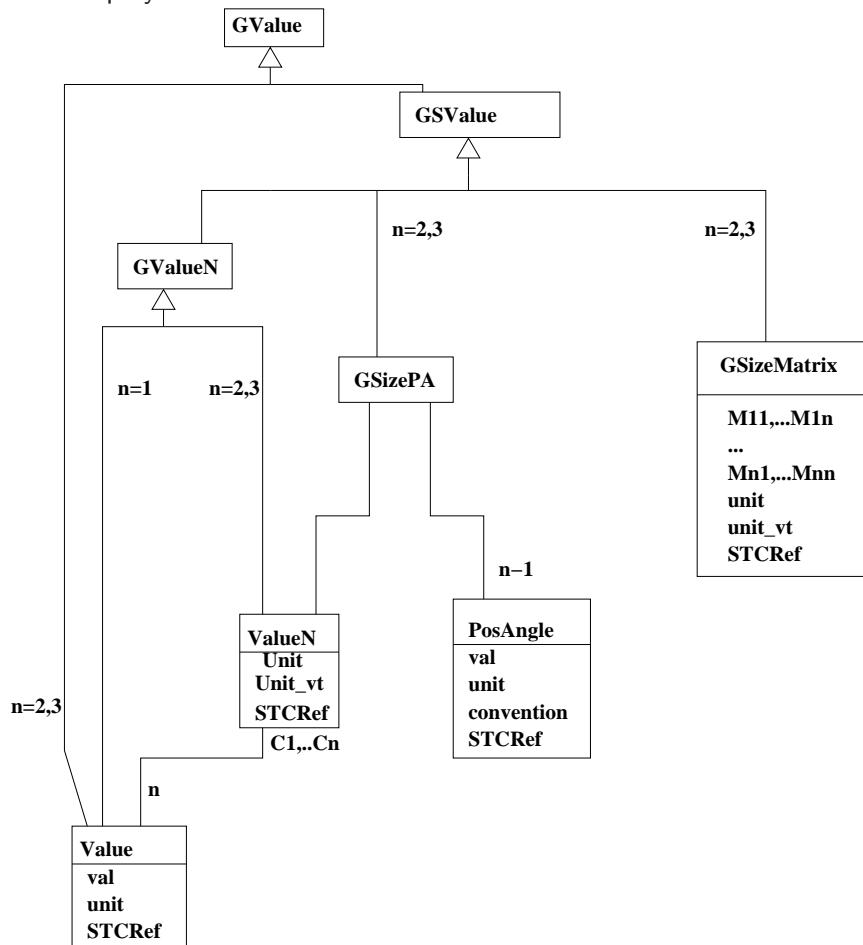
### 3.3.2 Error, Resolution, Size, PixSize

The Error, Resolution and Size objects are of type GValue, which can have dimension n=1, 2 or 3 and can be one of

- Value, a single radius (e.g. an error circle) if  $n=1,2,3$
- ValueN, an  $n$ -tuple indicating an ellipsoid aligned with the axes (the values are the semi-axes) (allowed if  $n > 1$ ).
- GSizePA, an  $n$ -tuple with angles indicating an ellipsoid (e.g. a rotated error ellipse in 2D). (allowed if  $n > 1$ ).
- GSizeMatrix, an  $n \times n$  matrix. For Error, this is an error covariance matrix. For Resolution, it's a matrix representation of the PSF. For Size and PixSize, it represents a coordinate system in which the pixel coordinate system is projected at angles to the coordinate axes (and in which axes may potentially be degenerate). (allowed if  $n > 1$ ).

The PixSize object is of type GValue, which is like a GValue except that it cannot be a Value if  $n > 1$ . The PixSize  $n$ -tuple values are interpreted as being the coordinate size of the sides of the pixel.

The objects beginning with G (GValue, etc.) are introduced at the DM level to generalize and simplify some of the similar elements in the XML.



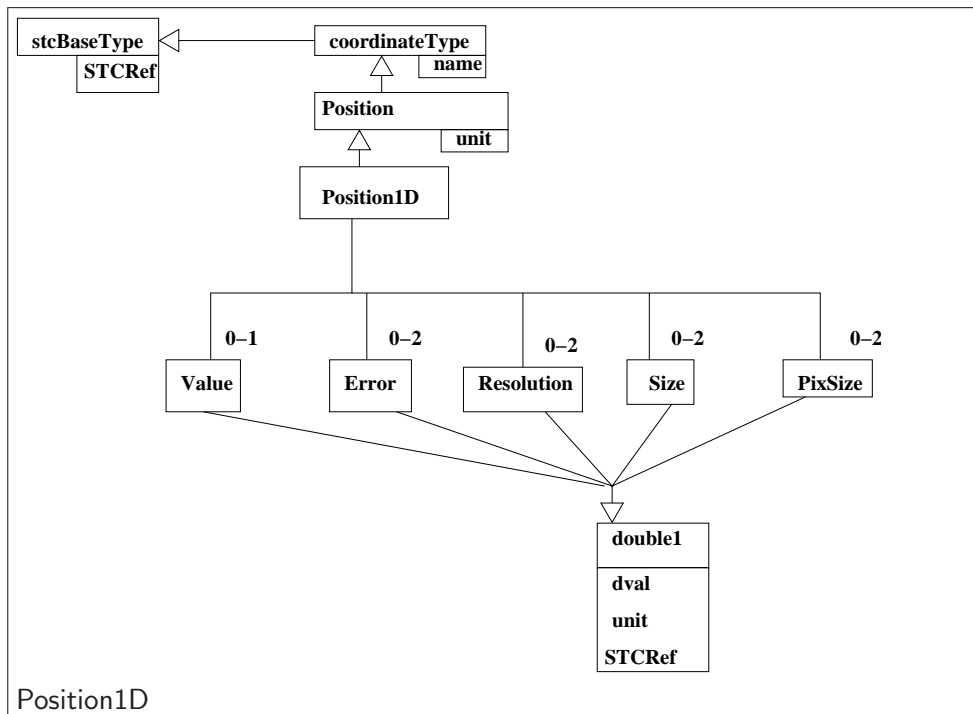
### 3.4 Specialized Coordinates: Position and Velocity

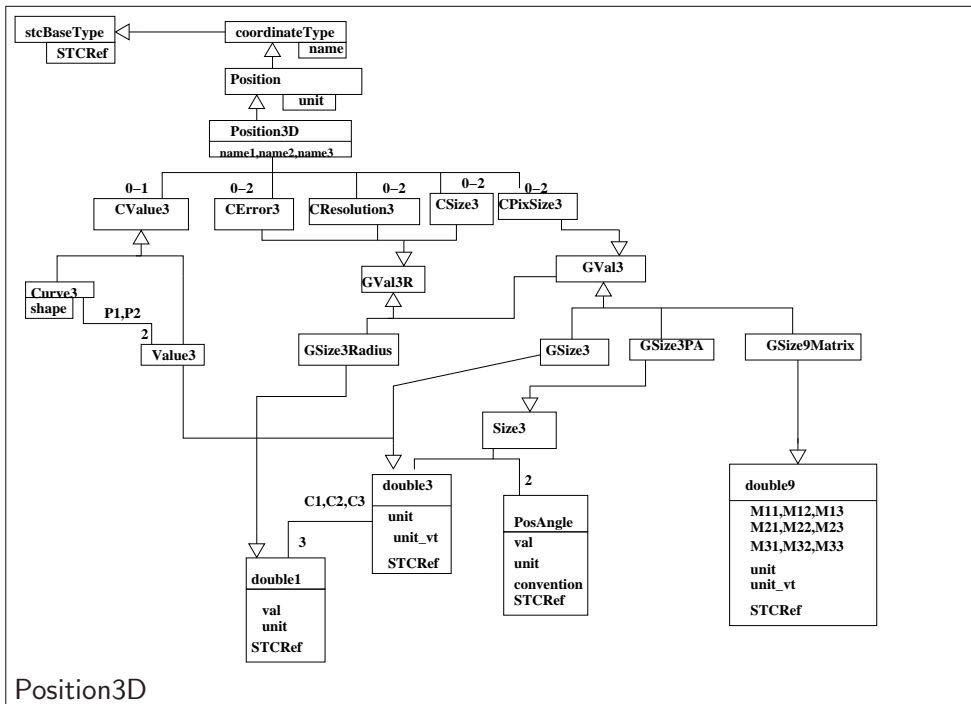
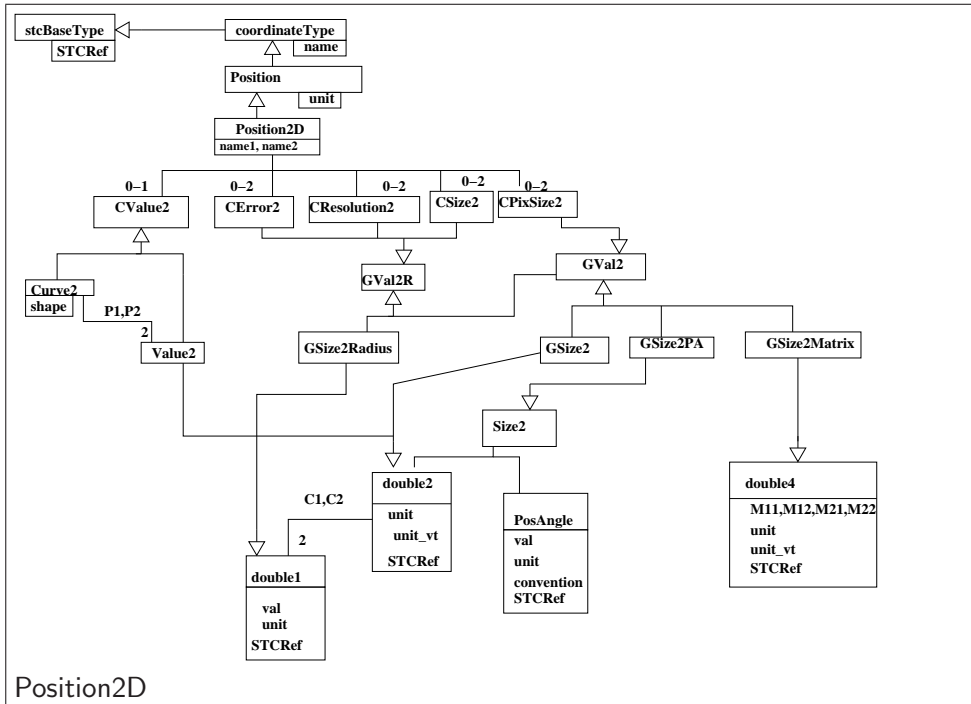
The following figures illustrate the complicated structure of the STC 1,2 and 3 dimensional Position and Velocity elements.

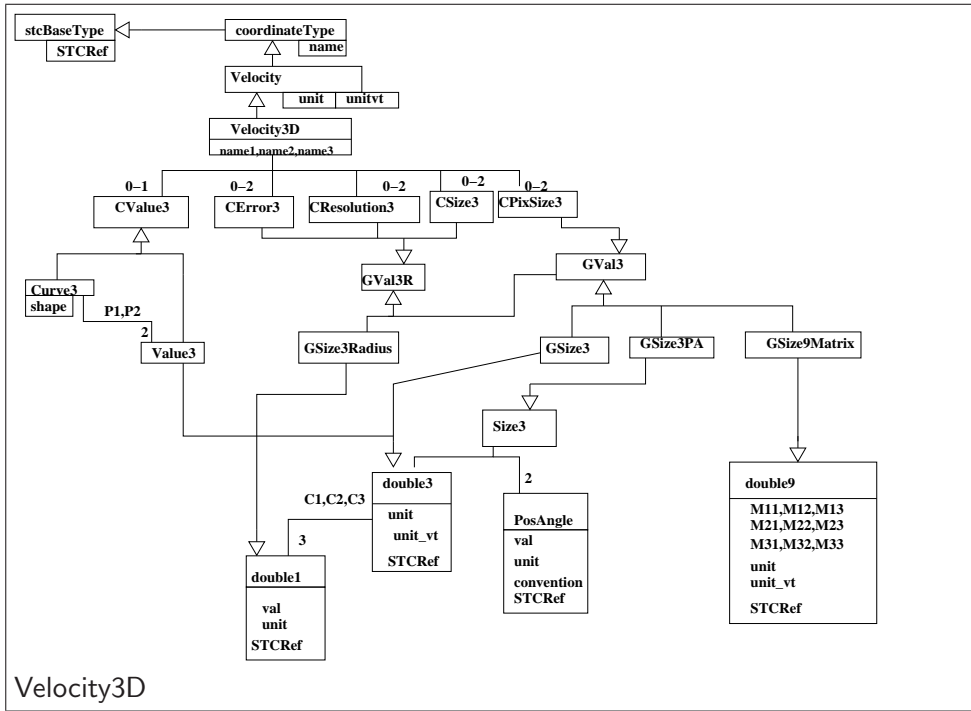
I have followed the XML schema design closely, except that I have replaced the plethora of intermediate types like PixSize3, Error3Matrix etc. with generic intermediate types beginning with G: GVal3R, GSize2 etc. In the figures, these have both upward and downward facing inheritance arrows. For example, the CError3, CResolution3, and CSize3 classes are subclasses of (examples of) both Error, Resolution and Size (not illustrated here) and of GVal3R: CError3 is-a GVal3R. But a GVal3R is (in XML a substitution group) either a GSize3Radius, a GSize3, a GSize3PA, or a GSize3Matrix; we can regard this by saying that GVal3R is an abstract type and the GSize3Radius class is-a GVal3R, - but the GSize3Radius also is-a double1. This complication is likely a failing in my understanding of OO; in actual implementation it's simple. Both CError3 and CResolution3 are a GVal3R, which can be either a double1, a double3, a Size3, or a double9.

The Velocity elements are the same as the Position elements except that they have an extra unit for the time derivative. Note the proliferation of units: in Position3D in the case where CValue3 is a Curve3 containing two Value3s, the overall class has a unit (from the parent Position class), each of the two Value3s is a double3 which can have its own unit, and each double3 contains three double1s which can each have its own unit. The idea is that there is a hierarchy of defaults with the lowest unit in the tree being the one used.

So what is the unit of Position3D.Curve3.P2.C1? It's Position3D.Curve3.P2.C1.unit if present; otherwise you use Position3D.Curve3.P2.unit (the double3) and if that isn't there you use Position3D.unit. The hope is that everything in a Position3D will have the same unit, Position3D.unit, and the lower level units will always be blank. But if you want to be perverse and use a different unit for each component of the value, error, etc., the flexibility is there.

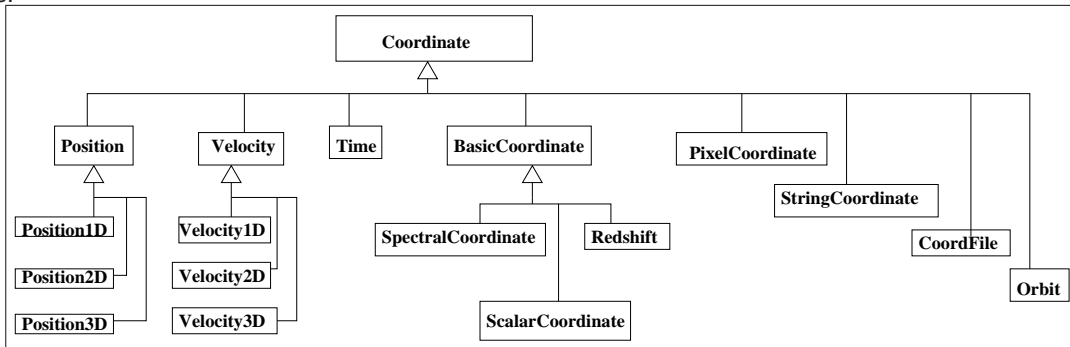






Velocity3D

The STC schema defines the different dimensionalities as different elements, in the following tree:



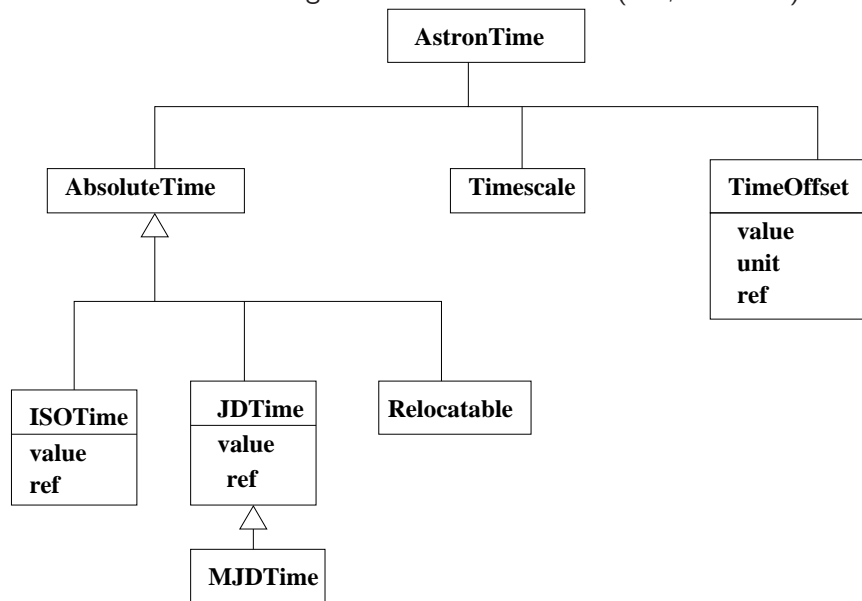
### 3.5 Specialized Coordinates: AstronTime

An AstronTime consists of an AbsoluteTime value, a Timescale, and an optional TimeOffset. The AbsoluteTime value can be used directly to store the time, or as the starting point from which the TimeOffset is measured. For example, in a time resolved observation the AbsoluteTime might be the observation start time in JD days and the TimeOffset might be seconds since the start of the observation.

- **Time.Type** The flavor of time: values are ISO, JD, MJD or RELOCATABLE. The latter implies a relative time used in a simulation; the TimeOffset field will be used to give the simulation time. This Time.Type field is implicit in XML and other object-oriented serializations but may be needed in VOTABLE or FITS serializations.
- **Time.ISOTime.Value** String giving ISO time.
- **Time.JDTime.Value** Decimal value of JD in days

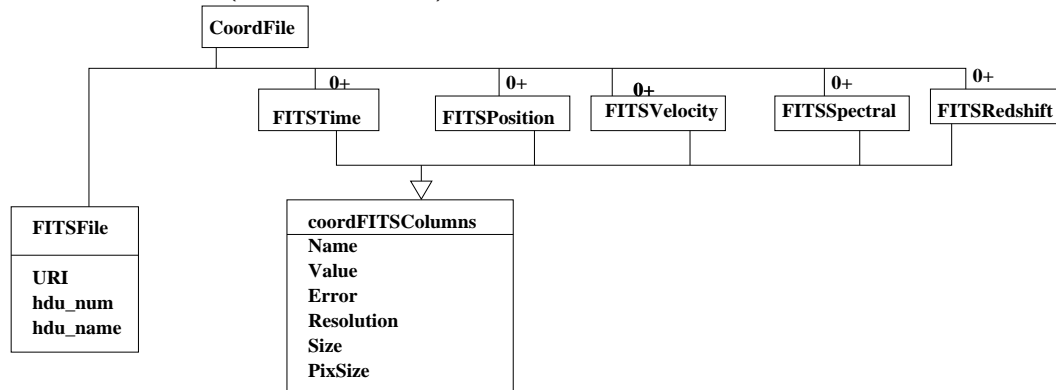
- **Time.MJDTime.Value** Decimal value of MJD in days.
- **Time.Timescale** Timescale - eg. UTC. Optional, defaults to timescale of coordinate system.
- **Time.TimeOffset.Value** Time offset.
- **Time.TimeOffset.Unit** Time offset unit, required if TimeOffset.Value is present
- **Time.TimeOffset.Ref**

The Timescale is as given in the CoordFrame (TT, UTC etc).



### 3.6 Coordinates in FITS files

STC provides a way to associate a FITS file with a coordinate system. The idea is that a single binary table HDU contains a set of columns which contain vectors of coordinate values. Only the explicit standard 'astronomical' coordinates are supported: Time, Position, Velocity, Spectral, Redshift. For each of these you can associate columns for the value, error, resolution, Size and PixSize entries in a coordFITSColumns object, whose fields are strings giving the column names (TTYPEn values). If the entries are multidimensional, the strings will be comma-delimited lists of column names (e.g. "RA,DEC".)

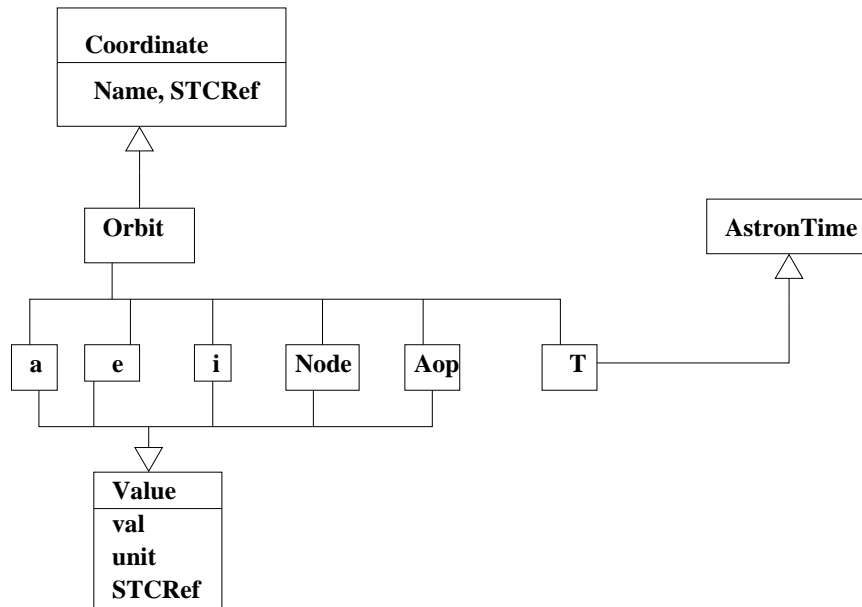


### 3.7 Keplerian Coordinates

To describe objects moving in a gravitational field, one can use 6-dimensional phase space position-velocity coordinates with a Position3D and Velocity3D. But for objects in a field dominated by a single mass and where non-gravitational forces are small, it can be useful to transform to 6-dimensional Keplerian coordinate space where the coordinates are the osculating Keplerian orbital elements. Then for a perfect point mass and no other forces, only the true anomaly coordinate will change and the others will be constant.

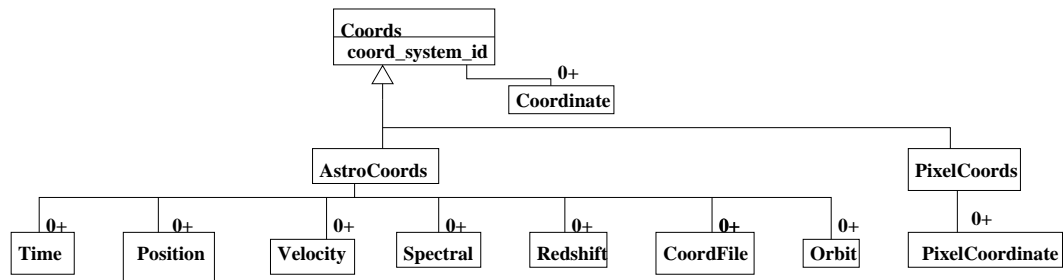
The elements for each coordinate are

- **Orbit.a**, the semi-major axis
- **Orbit.e**, the eccentricity
- **Orbit.i**, the inclination
- **Orbit.Node**, the longitude of ascending node
- **Orbit.Aop**, the argument of periapsis
- **Orbit.T**, the time of periapsis (of type AstronTime).



### 3.8 Coordinate Sets

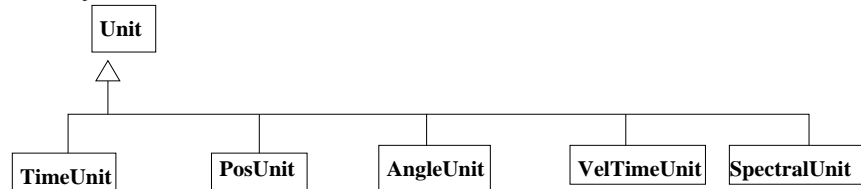
The Coords object is an arbitrary tuple of Coordinate objects together with an coord\_system\_id attribute linking the tuple to a CoordSys. There are two specialized forms of Coords: the AstroCoords which contains generic as well as specialized astronomical axes, and the PixelCoords which contains generic as well as pixel axes. The general Coords contains only generic axes, so AstroCoords is usually the one of interest.



## 4 Auxiliary types

### 4.1 Units

STC defines separate classes of allowed units for position (PosUnit), angle (AngleUnit), spectral coordinate (SpectralUnit) and two kinds of time unit, TimeUnit and VelTimeUnit. The latter is for use with a PosUnit to define a velocity unit; right now TimeUnit and VelTimeUnit are exactly the same.



### 4.2 PositionAngles

STC supports three different conventions for position angles: North (increasing north to east); X (increasing from +X to +Y) and Y (increasing from +Y to +X).

A position angle consists of a number and a unit (default is deg) and a convention (default is X), together with an optional set of reference pointers.

- **...PosAngle.Value**
- **...PosAngle.Unit** - the PosAngle unit, default deg.
- **...PosAngle.Convention** the PosAngle convention, default X.

<b>PosAngle</b>
<b>value</b> <b>unit</b> <b>convention</b>

### 4.3 Intervals

The abstract CoordInterval object has the STCRef metadata (UCD, ID, Ref) and some extra metadata:

- CoordInterval.UCD
- CoordInterval.ID
- CoordInterval.Ref
- **CoordInterval.lo\_include**, true (default) if the interval is closed below.
- **CoordInterval.hi\_include**, true (default) if the interval is closed above.
- **CoordInterval.fill\_factor**, filling factor of interval, optional with default 1.0.

This generic interval has many specialized versions. The following have actual ranges (LoLimit and HiLimit, or equivalents:) TimeInterval, CoordScalarInterval, Coord2VecInterval, Coord3VecInterval, PosScalarInterval, PosNVecInterval, VelScalarInterval, VelNVecInterval, SpectralInterval, RedshiftInterval. Some of these also have a unit field and a frame\_id field.

- **CoordInterval.LoLimit**, a Value giving the start of the interval. It may be a 1-D Value (with unit, UCD, etc.) or a 2D or 3D ValueN.
- **CoordInterval.HiLimit**, a Value giving the stop of the interval; as for LoLimit.
- **CoordInterval.Unit**, giving a common unit for both LoLimit and HiLimit.
- **CoordInterval.TimeUnit**, giving the time derivative unit for cases with a velocity.
- **CoordInterval.frame\_id**, a string giving the ID of the CoordFrame for that axis.
- **TimeInterval.StartTime**, an AstronTime giving the start of the interval. Equivalent to LoLimit.
- **TimeInterval.StopTime**, an AstronTime giving the stop of the interval. Equivalent to HiLimit.

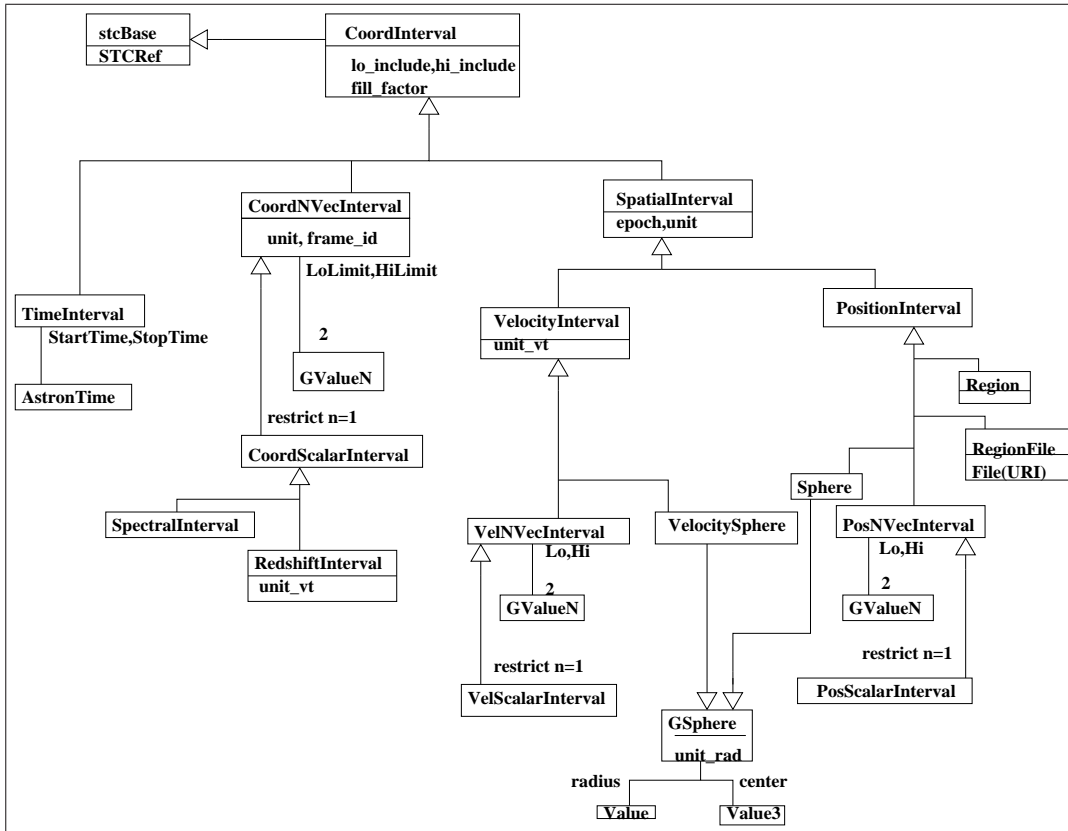
The VelocitySphere interval is a 3D region defined by a center location and a radius value: it includes the following data items inherited from Value and ValueN:

- **VelocitySphere.radius**
- **VelocitySphere.radius.unit**
- **VelocitySphere.center.C1**
- **VelocitySphere.center.C2**
- **VelocitySphere.center.C3**
- **VelocitySphere.center.unit**

The Sphere interval is identical (but does not have the CoordInterval.TimeUnit field since it is for a position interval, not a velocity interval).

The PositionInterval classes include the Region class, which is described below. They also include the RegionFile class which points to a file containing region information; the format of this file is not specified by STC.

- **RegionFile.File** Filename of region file.

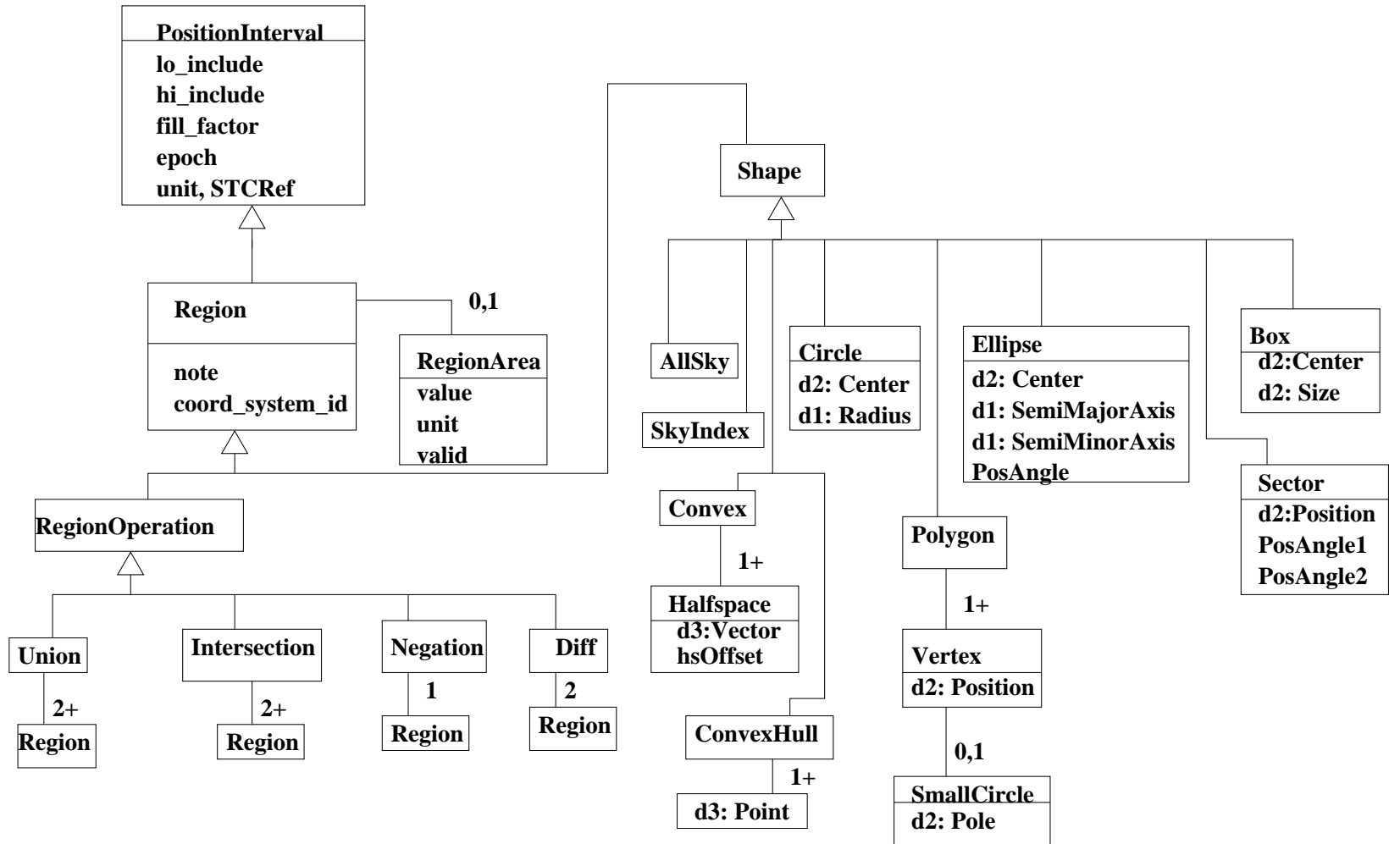


#### 4.4 Regions

The Region is a subclass of PositionInterval. It represents a subset of the 2D sphere or 2D cartesian plane. The simplest kind of Region is a Shape (e.g. Circle, AllSky...). Regions can be combined with the operators Union, Intersection, etc. to form a new (compound) Region.

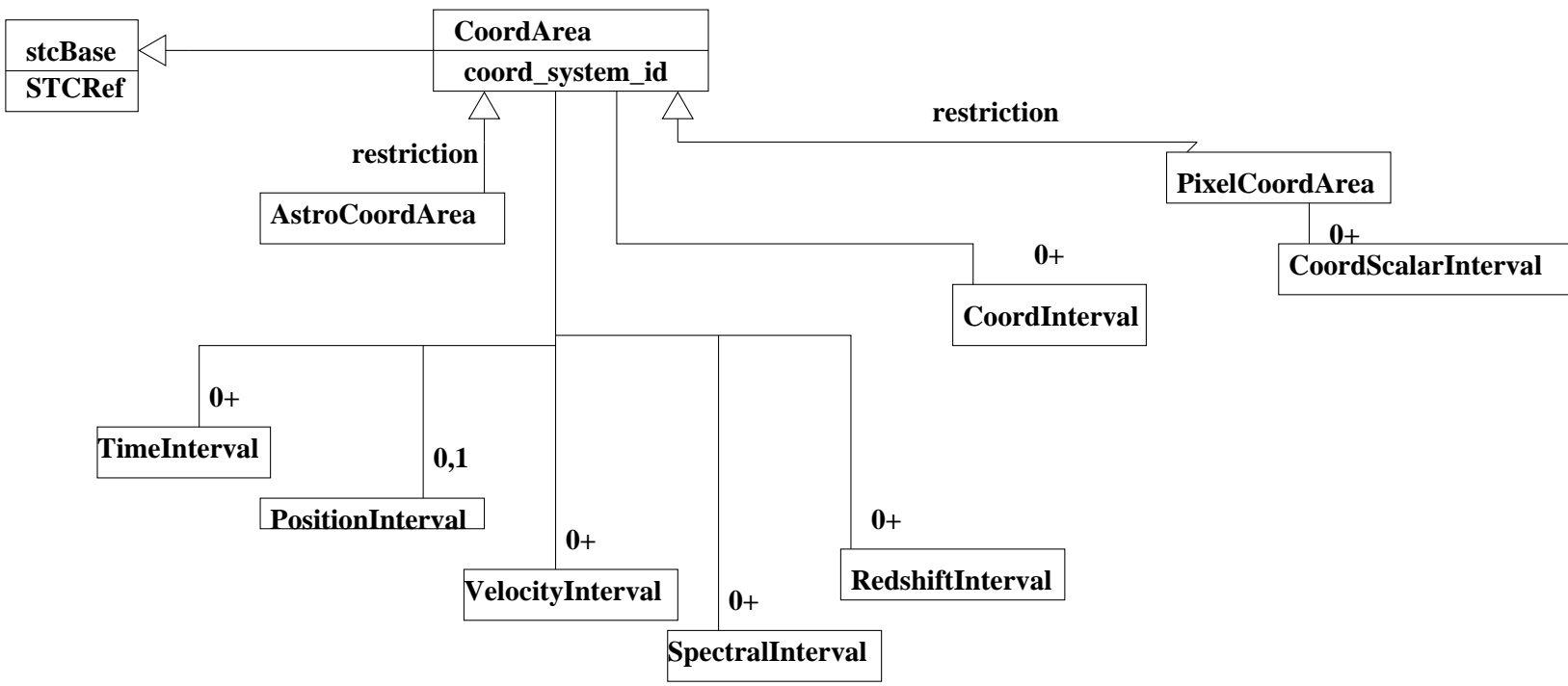
The AllSky region implies the whole sphere.

The diagram has been compressed so that the parameters of each shape appears as a class attribute instead of an associated class; the d1,d2 datatypes are shorthand for what I have earlier called Value and Value2 (i.e. ValueN with n=2).



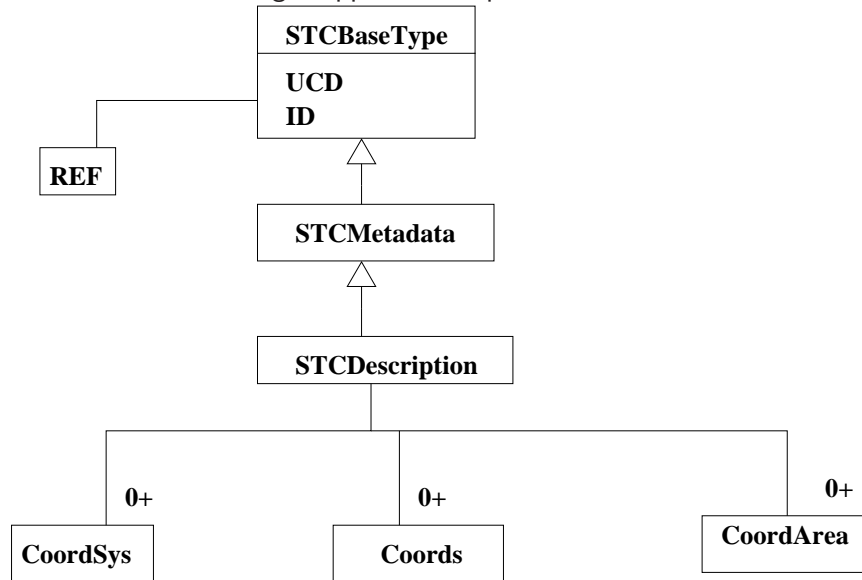
## 4.5 Areas

The `CoordArea` object can contain `TimeIntervals`, a `PositionInterval`, `VelocityIntervals`, `SpectralIntervals`, `RedshiftIntervals` and an arbitrary number of generic `CoordIntervals`. The specialized `AstroCoordArea` is the same except that the generic `CoordIntervals` are restricted to be `CoordScalarIntervals`.



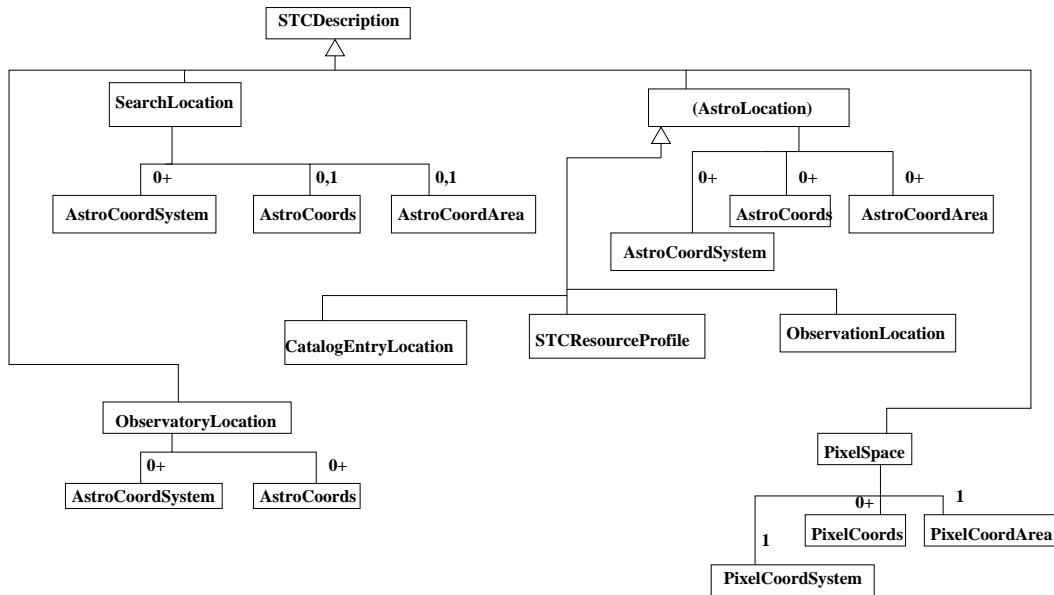
## 5 High level STC objects

The STC Description collects together CoordSys, Coords and CoordArea objects. A suggested use is to have one CoordSys, one Coords and one CoordArea representing the characterization 'location' and 'coverage support' concepts.



Some STCDescription subtypes include

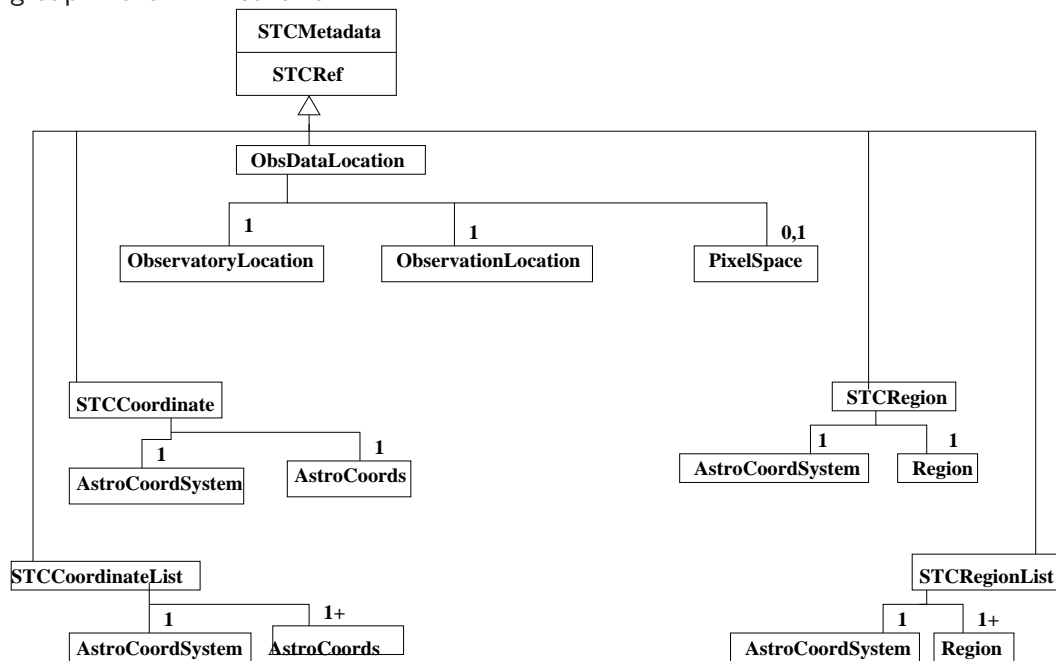
- SearchLocation: AstroCoordSystems, with 0 or 1 AstroCoords and AstroCoordsArea.
- AstroLocation (a generic type not in the XML): Like STCDescription but only AstroCoordSystems, AstroCoords and AstroCoordArea instead of their generic counterparts. Arbitrary numbers of each.
- CatalogEntryLocation, STCResourceProfile and ObservationLocation are examples of AstroLocation but with different semantics. The ObservationLocation is used to give the field of view of an observation.
- ObservatoryLocation, consisting of AstroCoordSystems and AstroCoords but no CoordArea. Used to give the location of your observatory on the Earth or in space.
- PixelSpace, with PixelCoordSystem PixelCoords and PixelCoordArea, specializing the STCDescription to a pixel-space example.



More high level objects:

- ObsDataLocation, which aggregates three STCDescription objects: an ObservatoryLocation, an ObservationLocation and an optional PixelSpace.
- The STCCoordinate aggregates one AstroCoordSystem and one AstroCoords
- The STCCoordinateList aggregates one AstroCoordSystem and multiple AstroCoords
- The STCRegion aggregates one AstroCoordSystem and one Region
- The STCRegionList aggregates one AstroCoordSystem and multiple Regions

STCResourceProfile and ObsDataLocation are also part of the STCMetadata substitution group in the XML schema.



## 6 Frequently Asked Questions

### 6.1 Why do you need a SpectralFrame in STC?

STC is intended for recording the coordinates of photons observed by astronomers, either as observed or transformed to the coordinates it would have had if observed at a different location. The spectral coordinate of a photon depends on the space and time frame in which it is observed:

- If observed from a moving telescope (e.g. on a satellite), the spectrum is Doppler-shifted.
- If corrected to the barycenter (e.g. for pulsar timing), the spectrum should strictly be corrected to the velocity frame of the barycenter.
- We may want to represent the spectrum in the rest frame of the source.

However, the space-time coordinate system does not depend on the spectral frame; in this sense the space and time axes are more fundamental than the spectral and redshift frames. Obviously, a dataset which represents the ephemeris of an asteroid, rather than an observation of it, does not have a spectral component and does not require a spectral frame.

### 6.2 Why do you need separate SpectralFrame and RedshiftFrame in STC?

The SpectralFrame represents an axis of the data labelled in a true spectral coordinate such as wavelength or frequency. The RedshiftFrame represents an axis of the data labelled in velocity or redshift. For direct observational spectral data, such an axis makes an assumption about identification of spectral lines and so is model-laden. In many cases we have a single spectrum of small bandwidth containing a single prominent line whose profile is being studied, and the spectral axis could be labelled either with a SpectralFrame or with a RedshiftFrame. When many lines are present, you could represent them as part of one (possibly piecewise) spectrum with a SpectralFrame, or you could represent them as an array of velocity profiles, with a SpectralFrame axis distinguishing the array elements - the individual lines - and a RedshiftFrame axis labelling the velocity profiles of each line. So, it is possible to have both a spectral and a redshift axis in one dataset.

### 6.3 Barycenter corrections: why might you need a TimeRefDirection?

The time coordinate in an STC instance is the time that a photon (or other phenomenon) was - or WOULD HAVE BEEN - observed at the location `CoordFrame.ReferencePosition`. If you change the `CoordFrame.ReferencePosition` from `TOPOCENTER` to some other location, e.g. `BARYCENTER`, you need to know the celestial position of the source and the location of the `TOPOCENTER` to figure out when the same photon wavefront would have reached the `BARYCENTER`. Now if you do a further transformation to go to say `MARS` center, you need to know what position was *assumed* for the previous transformation. In some data analysis systems the barycenter transformation is performed assuming a single position for all objects in the field. You need to have recorded that assumed position somewhere to perform later transformations; STC here provides a place to so record.

Hence, `TimeRefDirection` is needed in principle if (1) the `ReferencePosition` is not `TOPOCENTER` and (2) time corrections have been applied which did not use the source position given in the STC spatial coordinate instance.