Start here

STCmetadata (substGrp)
Toplevel: STCmetadata contains a choice of: ResourceProfile, SearchLocation, CatalogEntryLocation, or ObservatoryLocation plus ObservatoryLocation elements

ResourceProfile — Describes the spatial and temporal coverage of a resource
- **CoordArea** (type: stc:coordAreaType) — The coverage area of the resource; the fill factor does not need to be 1.0
- **CoordSpec** (type: crd:coordsType) — CoordSpec contains information on time and spatial resolution, errors, and pixelsizes (if fixed); typical best numbers are expected
- **CoordSystem** (type: stc:coordSystemType) 1..∞ — The coordinate system definition: spatial coordinate frame and reference position; time frame and reference position; the coordinate flavor; and the planetary ephemeris; an ID is required, since this is how coordinate elements are associated with their coordinate systems

SearchLocation — Defines the spatial and temporal coordinate space specified by a query
- **CoordArea** (type: stc:coordAreaType) — Region of Interest
- **CoordSpec** (type: crd:coordsType) — CoordSpec contains information on desired time and spatial resolution, errors, and pixelsizes (if fixed); typical best numbers are expected
- **CoordSystem** (type: stc:coordSystemType) 1..∞ — The coordinate system definition: spatial coordinate frame and reference position; time frame and reference position; the coordinate flavor; and the planetary ephemeris; an ID is required, since this is how coordinate elements are associated with their coordinate systems

CatalogEntryLocation — Describes the spatial and temporal coverage of a catalog (fragment)
- **Coords** (type: crd:coordsType) 1..∞ — The coordinate element consists either of a coordinate file or a sequence of time element, spatial coordinate element, velocity element, and redshift element; the spatial and velocity elements may be scalar or vector; it needs to refer to a coordinate system
- **CoordArea** (type: stc:coordAreaType) 0..∞ — Optional specification of the area covered by the catalog data
- **CoordSystem** (type: stc:coordSystemType) 1..∞ — The coordinate system definition: spatial coordinate frame and reference position; time frame and reference position; the coordinate flavor; and the planetary ephemeris; an ID is required, since this is how coordinate elements are associated with their coordinate systems

ObservatoryLocation — Specifies the location of the observatory during the observation
- **Coords** (type: crd:coordsType) — The coordinate element consists either of a coordinate file or a sequence of time element, spatial coordinate element, velocity element, and redshift element; the spatial and velocity elements may be scalar or vector; it needs to refer to a coordinate system
- **CoordArea** (type: stc:coordAreaType) — The field of view
- **CoordSystem** (type: stc:coordSystemType) 1..∞ — The coordinate system definition: spatial coordinate frame and reference position; time frame and reference position; the coordinate flavor; and the planetary ephemeris; an ID is required, since this is how coordinate elements are associated with their coordinate systems

ObservationLocation — Describes the spatial and temporal coverage of the observation

Describes the spatial and temporal volume covered by an observation

coordSystemType — The coordinate system definition: spatial coordinate frame and reference position; time frame and reference position; the coordinate flavor; and the planetary ephemeris; an ID is required, since this is how coordinate elements are associated with their coordinate systems

TimeFrame (type: stc:timeFrameType) — The time coordinate reference frame
- **TimeRefPosition** (type: stc:refPosType) — Origin of the coordinate reference frame: either a "known place" such as geocenter or barycenter, or a position defined in a known coordinate system
- **TimeRefDirection** (type: crd:coordsType) — Some time transformations (e.g., change of RefPos) depend on an assumed directional position of the source

stc:refPosType: **RefPos** (type: xs:string, default: TELESCOPE, enum: BARYCENTER HELIOCENTER ...); **CoordOrigin** (type: crd:coordsType)

SpaceFrame (type: stc:spaceFrameType) — The spatial coordinate reference frame

stc:spaceFrameType
- **CoordFrame** (type: stc:coordFrameType) — Coordinate reference frame: optional equinox with either a standard reference system (ICRS, FK5, FK4) and optional standard pole (equatorial, ecliptic, galactic, etc.), or pole (positive Z-axis) and positive X-axis direction
  - **CoordEquinox** (type: stc:coordEquinoxType) — Coordinate equinox: B(year) or J(year) with at least one decimal; do not use in conjunction with ICRS
  - **RefSystem** (type: xs:string, default: ICRS, enum: ICRS FK5 FK4 ...)
  - **StandardPole** (type: xs:string, default: EQUATORIAL, enum: EQUATORIAL ECLIPTIC GALA...) — Most common: standard reference system with optional pole direction (default: equatorial)
  - **Pole_Zaxis** (type: crd:coordsType)
  - **Xaxis** (type: crd:coordsType) — Define coordinate reference frame from scratch; pole and X-axis need to be defined in a known coordinate system
- **CoordRefPosition** (type: stc:refPosType) — Origin of the coordinate reference frame: either a "known place" such as geocenter or barycenter, or a position defined in a known coordinate system

stc:refPosType: **RefPos** (type: xs:string, default: TELESCOPE, enum: BARYCENTER HELIOCENTER ...); **CoordOrigin** (type: crd:coordsType)

SpectralFrame (type: stc:dopplerReferenceType, default, enum: OBSERVATORY GEOCENTER...) — The reference frame for the spectral coordinate; note presence of LSR

stc:dopplerType: **DopplerDefinition** (type: stc:dopplerDefinitionType, default, enum: OPTICAL RADIO RELATIVISTIC) — The Doppler definition used: optical, radio, or pseudo-relativistic (i.e., how is a redshift converted to a velocity); the most common is optical, except when the reference is LSR (usually radio); **DopplerReference** (type: stc:dopplerReferenceType, default, enum: OBSERVATORY GEOCENTER...) — The reference frame for the Doppler velocities; note presence of LSR

DopplerFrame (type: stc:dopplerType) — Contains the Doppler definitions, including whether the values are velocity or redshift (value_type)

CoordFlavor (type: stc:coordFlavorType) — Provides the coordinate definitions: number of axes, SPHERICAL or CARTESIAN, presence of velocities, and the Doppler definitions (if needed)

PlanetaryEphem (type: stc:planetaryEphemType, default, enum: JPL-DE200 JPL-DE405) — If solar system positions are implied anywhere, the planetary ephemeris to be used needs to be provided - usually JPL-DE405 with ICRS and JPL-DE200 with FK5

coordsType — The coordinate element consists either of a coordinate file or a sequence of time element, spatial coordinate element, velocity element, and redshift element; the spatial and velocity elements may be scalar or vector; it needs to refer to a coordinate system
- **Time** (type: crd:coordTimeType) — This contains all the elements of a time coordinate: value, error, resolution, size, pixel size, unit, and name
- **PosScalar** (type: crd:posScalarType)
- **Pos2Vector** (type: crd:pos2VectorType)
- **Pos3Vector** (type: crd:pos3VectorType)
- **VelScalar** (type: crd:velScalarType)
- **Vel2Vector** (type: crd:vel2VectorType)
- **Vel3Vector** (type: crd:vel3VectorType)
- **Redshift** (type: crd:velScalarType) — This contains all the elements of the redshift coordinate: value, error, resolution, size, pixel size, unit, and name
- **Spectrum** (type: crd:coordSpectralType) — This contains all the elements of the spectral coordinate: value, error, resolution, size, pixel size, unit, and name
- **CoordFile** (type: xs:anyURI, default, enum)

coordTimeType — This contains all the elements of a time coordinate: value, error, resolution, size, pixel size, unit, and name

CoordValue (type: crd:astronTimeType) — astronTime may be expressed in ISO8601 or as a double relative to a reference time; or it may be provided in an entity refered to by the Reference element

crd:astronTimeType
- **ISOTime** (type: xs:dateTime) — Only a limited subset of ISO 8601 is allowed: yyyy-mm-ddThh:mm:ss.sss...
- **JDTime** (type: xs:decimal) — Note that this time may require precision that exceeds that of a double
- **MJDTime** (type: xs:decimal) — Note that this time may require precision that exceeds that of a double
- **RelativeTime** (type: xs:double) — Relative time needs a time reference (a zero point) which should not be in relative time
- **ISORefTime** (type: xs:dateTime) — The reference time needs to be ISO8601, JD, or MJD; note: only a limited subset of ISO 8601 is allowed: yyyy-mm-ddThh:mm:ss.sss...
- **JDRefTime** (type: xs:decimal) — Note that this time may require precision that exceeds that of a double
- **MJDRefTime** (type: xs:decimal) — Note that this time may require precision that exceeds that of a double
- **Reference** (type: xs:IDREF)
- **ISORefTime** (type: xs:dateTime) — The reference time needs to be ISO8601, JD, or MJD; only needed if time_base="relative"
- **JDRefTime** (type: xs:decimal) — Note that this time may require precision that exceeds that of a double
- **MJDRefTime** (type: xs:decimal) — Note that this time may require precision that exceeds that of a double
- **TimeScale** (type: crd:timeScaleType, default, enum: TT TDT ET TDB TCG TCB TAI I...)

CoordError (type: crd:coordTimeValueType) — A scalar coordinate value consists of a double or a reference to an entity (that should contain a double)

crd:coordTimeValueType: **Value** (type: xs:double, default, enum); **Reference** (type: xs:IDREF, default, enum)

CoordResolution (type: crd:coordTimeValueType) — A scalar coordinate value consists of a double or a reference to an entity (that should contain a double)

CoordSize (type: crd:coordTimeValueType) — A scalar coordinate value consists of a double or a reference to an entity (that should contain a double)

CoordPixsize (type: crd:coordTimeValueType) — A scalar coordinate value consists of a double or a reference to an entity (that should contain a double)

Name (type: xs:string, default, enum)

pos2VectorType — This contains all the elements of a coordinate vector of length 2: value, error, resolution, size, pixelsize, unit, and names

crd:posCoordType: **Name** (type: xs:string, default, enum)

CoordValue (type: crd:coord2ValueType) — A coordinate vector of length 2 consists of either two doubles or two references (that should each contain one double)

crd:coord2ValueType: **Value** (type: crd:doubleArrayType, default, enum) — For vector coordinates we need arrays of doubles; **Value60** (type: xs:string, default, enum); **Reference** (type: xs:IDREFS, default, enum)

CoordError (type: crd:coord2SizeType) — A size (ellipse) in two coordinates contains: two doubles or two IDREFs, each with optional position angle; or a 2x2 matrix defining the ellipse

crd:coord2SizeType: **Value** (type: crd:doubleArrayType, default, enum) — For vector coordinates we need arrays of doubles; **PosAngle** (type: xs:double, default, enum); **Reference** (type: xs:IDREFS, default, enum); **PosAngleRef** (type: xs:IDREF, default, enum); **Matrix** (type: crd:doubleArrayType, default, enum) — For vector coordinates we need arrays of doubles

CoordResolution (type: crd:coord2SizeType) — A size (ellipse) in two coordinates contains: two doubles or two IDREFs, each with optional position angle; or a 2x2 matrix defining the ellipse

CoordSize (type: crd:coord2SizeType) — A size (ellipse) in two coordinates contains: two doubles or two IDREFs, each with optional position angle; or a 2x2 matrix defining the ellipse

CoordPixsize (type: crd:coord2SizeType) — A size (ellipse) in two coordinates contains: two doubles or two IDREFs, each with optional position angle; or a 2x2 matrix defining the ellipse

posScalarType — This contains all the elements of a scalar coordinate: value, error, resolution, size, pixel size, unit, and name

crd:posCoordType: **Name** (type: xs:string, default, enum)

CoordValue (type: crd:coordValueType) — A scalar coordinate value consists of a double or a reference to an entity (that should contain a double)

crd:coordValueType: **Value** (type: xs:double, default, enum); **Value60** (type: xs:string, default, enum); **Reference** (type: xs:IDREF, default, enum)

CoordError (type: crd:coordValueType) — A scalar coordinate value consists of a double or a reference to an entity (that should contain a double)

CoordResolution (type: crd:coordValueType) — A scalar coordinate value consists of a double or a reference to an entity (that should contain a double)

CoordSize (type: crd:coordValueType) — A scalar coordinate value consists of a double or a reference to an entity (that should contain a double)

CoordPixsize (type: crd:coordValueType) — A scalar coordinate value consists of a double or a reference to an entity (that should contain a double)

coordSpectralType — This contains all the elements of a spectral coordinate: value, error, resolution, size, pixel size, unit, and name

Name (type: xs:string, default, enum)

CoordValue (type: crd:coordSpectralValueType) — astronTime may be expressed in ISO8601 or as a double relative to a reference time; or it may be provided in an entity refered to by the Reference element

crd:coordSpectralValueType: **Value** (type: xs:double, default, enum); **Reference** (type: xs:IDREF, default, enum)

CoordError (type: crd:coordSpectralValueType) — A scalar coordinate value consists of a double or a reference to an entity (that should contain a double)

CoordResolution (type: crd:coordSpectralValueType) — A scalar coordinate value consists of a double or a reference to an entity (that should contain a double)

CoordSize (type: crd:coordSpectralValueType) — A scalar coordinate value consists of a double or a reference to an entity (that should contain a double)

CoordPixsize (type: crd:coordSpectralValueType) — A scalar coordinate value consists of a double or a reference to an entity (that should contain a double)

Region (substGrp: reg:Region)
- **Circle** (type: reg:circleType, substGrp: reg:Region) — The circle is defined by a center and a radius
  - reg:circleType: **Center** (type: crd:coordsType) — The coordinates of the circle's center; **Radius** (type: xs:double, default, enum) — The radius of the circle
- **Convex** (type: reg:convexType, substGrp: reg:Region) — A convex is defined as the intersection of one or more half-plane constraints with the unit sphere; this may also be described as the union of one or more convex polygons bounded by one or more constraint planes
  - reg:convexType: **Constraint** (type: reg:constraintType) 1..∞
    - reg:constraintType: **Vector** (type: crd:coordsType) — This needs to be a spherical coordinate vector; it is the unit vector that is normal to the plane that forms a constraint for a convex; **Offset** (type: xs:double, default, enum) — The distance along the normal vector where the constraint plane intersects that vector; if positive, the spherical sector on the far side (seen from the center) is selected; if negative, the point of intersection is in the opposite direction of the vector, resulting in more than a hemisphere
- **ConvexHull** (type: reg:convexHullType, substGrp: reg:Region) — A convex hull is the convex polygon that contains all of the specified positions; the points have to be constrained to lie in one hemisphere
  - reg:convexHullType: **Point** (type: crd:coordsType) 1..∞
- **Ellipse** (type: reg:ellipseType, substGrp: reg:Region) — Ellipse is an extension of the circle shape, with a minor axis radius and position angle
  - reg:ellipseType: **Center** (type: crd:coordsType) — The coordinates of the circle's center; **Radius** (type: xs:double, default, enum) — The radius of the circle; **MinorRadius** (type: xs:double, default, enum) — Half the minor axis of the ellipse, in radius_unit; **PosAngle** (type: xs:double, default, enum) — Position angle of major axis (Radius), counter-clockwise from north in pos_angle_unit
- **Intersection** (type: reg:intersectionType, substGrp: reg:Region)
  - reg:intersectionType: **Region** (type: reg:regionType) 2..∞
- **Negation** (type: reg:negationType, substGrp: reg:Region)
  - reg:negationType: **Region** (type: reg:regionType)
- **Polygon** (type: reg:polygonType, substGrp: reg:Region) — A polygon is a shape delineated by a list of vertices connected with lines, great circles, or small circles; the inside of the polygon is circumscribed counter-clockwise by the list of vertices; the polygon may be concave but not self-intersecting; the last vertex in the list is the predecessor of the first. For celestial coordinates it is assumed that we are looking at the sphere from the inside, with Right Ascension increasing to the left.
  - reg:polygonType: **Vertex** (type: reg:vertexType) 1..∞ — In order to form polygons, vertices are to be connected with straight line segments. In the case of spherical coordinates: greatcircle segments; if a smallCircle element is present, the vertex and its predecessor are to be connected with a smallcircle, by default in the CoordSys that is referenced; optionally, a pole may be specified (other than the CoordSys pole) that defines the smallcircle system
    - reg:vertexType: **Position** (type: crd:coordsType); **SmallCircle** (type: reg:smallCircleType)
      - reg:smallCircleType: **Pole** (type: crd:coordsType)
- **Sector** (type: reg:sectorType, substGrp: reg:Region) — A sector selects the area between two half great circles or half lines meeting in a specified point
  - reg:sectorType: **Position** (type: crd:coordsType) — The vertex position of the sector; **PosAngle1** (type: xs:double, default, enum) — The area ccw from this position angle is included; **PosAngle2** (type: xs:double, default, enum) — The area cw from this position angle is included
- **Union** (type: reg:unionType, substGrp: reg:Region)
  - reg:unionType: **Region** (type: reg:regionType) 2..∞

RegionFile (type: xs:anyURI, default, enum)

Region (type: reg:regionType)

coordAreaType — A coordinate area consists of a time interval, a spatial area, a velocity interval, and a redshift interval (as needed); the spatial area may be specified by an interval, a circle or sphere, a region element, or a region file

TimeInterval (type: stc:timeIntervalType) 1..∞
- stc:timeIntervalType: **StartTime** (type: crd:astronTimeType) — astronTime may be expressed in ISO8601 or as a double relative to a reference time; or it may be provided in an entity refered to by the Reference element; **StopTime** (type: crd:astronTimeType) — astronTime may be expressed in ISO8601 or as a double relative to a reference time; or it may be provided in an entity refered to by the Reference element

CoordInterval (type: stc:coordIntervalType) 1..∞
- stc:coordIntervalType: **LoLimitScalar** (type: crd:coordValueType) — A scalar coordinate value consists of a double or a reference to an entity (that should contain a double); **HiLimitScalar** (type: crd:coordValueType) — A scalar coordinate value consists of a double or a reference to an entity (that should contain a double); **LoLimit2Vec** (type: crd:coord2ValueType); **HiLimit2Vec** (type: crd:coord2ValueType); **LoLimit3Vec** (type: crd:coord3ValueType); **HiLimit3Vec** (type: crd:coord3ValueType)

CircleOrSphere (type: stc:circleOrSphereType)
- stc:circleOrSphereType: **Radius** (type: xs:double, default, enum); **Center** (type: crd:coordsType)

VelocityInterval (type: stc:velIntervalType) 0..∞
- stc:velIntervalType: **LoLimitScalar** (type: crd:coordValueType) — A scalar coordinate value consists of a double or a reference to an entity (that should contain a double); **HiLimitScalar** (type: crd:coordValueType) — A scalar coordinate value consists of a double or a reference to an entity (that should contain a double); **LoLimit2Vec** (type: crd:coord2ValueType); **HiLimit2Vec** (type: crd:coord2ValueType); **LoLimit3Vec** (type: crd:coord3ValueType); **HiLimit3Vec** (type: crd:coord3ValueType)

SpectralInterval (type: stc:coordSpectralValueType) 0..∞
- **LoLimit** (type: crd:coordSpectralValueType)
- **HiLimit** (type: crd:coordSpectralValueType)

RedshiftInterval (type: stc:coordValueType) 0..∞
- **LoLimit** (type: crd:coordValueType)
- **HiLimit** (type: crd:coordValueType)

For more information:

arots@head-cfa.harvard.edu

http://hea-www.harvard.edu/~arots/nvometa/STCdoc.pdf

http://hea-www.harvard.edu/~arots/nvometa/SpaceTime.html