APT Developer Primer

The following are some concepts and constructs used extensively in APT development that are not being reviewed, but should be understood in the context of the code review.

Java Constructs

Enums

Enums are a Java construct that allows explicitly named enumerated instances of a particular concept. They are often used in place of static int collections often seen in other programming languages. A Java Enum is a light-weight class, so it is fast to use, but also provides all the functionality of a class.

For more on Enums, there is a short description available here:

http://java.sun.com/j2se/1.5.0/docs/guide/language/enums.html

Generics

Generics are a means of identifying the underlying classes contained by another class, and ensuring type safety at compile time. It is an alternative to explicit class casting done at run time, which is necessary to have access to class methods of the underlying object. In the case of Collections, this is a more object-oriented approach to providing the same functionality of a typed array.

With generics you *know* the class of the object you are working with at compile time. With casting, you *hope* you know the class of the object you are working with at run time.

For more on Generics, there is a short description available here:

http://java.sun.com/j2se/1.5.0/docs/guide/language/generics.html

APT Team Libraries

CoSI

The CoSI (Constraint Satisfaction Interface) is a library of objects, and an infrastructure to support automatic rerunning of methods when dependent variables change. The basic design is that there are Properties, Constraints, and a Propagator.

The Propagator runs the Constraints. It has a queue of ones that are ready to run, and runs them sequentially. A Constraint is the equivalent of a method. It accesses some Properties, and sets others, all within the context of a single run() method. While the Constraint is running, all the Properties that it accesses track that this Constraint is dependent on that Property. When the Property changes, it tells the Propagator to add all dependent Constraints to its queue. This is a different way of handling property change events than the one normally supported by Java, but it allows a more declarative expression of the logic. This design was based on that used by Trans.

Tina

Tina (Tina is no acronym) is a library of objects designed to support the proposal hierarchy. It is the basis for the HST and JWST document models. It also provides the GUI elements that are used for editing proposals.

The document model is divided into Elements and Fields. A TinaDocumentElement is a complex type that can be inserted in a document hierarchy in a parent child relationship with the document as a whole. TinaFields are more akin to attributes of a TinaDocumentElement. They represent a specific value and present a specific way of editing that one value. Tina provides all the infrastructure for managing the relationship between the Elements.

For the JWST project, we have begun to merge the Cosi Properties into the Tina Fields, allowing Tina Fields to be used to force Constraints to rerun.

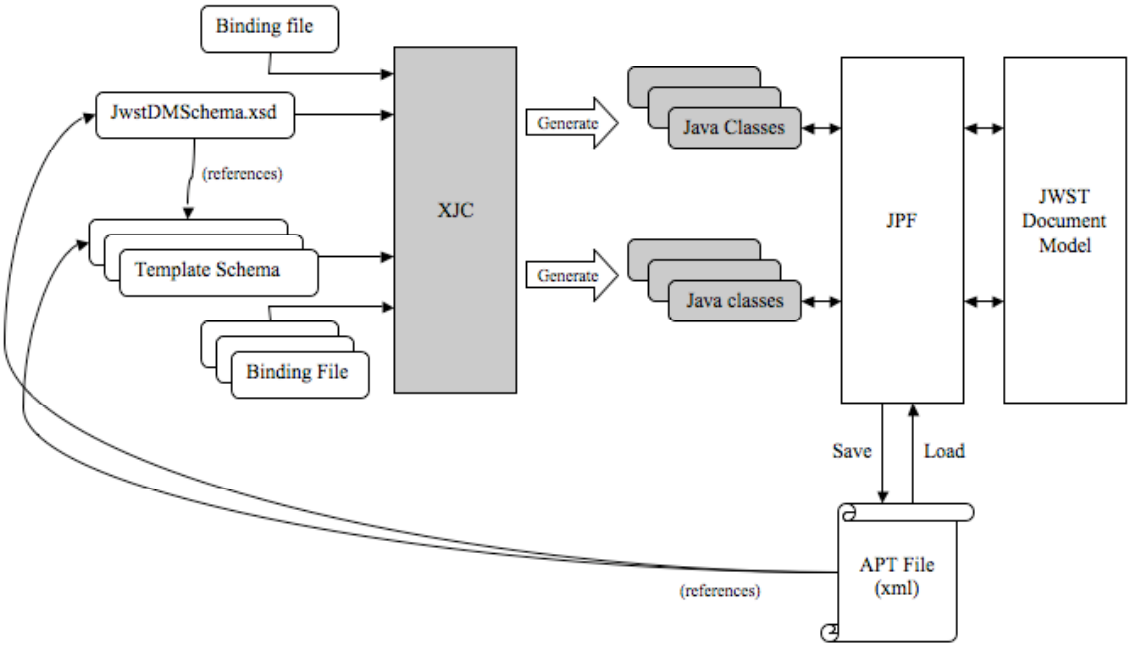
Topic

- ☐ Overview of JAXB/XML
  - ☐ XML Schemas
    - ☐ JwstDMSchema
    - ☐ MiriImaging
    - ☐ MiriLRS
    - ☐ MiriMRS
    - ☐ MiriCoron
    - ☐ MiriDark
    - ☐ MiriImagingFlat
    - ☐ MiriMRSFlat
    - ☐ MiriAnneal
- ☐ Java Class Diagram
  - ☐ JwstObservation
    - ☐ JwstTemplateFactory
    - ☐ JwstTemplate
      - ☐ Has
        - ☐ JwstInstrument
          - ☐ MiriInstrument
      - ☐ Subclasses
        - ☐ MiriImagingTemplate
        - ☐ MiriLrsTemplate
        - ☐ MiriMrsTemplate
        - ☐ MiriCoronImagingTemplate
        - ☐ MiriDarkTemplate
        - ☐ MiriImagingFlatTemplate
        - ☐ MiriMrsFlatTemplate
        - ☐ MiriAnnealTemplate
      - ☐ Other Helper Classes
        - ☐ MiriTargetAcq
        - ☐ MiriExposureTable
          - ☐ MiriImagingExposureTable
          - ☐ MiriCoronExposureTable
          - ☐ MiriDarkExposureTable
        - ☐ MiriExposure
          - ☐ MiriExposureTableRow
            - ☐ MiriImagingExposureTableRow
            - ☐ MiriCoronExposureTableRow
            - ☐ MiriDarkExposureTableRow
          - ☐ MiriLrsExposure
          - ☐ MiriMrsExposure
        - ☐ MiriExposureTimeCalculator

Save/Load Design with JAXB

Once schemas are defined, they are passed into the JAXB XJC processor to generate JAXB Java classes. These generated classes are trivially marshaled to and unmarshaled from XML files through JAXB libraries.

In order to save the file, the JAXB classes must be populated from the Document Model. This is the responsibility of the JWST Proposal File (JPF) class. Once the JAXB classes are populated, they are marshaled to an XML file. Loading is the reverse of this process: XML is unmarshaled into JAXB classes, and the JPF converts them into the Document Model classes.



XML Schema Design

The root JWST Schema is JwstDMSchema.xsd. In order to provide extensible Templates, each Template defines its own schema in a separate XSD file. For example, the Miri Imaging template is defined in MiriImaging.xsd. The root schema references all template schemas which allows it to completely define the structure for a proposal containing multiple templates.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- As a rule, the schema "version" attribute must be an integer -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.stsci.edu/JWST/APT"
    xmlns="http://www.stsci.edu/JWST/APT"
    xmlns:mit="http://www.stsci.edu/JWST/APT/Template/MiriImaging"
    xmlns:mmrsf="http://www.stsci.edu/JWST/APT/Template/MiriMRSFlat"
    xmlns:mann="http://www.stsci.edu/JWST/APT/Template/MiriAnneal"
    xmlns:md="http://www.stsci.edu/JWST/APT/Template/MiriDark"
    xmlns:mif="http://www.stsci.edu/JWST/APT/Template/MiriImagingFlat"
    xmlns:mc="http://www.stsci.edu/JWST/APT/Template/MiriCoron"
    xmlns:mlrs="http://www.stsci.edu/JWST/APT/Template/MiriLRS"
    xmlns:mmrs="http://www.stsci.edu/JWST/APT/Template/MiriMRS"

    elementFormDefault="qualified" version="1">

    <xs:import namespace="http://www.stsci.edu/JWST/APT/Template/MiriImaging" schemaLocation="TemplateSchemas/MiriImaging/MiriImaging.xsd"/>
    <xs:import namespace="http://www.stsci.edu/JWST/APT/Template/MiriMRSFlat" schemaLocation="TemplateSchemas/MiriMRSFlat/MiriMRSFlat.xsd"/>
    <xs:import namespace="http://www.stsci.edu/JWST/APT/Template/MiriAnneal" schemaLocation="TemplateSchemas/MiriAnneal/MiriAnneal.xsd"/>
    <xs:import namespace="http://www.stsci.edu/JWST/APT/Template/MiriDark" schemaLocation="TemplateSchemas/MiriDark/MiriDark.xsd"/>
    <xs:import namespace="http://www.stsci.edu/JWST/APT/Template/MiriImagingFlat" schemaLocation="TemplateSchemas/MiriImagingFlat/MiriImagingFlat.xsd"/>
    <xs:import namespace="http://www.stsci.edu/JWST/APT/Template/MiriCoron" schemaLocation="TemplateSchemas/MiriCoron/MiriCoron.xsd"/>
    <xs:import namespace="http://www.stsci.edu/JWST/APT/Template/MiriLRS" schemaLocation="TemplateSchemas/MiriLRS/MiriLRS.xsd"/>
    <xs:import namespace="http://www.stsci.edu/JWST/APT/Template/MiriMRS" schemaLocation="TemplateSchemas/MiriMRS/MiriMRS.xsd"/>

    <!--  ********************************************** -->
    <!--                  Root Element                  -->
    <!--  ********************************************** -->

    <xs:element name="JwstProposal">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="ProposalHistory" type="ProposalHistoryType" minOccurs="0"/>
                <xs:element name="ProposalID" type="xs:int" minOccurs="0"/>
                <xs:element name="ProposalInformation" type="ProposalInformation" minOccurs="0"/>
                <xs:element name="Targets" type="Targets" minOccurs="0"/>
                <xs:element name="DataRequests" type="DataRequestsType" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="schemaVersion" type="xs:int" use="required"/>
        </xs:complexType>
    </xs:element>

    <!--  ********************************************** -->
    <!--                  Complex Types                 -->
    <!--  ********************************************** -->

    <xs:complexType name="ProposalHistoryType">
        <xs:sequence>
            <xs:element name="HistoryEntry" type="HistoryEntryType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="HistoryEntryType">
        <xs:attribute name="Date" type="xs:string"/>
        <xs:attribute name="APTVersion" type="xs:string"/>
        <xs:attribute name="ProposalPhase" type="xs:string"/>
        <xs:attribute name="User" type="xs:string"/>
        <xs:attribute name="Platform" type="xs:string"/>
    </xs:complexType>
```

```
<xs:complexType name="Targets">
    <xs:sequence>
        <xs:element name="Target" type="AbstractTargetType" minOccurs="0" maxOccurs="unbounded"
        />
    </xs:sequence>
</xs:complexType>


<xs:complexType name="ProposalInformation">
    <xs:sequence>
        <xs:element name="Title" type="xs:string" minOccurs="0"/>
        <xs:element name="Abstract" type="xs:string" minOccurs="0"/>
        <xs:element name="ProposalID" type="xs:int" minOccurs="0"/>
        <xs:element name="ProposalCategory" type="ProposalCategoryType" minOccurs="0"/>
        <xs:element name="ProposalCategorySubtype" type="ProposalCategorySubtypeType" minOccurs="0"/>
        <xs:element name="ScientificCategory" type="ScientificCategoryType" minOccurs="0"/>
        <xs:element name="ScientificKeyword" type="ScientificKeywordType" minOccurs="0"
            maxOccurs="unbounded"/>
        <xs:element name="Cycle" type="xs:int" minOccurs="0"/>
        <xs:element name="RequestedTime" type="RequestedTimeType" minOccurs="0"
            maxOccurs="unbounded"/>
        <xs:element name="Calibration" type="xs:boolean" minOccurs="0"/>
        <xs:element name="PdfAttachment" type="xs:anyURI" minOccurs="0"/>
        <xs:element name="STScIEditingNumber" type="xs:int" minOccurs="0"/>
        <xs:element name="Availability" type="AvailabilityType" minOccurs="0"/>
        <xs:element name="ObservingDescription" type="xs:string" minOccurs="0"/>
        <xs:element name="AdditionalComments" type="xs:string" minOccurs="0"/>
        <xs:element name="PrincipalInvestigator" type="PrincipalInvestigator" minOccurs="0"/>
        <xs:element name="CoInvestigator" type="CoInvestigator" minOccurs="0"
            maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="PrincipalInvestigator">
    <xs:sequence>
        <xs:element name="InvestigatorAddress" type="InvestigatorAddressType" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="CoInvestigator">
    <xs:sequence>
        <xs:element name="InvestigatorAddress" type="InvestigatorAddressType" minOccurs="0"/>
        <xs:element name="AdminUSPI" type="xs:boolean" minOccurs="0"/>
        <xs:element name="Contact" type="xs:boolean" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="InvestigatorAddressType">
    <xs:sequence>
        <xs:element name="Country" type="xs:string" minOccurs="0"/>
        <xs:element name="EMail" type="xs:string" minOccurs="0"/>
        <xs:element name="Institution" type="xs:string" minOccurs="0"/>
        <xs:element name="USState" type="xs:string" minOccurs="0"/>
        <xs:element name="ESAMember" type="xs:boolean" minOccurs="0"/>
        <xs:element name="FirstName" type="xs:string" minOccurs="0"/>
        <xs:element name="Honorific" type="xs:string" minOccurs="0"/>
        <xs:element name="LastName" type="xs:string" minOccurs="0"/>
        <xs:element name="MiddleInitial" type="xs:string" minOccurs="0"/>
```

```
        <xs:element name="Suffix" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="ProposalCategoryType">
    <xs:choice>
        <xs:element name="GO" type="GOType" minOccurs="0"/>
        <xs:element name="AR" type="ARType" minOccurs="0"/>
        <xs:element name="GODD" type="GODDType" minOccurs="0"/>
        <xs:element name="GTO" type="GTOType" minOccurs="0"/>
        <xs:element name="CAL" type="CALType" minOccurs="0"/>
        <xs:element name="ENG" type="ENGType" minOccurs="0"/>
        <xs:element name="NASA" type="NASAType" minOccurs="0"/>
    </xs:choice>
</xs:complexType>

<xs:simpleType name="ProposalCategorySubtypeType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="TFI"/>
        <xs:enumeration value="FGS"/>
        <xs:enumeration value="SC"/>
        <xs:enumeration value="NIRSPEC"/>
        <xs:enumeration value="NIRCAM"/>
        <xs:enumeration value="MIRI"/>
    </xs:restriction>
</xs:simpleType>

<xs:complexType name="AbstractGOType" abstract="true">
    <xs:sequence>
        <xs:element name="RequestedTime" type="DoubleString" minOccurs="0"/>
        <xs:element name="ProprietaryPeriod" minOccurs="0">
            <xs:simpleType>
                <xs:restriction base="xs:int">
                    <xs:enumeration value="0"/>
                    <xs:enumeration value="3"/>
                    <xs:enumeration value="6"/>
                    <xs:enumeration value="12"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="CoordinatedObservation" type="CoordinatedObservationType"
            minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="Treasury" type="xs:boolean" minOccurs="0"/>
        <xs:element name="Large" type="xs:boolean" minOccurs="0"/>
        <xs:element name="Survey" type="xs:boolean" minOccurs="0"/>
        <xs:element name="Duplication" type="xs:boolean" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="ARType">
    <xs:sequence>
        <xs:element name="Budget" minOccurs="0">
            <xs:simpleType>
                <xs:restriction base="xs:decimal">
                    <xs:fractionDigits value="2"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="BudgetExplanation" minOccurs="0">
```

```
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:maxLength value="40"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="Theory" type="xs:boolean" minOccurs="0"/>
        <xs:element name="Legacy" type="xs:boolean" minOccurs="0"/>
        <xs:element name="DataSet" type="DataSetType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="GTOType">
    <xs:complexContent>
        <xs:extension base="AbstractGOType">
            <xs:sequence/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="CALType">
    <xs:complexContent>
        <xs:extension base="AbstractGOType">
            <xs:sequence/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="ENGType">
    <xs:complexContent>
        <xs:extension base="AbstractGOType">
            <xs:sequence/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="GOType">
    <xs:complexContent>
        <xs:extension base="AbstractGOType">
            <xs:sequence/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="GODDType">
    <xs:complexContent>
        <xs:extension base="AbstractGOType">
            <xs:sequence/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="NASAType"/>

<xs:complexType name="RequestedTimeType">
    <xs:sequence>
        <xs:element name="Cycle" type="xs:int" minOccurs="0"/>
        <xs:element name="RequestedTime" type="DoubleString" minOccurs="0"/>
    </xs:sequence>
```

```
</xs:complexType>

<xs:complexType name="CoordinatedObservationType">
    <xs:sequence>
        <xs:element name="Observatory" type="xs:string" minOccurs="0"/>
        <xs:element name="RequestedTime" type="DoubleString" minOccurs="0"/>
        <xs:element name="Details" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="DataSetType">
    <xs:sequence>
        <xs:element name="Number" type="xs:int" minOccurs="0"/>
        <xs:element name="Instrument" type="InstrumentType" minOccurs="0"/>
        <xs:element name="NumberOfDataSets" type="xs:int" minOccurs="0"/>
        <xs:element name="RetrievalMethod" type="RetrievalMethodType" minOccurs="0"/>
        <xs:element name="RetrievalPlan" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="AbstractTargetType" abstract="true">
    <xs:sequence>
        <xs:element name="Number" type="xs:int" minOccurs="0"/>
        <xs:element name="TargetName" type="xs:string" minOccurs="0"/>
        <xs:element name="TargetID" type="xs:string" minOccurs="0"/>
        <xs:element name="Description" type="TargetDescriptionType" minOccurs="0"/>
        <xs:element name="Fluxes" type="FluxesType" minOccurs="0"/>
        <xs:element name="Comments" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="AbstractSelectedTargetType" abstract="true">
    <xs:complexContent>
        <xs:extension base="AbstractTargetType">
            <xs:sequence>
                <xs:element name="Redshift" type="DoubleString" minOccurs="0"/>
                <xs:element name="RAProperMotion" type="DoubleString" minOccurs="0"/>
                <xs:element name="DecProperMotion" type="DoubleString" minOccurs="0"/>
                <xs:element name="Epoch" type="DoubleString" minOccurs="0"/>
                <xs:element name="AnnualParallax" type="DoubleString" minOccurs="0"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="FluxesType">
    <xs:sequence>
        <xs:element name="BroadBandMagnitude" type="BroadBandMagnitudeType" minOccurs="0"
            maxOccurs="unbounded"/>
        <xs:element name="OtherFluxes" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="BroadBandMagnitudeType">
    <xs:sequence>
        <xs:element name="Band" minOccurs="0">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="V"/>
```

```
                    <xs:enumeration value="R"/>
                    <xs:enumeration value="I"/>
                    <xs:enumeration value="J"/>
                    <xs:enumeration value="H"/>
                    <xs:enumeration value="K"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="Magnitude" type="DoubleString" minOccurs="0"/>
        <xs:element name="MagnitudeUnc" type="DoubleString" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="FixedTargetType">
    <xs:complexContent>
        <xs:extension base="AbstractSelectedTargetType">
            <xs:sequence>
                <xs:element name="EquatorialCoordinates" type="EquatorialCoordinatesType" minOccurs="0"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="OffsetTargetType">
    <xs:complexContent>
        <xs:extension base="AbstractSelectedTargetType">
            <xs:sequence>
                <xs:element name="OffsetCoordinates" type="OffsetCoordinatesType" minOccurs="0"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="GenericTargetType">
    <xs:complexContent>
        <xs:extension base="AbstractTargetType">
            <xs:sequence>
                <xs:element name="Criteria" type="xs:string" minOccurs="0"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="EquatorialCoordinatesType">
    <xs:sequence>
        <xs:element name="RA" type="RAType" minOccurs="0"/>
        <xs:element name="RAUncertainty" type="RAUncertaintyType" minOccurs="0"/>
        <xs:element name="Dec" type="DecType" minOccurs="0"/>
        <xs:element name="DecUncertainty" type="DecUncertaintyType" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="RAType">
    <xs:attribute name="Hours" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:int">
                <xs:minExclusive value="-24"/>
                <xs:maxExclusive value="24"/>
            </xs:restriction>
```

```
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Minutes" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:int">
                <xs:minInclusive value="0"/>
                <xs:maxInclusive value="59"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Seconds" type="DoubleString" use="required"/>
</xs:complexType>

<xs:complexType name="RAUncertaintyType">
    <xs:sequence>
        <xs:element name="Value" type="DoubleString" minOccurs="0"/>
        <xs:element name="Units" minOccurs="0">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="Arcminutes"/>
                    <xs:enumeration value="Arcseconds"/>
                    <xs:enumeration value="Minutes"/>
                    <xs:enumeration value="Seconds"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="DecType">
    <xs:attribute name="Degrees" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:int">
                <xs:minExclusive value="-90"/>
                <xs:maxExclusive value="90"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Minutes" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:int">
                <xs:minInclusive value="0"/>
                <xs:maxInclusive value="59"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Seconds" type="DoubleString" use="required"/>
</xs:complexType>

<xs:complexType name="DecUncertaintyType">
    <xs:sequence>
        <xs:element name="Value" type="DoubleString" minOccurs="0"/>
        <xs:element name="Units" minOccurs="0">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="Arcminutes"/>
                    <xs:enumeration value="Arcseconds"/>
                    <xs:enumeration value="Degrees"/>
                </xs:restriction>
```

```
            </xs:simpleType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="OffsetCoordinatesType">
    <xs:sequence>
        <xs:element name="RA" type="RAOffsetType" minOccurs="0"/>
        <xs:element name="Dec" type="DecOffsetType" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="RAOffsetType">
    <xs:sequence>
        <xs:element name="Offset" type="DoubleString" minOccurs="0"/>
        <xs:element name="Uncertainty" type="DoubleString" minOccurs="0"/>
        <xs:element name="Units" minOccurs="0">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="Seconds"/>
                    <xs:enumeration value="Degrees"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="DecOffsetType">
    <xs:sequence>
        <xs:element name="Offset" type="DoubleString" minOccurs="0"/>
        <xs:element name="Uncertainty" type="DoubleString" minOccurs="0"/>
        <xs:element name="Units" minOccurs="0">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="Arcminutes"/>
                    <xs:enumeration value="Arcseconds"/>
                    <xs:enumeration value="Degrees"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="TargetDescriptionType">
    <xs:choice>
        <xs:element name="SolarSystem" type="SolarSystemTargetType"/>
        <xs:element name="Star" type="StarTargetType"/>
        <xs:element name="ExtStar" type="ExtStarTargetType"/>
        <xs:element name="StellarCluster" type="StellarClusterTargetType"/>
        <xs:element name="ExtCluster" type="ExtClusterTargetType"/>
        <xs:element name="Galaxy" type="GalaxyTargetType"/>
        <xs:element name="ClusterOfGalaxies" type="ClusterOfGalaxiesTargetType"/>
        <xs:element name="ISM" type="ISMTargetType"/>
        <xs:element name="ExtMedium" type="ExtMediumTargetType"/>
        <xs:element name="Unidentified" type="UnidentifiedTargetType"/>
        <xs:element name="Calibration" type="CalibrationTargetType"/>
    </xs:choice>
</xs:complexType>
```

```
<xs:complexType name="SolarSystemTargetType">
    <xs:sequence>
        <xs:element name="Keyword" minOccurs="0" maxOccurs="unbounded">
            <xs:simpleType>
                <xs:union memberTypes="AllTargetKeyword SolarSystemTargetKeyword"/>
            </xs:simpleType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="StarTargetType">
    <xs:sequence>
        <xs:element name="Keyword" minOccurs="0" maxOccurs="unbounded">
            <xs:simpleType>
                <xs:union memberTypes="AllTargetKeyword StarTargetKeyword"/>
            </xs:simpleType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="ExtStarTargetType">
    <xs:sequence>
        <xs:element name="Keyword" minOccurs="0" maxOccurs="unbounded">
            <xs:simpleType>
                <xs:union memberTypes="AllTargetKeyword StarTargetKeyword"/>
            </xs:simpleType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="StellarClusterTargetType">
    <xs:sequence>
        <xs:element name="Keyword" minOccurs="0" maxOccurs="unbounded">
            <xs:simpleType>
                <xs:union memberTypes="AllTargetKeyword StellarClusterTargetKeyword"/>
            </xs:simpleType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="ExtClusterTargetType">
    <xs:sequence>
        <xs:element name="Keyword" minOccurs="0" maxOccurs="unbounded">
            <xs:simpleType>
                <xs:union memberTypes="AllTargetKeyword StellarClusterTargetKeyword"/>
            </xs:simpleType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="GalaxyTargetType">
    <xs:sequence>
        <xs:element name="Keyword" minOccurs="0" maxOccurs="unbounded">
            <xs:simpleType>
                <xs:union memberTypes="AllTargetKeyword GalaxyTargetKeyword"/>
            </xs:simpleType>
        </xs:element>
    </xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="ClusterOfGalaxiesTargetType">
    <xs:sequence>
        <xs:element name="Keyword" minOccurs="0" maxOccurs="unbounded">
            <xs:simpleType>
                <xs:union memberTypes="AllTargetKeyword ClusterOfGalaxiesTargetKeyword"/>
            </xs:simpleType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="ISMTargetType">
    <xs:sequence>
        <xs:element name="Keyword" minOccurs="0" maxOccurs="unbounded">
            <xs:simpleType>
                <xs:union memberTypes="AllTargetKeyword ISMTargetKeyword"/>
            </xs:simpleType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="ExtMediumTargetType">
    <xs:sequence>
        <xs:element name="Keyword" minOccurs="0" maxOccurs="unbounded">
            <xs:simpleType>
                <xs:union memberTypes="AllTargetKeyword ISMTargetKeyword"/>
            </xs:simpleType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="UnidentifiedTargetType">
    <xs:sequence>
        <xs:element name="Keyword" minOccurs="0" maxOccurs="unbounded">
            <xs:simpleType>
                <xs:union memberTypes="AllTargetKeyword UnidentifiedTargetKeyword"/>
            </xs:simpleType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="CalibrationTargetType">
    <xs:sequence>
        <xs:element name="Keyword" minOccurs="0" maxOccurs="unbounded">
            <xs:simpleType>
                <xs:union memberTypes="AllTargetKeyword CalibrationTargetKeyword"/>
            </xs:simpleType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="DataRequestsType">
    <xs:sequence>
        <xs:element name="ObservationGroup" type="ObservationGroupType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="ObservationGroupType">
    <xs:sequence>
```

```
            <xs:element name="Label" type="xs:string" minOccurs="0"/>
            <xs:element name="Comments" type="xs:string" minOccurs="0"/>
            <xs:element name="Observation" type="ObservationType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ObservationType">
        <xs:sequence>
            <xs:element name="Number" type="xs:int" minOccurs="0"/>
            <xs:element name="TargetID" type="xs:string" minOccurs="0"/>
            <xs:element name="Instrument" type="InstrumentType" minOccurs="0"/>
            <xs:element name="Template" type="TemplateType" minOccurs="0"/>
            <xs:element name="Comments" type="xs:string" minOccurs="0"/>
            <xs:element name="MosaicParameters" type="MosaicParametersType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="TemplateType">
        <xs:choice>
            <xs:element ref="mit:MiriImaging" minOccurs="0"/>
            <xs:element ref="mmrsf:MiriMRSFlat" minOccurs="0"/>
            <xs:element ref="mann:MiriAnneal" minOccurs="0"/>
            <xs:element ref="md:MiriDark" minOccurs="0"/>
            <xs:element ref="mif:MiriImagingFlat" minOccurs="0"/>
            <xs:element ref="mc:MiriCoron" minOccurs="0"/>
            <xs:element ref="mlrs:MiriLRS" minOccurs="0"/>
            <xs:element ref="mmrs:MiriMRS" minOccurs="0"/>
        </xs:choice>
    </xs:complexType>

    <xs:complexType name="MosaicParametersType">
        <xs:sequence>
            <xs:element name="Rows" type="xs:int" minOccurs="0"/>
            <xs:element name="Columns" type="xs:int" minOccurs="0"/>
            <xs:element name="RowOverlapPercent" type="DoubleString" minOccurs="0"/>
            <xs:element name="ColumnOverlapPercent" type="DoubleString" minOccurs="0"/>
            <xs:element name="SkewDegreesX" type="DoubleString" minOccurs="0"/>
            <xs:element name="SkewDegreesY" type="DoubleString" minOccurs="0"/>
            <xs:element name="PatternOrient" type="DoubleString" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <!--  ********************************************** -->
    <!--                   Simple Types                 -->
    <!--  ********************************************** -->

    <xs:simpleType name="StandardTargetNameType">
        <xs:restriction base="xs:string">
            <xs:pattern value="([A-Z0-9\-\.\+])*"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="DoubleString">
        <xs:restriction base="xs:string">
            <!-- This RegEx is a close approximation to the built-in regular expression in the Java Double valueOf method. -->
            <xs:pattern value="[+\-]?(NaN|Infinity|((((\p{Nd}+)(\.)?((\p{Nd}+)?)([eE][+\-]?(\p{Nd}+))?)|(\.((\p{Nd}+))([eE][+\-]?(\p{Nd}+))?)|(((0[xX](\p{Nl}
+)(\.)?)|(0[xX](\p{Nl}+)?(\.)(\p{Nl}+)))[pP][+\-]?(\p{Nd}+)))[fFdD]?))"/>
        </xs:restriction>
    </xs:simpleType>
```

```
<!--  ********************************************** -->
<!--                   Enumerations                 -->
<!--  ********************************************** -->

<xs:simpleType name="AllTargetKeyword">
    <xs:restriction base="xs:string">
        <xs:enumeration value="AllTargetKeywords Go Here"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="SolarSystemTargetKeyword">
    <xs:restriction base="xs:string">
        <xs:enumeration value="SolarSystemTargetKeywords Go Here"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="StarTargetKeyword">
    <xs:restriction base="xs:string">
        <xs:enumeration value="StarTargetKeywords Go Here"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="StellarClusterTargetKeyword">
    <xs:restriction base="xs:string">
        <xs:enumeration value="StellarClusterTargetKeywords Go Here"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="GalaxyTargetKeyword">
    <xs:restriction base="xs:string">
        <xs:enumeration value="GalaxyTargetKeywords Go Here"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="ClusterOfGalaxiesTargetKeyword">
    <xs:restriction base="xs:string">
        <xs:enumeration value="ClusterOfGalaxiesTargetKeywords Go Here"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="ISMTargetKeyword">
    <xs:restriction base="xs:string">
        <xs:enumeration value="ISMTargetKeywords Go Here"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="UnidentifiedTargetKeyword">
    <xs:restriction base="xs:string">
        <xs:enumeration value="UnidentifiedTargetKeywords Go Here"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="CalibrationTargetKeyword">
    <xs:restriction base="xs:string">
        <xs:enumeration value="CalibrationTargetKeywords Go Here"/>
    </xs:restriction>
</xs:simpleType>
```

```
    <xs:simpleType name="ScientificCategoryType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="EXTRA-SOLAR PLANETS"/>
            <xs:enumeration value="STAR FORMATION"/>
            <xs:enumeration value="COOL STARS"/>
            <xs:enumeration value="HOT STARS"/>
            <xs:enumeration value="ISM AND CIRCUMSTELLAR MATTER"/>
            <xs:enumeration value="RESOLVED STELLAR POPULATIONS"/>
            <xs:enumeration value="UNRESOLVED STELLAR POPULATIONS AND GALAXY STRUCTURE"/>
            <xs:enumeration value="ISM IN EXTERNAL GALAXIES"/>
            <xs:enumeration value="AGN/QUASARS"/>
            <xs:enumeration value="QUASAR ABSORPTION LINES AND IGM"/>
            <xs:enumeration value="COSMOLOGY"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="ScientificKeywordType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="SciKeywd1"/>
            <xs:enumeration value="SciKeywd2"/>
            <xs:enumeration value="SciKeywd3"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="AvailabilityType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="SUPPORTED"/>
            <xs:enumeration value="TBD1"/>
            <xs:enumeration value="TBD2"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="InstrumentType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="MIRI"/>
            <xs:enumeration value="NIRCAM"/>
            <xs:enumeration value="NIRSPEC"/>
            <xs:enumeration value="FGS"/>
            <xs:enumeration value="TFI"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="RetrievalMethodType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="CD"/>
            <xs:enumeration value="DVD"/>
            <xs:enumeration value="FTP"/>
            <xs:enumeration value="DISK"/>
        </xs:restriction>
    </xs:simpleType>

</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns="http://www.stsci.edu/JWST/APT/Template/MiriImaging"
    targetNamespace="http://www.stsci.edu/JWST/APT/Template/MiriImaging"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    version="1">
    <xs:element name="MiriImaging">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="ObjectType" type="ObjectTypeType" minOccurs="0"/>
                <xs:element name="Subarray" type="SubarrayType" minOccurs="0"/>
                <xs:element name="Filters" type="FiltersType" minOccurs="0"/>
                <xs:element name="Dither" type="DitherType" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="FiltersType">
        <xs:sequence>
            <xs:element name="FilterConfig" type="FilterConfigType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="FilterConfigType">
        <xs:sequence>
            <xs:element name="Filter" type="FilterType" minOccurs="0"/>
            <xs:element name="RequestedExposureTime" type="xs:double" minOccurs="0"/>
            <xs:element name="ReadoutPattern" type="ReadoutPatternType" minOccurs="0"/>
            <xs:element name="Groups" type="xs:int" minOccurs="0"/>
            <xs:element name="Integrations" type="xs:int" minOccurs="0"/>
            <xs:element name="ActualExposureTime" type="xs:double" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="DitherType">
        <xs:choice>
            <xs:element name="PredefinedDither" type="PredefinedDitherType" minOccurs="0"/>
            <xs:element name="CustomDither" type="CustomDitherType" minOccurs="0">
            </xs:element>
        </xs:choice>
    </xs:complexType>

    <xs:complexType name="CustomDitherType">
        <xs:sequence>
            <xs:element name="DitherPoint" type="DitherPointType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="DitherPointType">
        <xs:sequence>
            <xs:element name="XOffset" type="xs:double" minOccurs="0"/>
            <xs:element name="YOffset" type="xs:double" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="PredefinedDitherType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Dither Pattern 1"/>
```

```
            <xs:enumeration value="Dither Pattern 2"/>
            <xs:enumeration value="Dither Pattern 3"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="ReadoutPatternType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="FAST"/>
            <xs:enumeration value="SLOW"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="ObjectTypeType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="FAINT"/>
            <xs:enumeration value="BRIGHT"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="SubarrayType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="FULL"/>
            <xs:enumeration value="BRIGHTSKY"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="FilterType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="F560W"/>
            <xs:enumeration value="F770W"/>
            <xs:enumeration value="F1000W"/>
            <xs:enumeration value="F1130W"/>
            <xs:enumeration value="F1280W"/>
            <xs:enumeration value="F1500W"/>
            <xs:enumeration value="F1800W"/>
            <xs:enumeration value="F2100W"/>
            <xs:enumeration value="F2550W"/>
            <xs:enumeration value="F2550WR"/>
            <xs:enumeration value="FND"/>
            <xs:enumeration value="FLENS"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns="http://www.stsci.edu/JWST/APT/Template/MiriLRS"
    targetNamespace="http://www.stsci.edu/JWST/APT/Template/MiriLRS"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    version="1">
    <xs:element name="MiriLRS">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="AcqTargetID" type="xs:string" minOccurs="0"/>
                <xs:element name="AcqFilter" type="AcqFilterType" minOccurs="0"/>
                <xs:element name="AcqFlux" type="xs:string" minOccurs="0"/>
                <xs:element name="ObjectType" type="ObjectTypeType" minOccurs="0"/>
                <xs:element name="RequestedExposureTime" type="xs:double" minOccurs="0"/>
                <xs:element name="ReadoutPattern" type="ReadoutPatternType" minOccurs="0"/>
                <xs:element name="NumberOfGroups" type="xs:int" minOccurs="0"/>
                <xs:element name="NumberOfIntegrations" type="xs:int" minOccurs="0"/>
                <xs:element name="CalculatedExposureTime" type="xs:double" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:simpleType name="AcqFilterType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="F560W"/>
            <xs:enumeration value="F770W"/>
            <xs:enumeration value="F1000W"/>
            <xs:enumeration value="F1065C"/>
            <xs:enumeration value="F1130W"/>
            <xs:enumeration value="F1140C"/>
            <xs:enumeration value="F1280W"/>
            <xs:enumeration value="F1500W"/>
            <xs:enumeration value="F1550C"/>
            <xs:enumeration value="F1800W"/>
            <xs:enumeration value="F2100W"/>
            <xs:enumeration value="F2300C"/>
            <xs:enumeration value="F2550W"/>
            <xs:enumeration value="F2550WR"/>
            <xs:enumeration value="P750L"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="ObjectTypeType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="FAINT"/>
            <xs:enumeration value="BRIGHT"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="ReadoutPatternType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="FAST"/>
            <xs:enumeration value="SLOW"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns="http://www.stsci.edu/JWST/APT/Template/MiriMRS"
    targetNamespace="http://www.stsci.edu/JWST/APT/Template/MiriMRS"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    version="1">
    <xs:element name="MiriMRS">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="AcqTargetID" type="xs:string" minOccurs="0"/>
                <xs:element name="AcqFilter" type="AcqFilterType" minOccurs="0"/>
                <xs:element name="AcqFlux" type="xs:string" minOccurs="0"/>
                <xs:element name="ObjectType" type="ObjectTypeType" minOccurs="0"/>
                <xs:element name="Wavelength" type="WavelengthType" minOccurs="0"/>
                <xs:element name="RequestedExposureTime" type="xs:double" minOccurs="0"/>
                <xs:element name="ReadoutPattern" type="ReadoutPatternType" minOccurs="0"/>
                <xs:element name="NumberOfGroups" type="xs:int" minOccurs="0"/>
                <xs:element name="NumberOfIntegrations" type="xs:int" minOccurs="0"/>
                <xs:element name="CalculatedExposureTime" type="xs:double" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:simpleType name="AcqFilterType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="F560W"/>
            <xs:enumeration value="F770W"/>
            <xs:enumeration value="F1000W"/>
            <xs:enumeration value="F1065C"/>
            <xs:enumeration value="F1130W"/>
            <xs:enumeration value="F1140C"/>
            <xs:enumeration value="F1280W"/>
            <xs:enumeration value="F1500W"/>
            <xs:enumeration value="F1550C"/>
            <xs:enumeration value="F1800W"/>
            <xs:enumeration value="F2100W"/>
            <xs:enumeration value="F2300C"/>
            <xs:enumeration value="F2550W"/>
            <xs:enumeration value="F2550WR"/>
            <xs:enumeration value="P750L"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="WavelengthType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="SHORT"></xs:enumeration>
            <xs:enumeration value="MEDIUM"></xs:enumeration>
            <xs:enumeration value="LONG"></xs:enumeration>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="ObjectTypeType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="FAINT"/>
            <xs:enumeration value="BRIGHT"/>
        </xs:restriction>
    </xs:simpleType>
```

```
    <xs:simpleType name="ReadoutPatternType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="FAST"/>
            <xs:enumeration value="SLOW"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns="http://www.stsci.edu/JWST/APT/Template/MiriCoron"
    targetNamespace="http://www.stsci.edu/JWST/APT/Template/MiriCoron"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    version="1">
    <xs:element name="MiriCoron">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="AcqTargetID" type="xs:string" minOccurs="0"/>
                <xs:element name="AcqFilter" type="FilterType" minOccurs="0"/>
                <xs:element name="AcqFilterFlux" type="xs:string" minOccurs="0"/>
                <xs:element name="Filter" type="FilterType" minOccurs="0"/>
                <xs:element name="ObjectType" type="ObjectTypeType" minOccurs="0"/>
                <xs:element name="Filters" type="FiltersType" minOccurs="0"/>

                <xs:element name="RequestedExposureTime" type="xs:double" minOccurs="0"/>
                <xs:element name="ReadoutPattern" type="ReadoutPatternType" minOccurs="0"/>
                <xs:element name="NumberOfGroups" type="xs:int" minOccurs="0"/>
                <xs:element name="NumberOfIntegrations" type="xs:int" minOccurs="0"/>
                <xs:element name="CalculatedExposureTime" type="xs:double" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="FiltersType">
        <xs:sequence>
            <xs:element name="FilterConfig" type="FilterConfigType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="FilterConfigType">
        <xs:sequence>
            <xs:element name="Mask" type="MaskType" minOccurs="0"/>
            <xs:element name="Filter" type="CoronFilterType" minOccurs="0"/>
            <xs:element name="RequestedExposureTime" type="xs:double" minOccurs="0"/>
            <xs:element name="ReadoutPattern" type="ReadoutPatternType" minOccurs="0"/>
            <xs:element name="NumberOfGroups" type="xs:int" minOccurs="0"/>
            <xs:element name="NumberOfIntegrations" type="xs:int" minOccurs="0"/>
            <xs:element name="CalculatedExposureTime" type="xs:double" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="MaskType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="FOUR_QPM"/>
            <xs:enumeration value="LYOT"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="CoronFilterType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="F1065C"/>
            <xs:enumeration value="F1140C"/>
            <xs:enumeration value="F1550C"/>
            <xs:enumeration value="F2300C"/>
        </xs:restriction>
    </xs:simpleType>
```

```
    <xs:simpleType name="FilterType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="F560W"/>
            <xs:enumeration value="F770W"/>
            <xs:enumeration value="F1000W"/>
            <xs:enumeration value="F1065C"/>
            <xs:enumeration value="F1130W"/>
            <xs:enumeration value="F1140C"/>
            <xs:enumeration value="F1280W"/>
            <xs:enumeration value="F1500W"/>
            <xs:enumeration value="F1550C"/>
            <xs:enumeration value="F1800W"/>
            <xs:enumeration value="F2100W"/>
            <xs:enumeration value="F2300C"/>
            <xs:enumeration value="F2550W"/>
            <xs:enumeration value="F2550WR"/>
            <xs:enumeration value="P750L"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="WavelengthType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="SHORT"></xs:enumeration>
            <xs:enumeration value="MEDIUM"></xs:enumeration>
            <xs:enumeration value="LONG"></xs:enumeration>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="ObjectTypeType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="FAINT"/>
            <xs:enumeration value="BRIGHT"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="ReadoutPatternType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="FAST"/>
            <xs:enumeration value="SLOW"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns="http://www.stsci.edu/JWST/APT/Template/MiriDark"
    targetNamespace="http://www.stsci.edu/JWST/APT/Template/MiriDark"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    version="1">
    <xs:element name="MiriDark">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Detector" type="DetectorType" minOccurs="0"/>
                <xs:element name="Filters" type="FiltersType" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="FiltersType">
        <xs:sequence>
            <xs:element name="FilterConfig" type="FilterConfigType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="FilterConfigType">
        <xs:sequence>
            <xs:element name="Filter" type="FilterType" minOccurs="0" />
            <xs:element name="ReadoutPattern" type="ReadoutPatternType"
                minOccurs="0" />
            <xs:element name="Groups" type="xs:int" minOccurs="0" />
            <xs:element name="Integrations" type="xs:int" minOccurs="0" />
            <xs:element name="NumberOfExposures" type="xs:int" maxOccurs="1" minOccurs="0"></xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="FilterType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="F560W"/>
            <xs:enumeration value="F770W"/>
            <xs:enumeration value="F1000W"/>
            <xs:enumeration value="F1065C"/>
            <xs:enumeration value="F1130W"/>
            <xs:enumeration value="F1140C"/>
            <xs:enumeration value="F1280W"/>
            <xs:enumeration value="F1500W"/>
            <xs:enumeration value="F1550C"/>
            <xs:enumeration value="F1800W"/>
            <xs:enumeration value="F2100W"/>
            <xs:enumeration value="F2300C"/>
            <xs:enumeration value="F2550W"/>
            <xs:enumeration value="F2550WR"/>
            <xs:enumeration value="P750L"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="DetectorType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="IMAGER"/>
            <xs:enumeration value="MRS"/>
            <xs:enumeration value="ALL"/>
        </xs:restriction>
```

```
    </xs:simpleType>

    <xs:simpleType name="ReadoutPatternType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="FAST"/>
            <xs:enumeration value="SLOW"/>
        </xs:restriction>
    </xs:simpleType>

</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns="http://www.stsci.edu/JWST/APT/Template/MiriImagingFlat"
    targetNamespace="http://www.stsci.edu/JWST/APT/Template/MiriImagingFlat"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    version="1">
    <xs:element name="MiriImagingFlat">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="FlatSuite" type="FlatSuiteType" minOccurs="0"/>
                <xs:element name="Filter" type="FilterType" minOccurs="0"/>
                <xs:element name="NumberOfIntegrations" type="xs:int" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:simpleType name="FlatSuiteType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="ALL"/>
            <xs:enumeration value="ONE"/>
            <xs:enumeration value="PRINCIPAL"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="FilterType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="F560W"/>
            <xs:enumeration value="F770W"/>
            <xs:enumeration value="F1000W"/>
            <xs:enumeration value="F1065C"/>
            <xs:enumeration value="F1130W"/>
            <xs:enumeration value="F1140C"/>
            <xs:enumeration value="F1280W"/>
            <xs:enumeration value="F1500W"/>
            <xs:enumeration value="F1550C"/>
            <xs:enumeration value="F1800W"/>
            <xs:enumeration value="F2100W"/>
            <xs:enumeration value="F2300C"/>
            <xs:enumeration value="F2550W"/>
            <xs:enumeration value="F2550WR"/>
            <xs:enumeration value="P750L"/>
        </xs:restriction>
    </xs:simpleType>

</xs:schema>
```

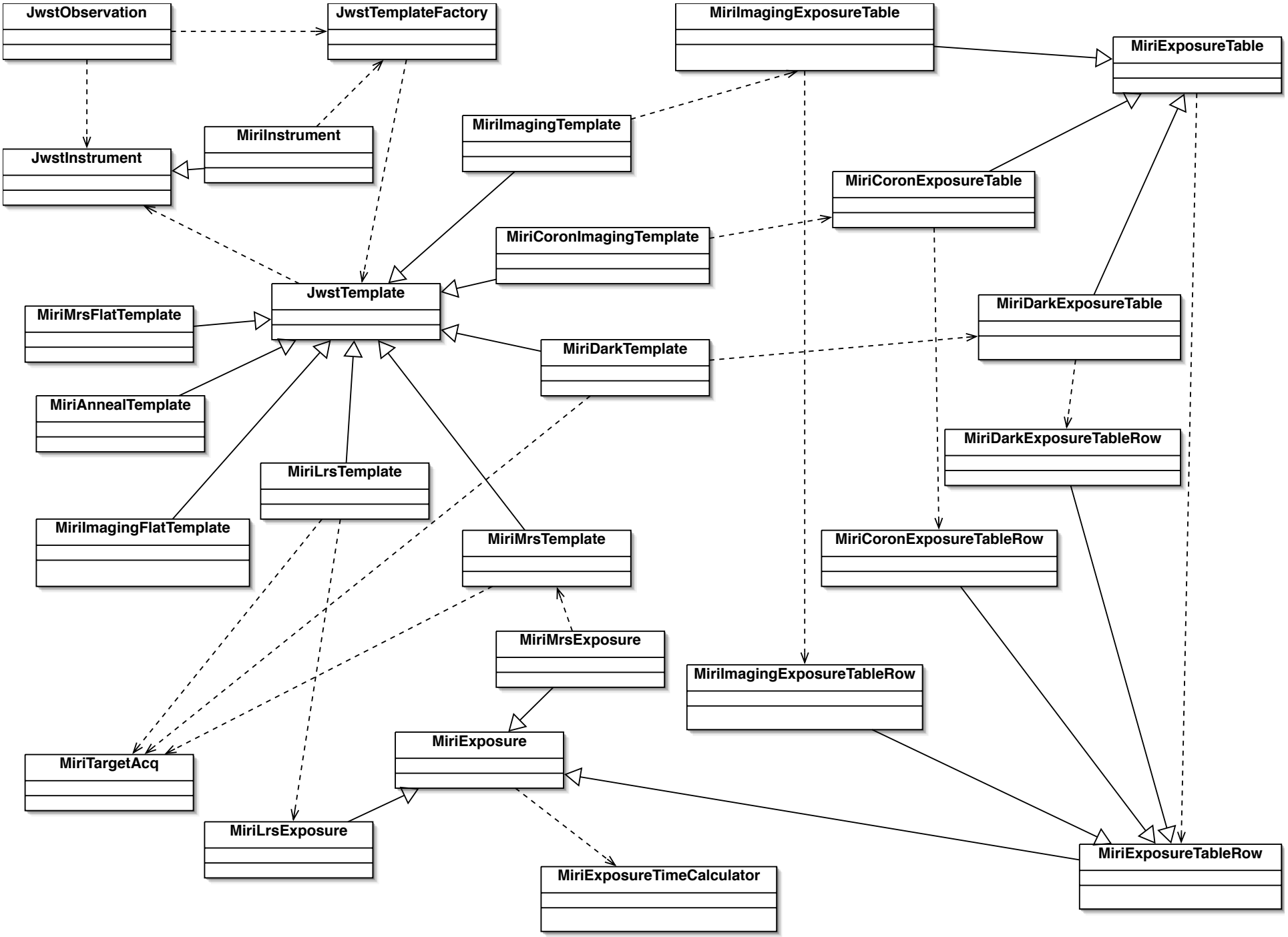
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns="http://www.stsci.edu/JWST/APT/Template/MiriMRSFlat"
    targetNamespace="http://www.stsci.edu/JWST/APT/Template/MiriMRSFlat"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    version="1">
    <xs:element name="MiriMRSFlat">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="FlatSuite" type="FlatSuiteType" minOccurs="0"/>
                <xs:element name="Wavelength" type="WavelengthType" minOccurs="0"/>
                <xs:element name="NumberOfIntegrations" type="xs:int" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:simpleType name="FlatSuiteType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="ONE"/>
            <xs:enumeration value="ALL"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="WavelengthType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="SHORT"/>
            <xs:enumeration value="MEDIUM"/>
            <xs:enumeration value="LONG"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns="http://www.stsci.edu/JWST/APT/Template/MiriAnneal"
    targetNamespace="http://www.stsci.edu/JWST/APT/Template/MiriAnneal"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    version="1">
    <xs:element name="MiriAnneal">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Imager" type="ImagerType" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:simpleType name="ImagerType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="IMAGER"/>
            <xs:enumeration value="MRS"/>
            <xs:enumeration value="ALL"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```


JwstObservation
JwstTemplateFactory
MiriImagingExposureTable
MiriExposureTable
MiriInstrument
JwstInstrument
MiriImagingTemplate
MiriCoronExposureTable
MiriCoronImagingTemplate
JwstTemplate
MiriMrsFlatTemplate
MiriDarkExposureTable
MiriDarkTemplate
MiriAnnealTemplate
MiriDarkExposureTableRow
MiriLrsTemplate
MiriImagingFlatTemplate
MiriMrsTemplate
MiriCoronExposureTableRow
MiriMrsExposure
MiriImagingExposureTableRow
MiriExposure
MiriTargetAcq
MiriLrsExposure
MiriExposureTableRow
MiriExposureTimeCalculator

```java
/**
package edu.stsci.jwst.apt.model;

import java.util.*;

/**
 * @author Rob Douglas
 *
 */
public class JwstObservation extends AbstractTinaDocumentElement {

    private static final String NUMBER = "Number";
    private static final String LABEL = "Label";
    private static final String COMMENTS = "Comments";
    private static final String TARGET = "Target";
    private static final String INSTRUMENT = "Instrument";
    private static final String TEMPLATE = "Template";
    public static final String TYPE_NAME = "Observation";
    public static ImageIcon ICON = null;
    static {
        try { ICON = new ImageIcon(JwstObservation.class.getResource
                ("/resources/images/ObservationIcon.gif"));
        } catch (Exception ex) {}
    }

    private final Hashtable<JwstTemplateFactory, JwstTemplate<? extends JwstInstrument>> templateTable =
        new Hashtable<JwstTemplateFactory, JwstTemplate<? extends JwstInstrument>>();

    private final CosiConstrainedInt number = new CosiConstrainedInt(this, NUMBER, true, 1, 999); {
        number.setEditable(false);
    }

    private TinaCosiStringField label = new TinaCosiStringField(this, LABEL, false);
    private BigString comments = new BigString(this, COMMENTS);

    private final TargetChooser targetChooser = new TargetChooser(this, TARGET);

    private static final List<JwstInstrument> INSTRUMENTS = new Vector<JwstInstrument>(); {
        for(JwstInstrument inst : JwstInstrument.getInstruments()) {
            INSTRUMENTS.add(inst);
        }
    }

    private JwstQuadPattern obsPattern = new JwstQuadPattern(); {
        add(obsPattern, false);
    }
```

```
private TinaField[] obsPatternProps = obsPattern.getProperties();
private TinaFieldGroupBox obsPatternBox = new TinaFieldGroupBox(this, "Observation Mosaic"); {
    for(TinaField lField : obsPatternProps) {
        TinaField lChildField = lField;
        while(lChildField.getTinaFieldGroup() != null) {
            lChildField = lChildField.getTinaFieldGroup();
        }
        if(obsPatternBox != lChildField) {
            lChildField.setTinaFieldGroup(obsPatternBox);
        }
    }
}

private JwstQuadPattern visitPattern = new JwstQuadPattern(); {
    add(visitPattern, false);
}
private TinaField[] visitPatternProps = visitPattern.getProperties();
private TinaFieldGroupBox visitPatternBox = new TinaFieldGroupBox(this, "Visit Mosaic"); {
    for(TinaField lField : visitPatternProps) {
        TinaField lChildField = lField;
        while(lChildField.getTinaFieldGroup() != null) {
            lChildField = lChildField.getTinaFieldGroup();
        }
        if(visitPatternBox != lChildField) {
            lChildField.setTinaFieldGroup(visitPatternBox);
        }
    }
}


private final CosiConstrainedSelection<JwstInstrument> instrumentChooser =
    new CosiConstrainedSelection<JwstInstrument>(this, INSTRUMENT, INSTRUMENTS, true);

private final CosiConstrainedSelection<JwstTemplateFactory> templateChooser =
    new CosiConstrainedSelection<JwstTemplateFactory>(this, TEMPLATE, true);
//TODO ROB ANDY Look at this with Andy's fixes to make sure the propagation fixes the infinite loop
//then move the adding of the template to the Observation to be in a separate Constraint.
private final CosiDerivedProperty<JwstTemplate<? extends JwstInstrument>> template =
    CosiDerivedProperty.createUninitializedProperty(this, null, new Calculator<JwstTemplate<? extends JwstInstrument>>(){
        public JwstTemplate<? extends JwstInstrument> calculate() {
            if (templateChooser.isSpecified()) {
                JwstTemplate<? extends JwstInstrument> lOldTemplate = template.get();
                if (lOldTemplate!=null) {
                    JwstObservation.this.remove(lOldTemplate);
                }
                JwstTemplate<? extends JwstInstrument> lNewTemplate = getTemplateInstance();
```

```
                if (lOldTemplate != lNewTemplate) {
                    JwstObservation.this.add(lNewTemplate, false);
                }
                return lNewTemplate;
            }
            return null;
        }
    });

{
    Propagator.addDelayedConstraint(new Constraint(this, "Legal Templates") {
        public void run () {
            List<JwstTemplateFactory> lTemplateFactories = new Vector<JwstTemplateFactory>();
            if (instrumentChooser.isSpecified()) {
                lTemplateFactories = JwstTemplateFactory.factoriesForIntrument(instrumentChooser.get());
            }
            templateChooser.setLegalValues(lTemplateFactories);
        }
    });

    Propagator.addDelayedConstraint(new Constraint(this, "Observation Properties") {
        public void run () {
            updateProperties();
        }
    });

    Propagator.addDelayedConstraint(new Constraint(this, "Ensure Visits"){
        public void run() {
            ensureVisits();
        }
    });
}

private final Map<Object,JwstVisit> visitManager = new HashMap<Object,JwstVisit>();

public JwstObservation () {
    Propagator.completeInitialization(this, JwstObservation.class);
}

private void updateProperties() {
    TinaField[] lBaseProps = {number, label, comments, targetChooser, instrumentChooser, templateChooser};
    TinaField[] lTemplateProps = {};

    if(getTemplate() != null) {
        lTemplateProps = getTemplate().getAttributes();
    }
```

```
        TinaField[] lProps = (TinaField[])ArrayUtils.addArrays(lBaseProps,lTemplateProps);
//        lProps = (TinaField[])ArrayUtils.addArrays(lProps, obsPatternProps);
//        lProps = (TinaField[])ArrayUtils.addArrays(lProps, visitPatternProps);
        setProperties(lProps);
    }

    private JwstTemplate<? extends JwstInstrument> getTemplateInstance () {
        JwstTemplateFactory lTemplateChoice = templateChooser.getValue();
        JwstTemplate<? extends JwstInstrument> lTemplate = templateTable.get(lTemplateChoice);
        if (lTemplate == null) {
            lTemplate = lTemplateChoice.makeInstance();
            templateTable.put(lTemplateChoice, lTemplate);
        }
        return lTemplate;
    }

    @Override
    public void elementInsertedIntoHierarchy() {
        // This not only refreshes the legal values for Targets, but it internally sets up property
        // change listeners to keep the "targetChooser" field in sync whenever a Target is added
        // or removed
        targetChooser.refreshLegalValues();
        super.elementInsertedIntoHierarchy();
    }

    public Integer getNumber() {
        return number.getValue();
    }

    public Target getTarget() {
        if(targetChooser.getSelectedValue() == null) {
            return null;
        }

        Object lValue = targetChooser.getSelectedValue();
        if(!(lValue instanceof Target)) {
            return null;
        }

        return (Target)targetChooser.getSelectedValue();
    }

    public void setTarget(String iID) {
        targetChooser.setValue(iID);
    }
```

```
public JwstInstrument getInstrument() {
    return instrumentChooser.get();
}

public void setInstrument(JwstInstrument iInstrument) {
    instrumentChooser.setValue(iInstrument);
}

public JwstTemplate<? extends JwstInstrument> getTemplate() {
    return template.get();
}

public void setTemplateChooser(JwstTemplateFactory iFactory) {
    templateChooser.set(iFactory);
}

private void ensureVisits () {
    removeAll();
    JwstTemplate<? extends JwstInstrument> lTemplate = getTemplate();
    if (lTemplate != null) {
        lTemplate.ensureVisits(this, getProperties());
    }
}

public JwstVisit ensureVisit (JwstTemplate iTemplate, TinaField[] iProperties) {
    JwstVisit lVisit = visitManager.get(iProperties);
    if (lVisit==null) {
        lVisit = getTemplate().createVisit(iProperties);
        visitManager.put(iTemplate, lVisit);
    }
    return lVisit;
}

@Override
public int setNumber (int iNumber) {
    number.setValue(iNumber);
    fFinalIndexNumber = fIndexNumber = iNumber;
    int lNextVisitNumber = 0;
    List<JwstVisit> lVisitsToNumber = new Vector<JwstVisit>();
    Iterator lIter = getChildren(JwstVisit.class).iterator();
    while (lIter.hasNext()) {
        JwstVisit lVisit = (JwstVisit)lIter.next();
        Integer lVisitNumber = lVisit.getNumber();
        if (lVisitNumber == null) {
            lVisitsToNumber.add(lVisit);
```

```
            } else {
                lNextVisitNumber = Math.max(lVisitNumber.intValue(), lNextVisitNumber);
            }
        }
        lIter = lVisitsToNumber.iterator();
        while (lIter.hasNext()) {
            //This code will number visits by adding 1000 times the Obs number, in case we want to go back to that.
//            lNextVisitNumber = lNextVisitNumber % 1000;
//            lNextVisitNumber = Math.min(999, ++lNextVisitNumber);
//          ((Visit)lIter.next()).setNumber(new Integer(lNextVisitNumber + 1000 * getNumberAsInt()));
            ((JwstVisit)lIter.next()).setNumber(new Integer(++lNextVisitNumber));
        }
        return ++iNumber;
    }

    @Override
    public Icon getIcon () {
        return ICON;
    }

    @Override
    public String getTypeName() {
        return TYPE_NAME;
    }

    public Element getDomElement() {
        // TODO Auto-generated method stub
        return null;
    }

    public void updateVisitNumbers() {
//        for(Visit lVisit : getChildren(Visit.class)) {
//            //lVisit.setNumber(lVisit.getNumberAsInt() % 1000 + 1000 * getNumberAsInt());
//            lVisit.setNumber(lVisit.getNumberAsInt());
//        }
    }

    @Override
    public String toString() {
        return "Observation "+getNumber();
    }

    @Override
    public JwstObservation clone() {
        JwstObservation lNewObs = new JwstObservation();
        JaxbObservationType lJaxbCopy = JwstProposalFile.convertToJaxb(this);
```

```
        JwstProposalFile.convertToDm(lJaxbCopy,lNewObs);
        return lNewObs;
    }

    @Override
    public void setTinaDocument(TinaDocument document) {
        super.setTinaDocument(document);
        obsPattern.refreshLegalValues();
    }
}
```

```java
package edu.stsci.jwst.apt.model.template;

import java.util.List;


public enum JwstTemplateFactory {
    MIRI_IMAGING("MIRI Imaging", MiriInstrument.getInstance(), MiriImagingTemplate.class),
    MIRI_LRS("MIRI Low Resolution Spectroscopy", MiriInstrument.getInstance(), MiriLrsTemplate.class),
    MIRI_MRS("MIRI Medium Resolution Spectroscopy", MiriInstrument.getInstance(), MiriMrsTemplate.class),
    MIRI_CORON("MIRI Coronagraphic Imaging", MiriInstrument.getInstance(), MiriCoronImagingTemplate.class),
    MIRI_DARK("MIRI Dark", MiriInstrument.getInstance(), MiriDarkTemplate.class),
    MIRI_IMG_FLAT("MIRI Imaging Flat", MiriInstrument.getInstance(), MiriImagingFlatTemplate.class),
    MIRI_MRS_FLAT("MIRI MRS Flat", MiriInstrument.getInstance(), MiriMrsFlatTemplate.class),
    MIRI_ANNEAL("MIRI Anneal", MiriInstrument.getInstance(), MiriAnnealTemplate.class);

    private String templateName;
    private JwstInstrument instrument;
    private Class<?> templateClass;

    JwstTemplateFactory(String iTemplateName, JwstInstrument iInstrument, Class<?> iTemplateClass){
        this.templateName = iTemplateName;
        this.instrument = iInstrument;
        this.templateClass = iTemplateClass;
    }

    public String getTemplateName() {return templateName;}

    public JwstInstrument getInstrument() {return instrument;}

    public Class<?> getTemplateClass() {return templateClass;}

    public JwstTemplate<? extends JwstInstrument> makeInstance () {
        try {
            return (JwstTemplate<? extends JwstInstrument>)templateClass.newInstance();
        } catch (InstantiationException e) {
            //TODO Handle this exception
            return null;
        } catch (IllegalAccessException e) {
            //TODO Handle this exception
            return null;
        }
    }

    // This method takes an instance of a JwstTemplate and finds the corresponding
    // factory method
    @Deprecated //tell me if this is used anywhere.  It shouldn't be anymore.
```

```
    }
}
```

```java
package edu.stsci.jwst.apt.model.template;

import org.jdom.Element;

public abstract class JwstTemplate<T extends JwstInstrument> extends AbstractTinaDocumentElement {

    public final T instrument;

    public JwstTemplate(T iInstrument) {
        instrument = iInstrument;
        Propagator.completeInitialization(this, JwstTemplate.class);
    }

    public abstract TinaField[] getAttributes();

    @Override
    public String getTypeName() {
        // TODO Auto-generated method stub
        return null;
    }

    public Element getDomElement() {
        throw new UnsupportedOperationException("DOM is not supported for JWST.");
    }

    /*
     * Returns the Instrument used by this Template
     */
    public T getInstrument() {
        return instrument;
    }

    public JwstVisit createVisit (TinaField[] iProperties) {
        //Need to decipher iProperties to get the parameters that should be set on the visit
        //For now, a blank visit is ok.
        return new JwstVisit();
    }

    public void ensureVisits (JwstObservation iObs, TinaField[] iProperties) {
        JwstVisit lVisit = iObs.ensureVisit(this, iProperties);
        iObs.add(lVisit, true);
        lVisit.setEditable(false);
    }
}
```

```java
package edu.stsci.jwst.apt.model.instrument;

import java.util.List;


/**
 * @author Rob Douglas
 *
 */
public abstract class JwstInstrument {

    protected String name;

    public abstract double computeOverheadTime ();

    @Override
    public String toString() {
        return getName();
    }

    public String getName() {
        return name;
    }

    public static List<JwstInstrument> getInstruments () {
        List<JwstInstrument> lInstruments = new Vector<JwstInstrument>();
        lInstruments.add(MiriInstrument.getInstance());
        lInstruments.add(NirCamInstrument.getInstance());
        lInstruments.add(NirSpecInstrument.getInstance());
        lInstruments.add(TfiInstrument.getInstance());
        return lInstruments;
    }
}
```

```java
package edu.stsci.jwst.apt.model.instrument;

import java.util.Arrays;

/**
 * @author Rob Douglas
 *
 */
public class MiriInstrument extends JwstInstrument {

    private static final MiriInstrument MIRI_INSTRUMENT = new MiriInstrument();

    public static final String DETECTOR = "Detector";
    public static final String FLAT_SUITE = "Flat Suite";
    public static final String WAVELENGTH = "Wavelength";
    public static final String REQUESTED_EXPOSURE_TIME = "Requested ExpTime";
    public static final String NUMBER_OF_EXPS = "No. of Exposures";
    public static final String NUMBER_OF_GROUPS = "No. of Groups";
    public static final String NUMBER_OF_INTS = "No. of Integrations";
    public static final String CALCULATED_EXP_TIME = "Calculated Exp Time";
    public static final String SUBARRAY = "Subarray";
    public static final String READOUT_PATTERN = "Readout Pattern";
    public static final String OBJECT_TYPE = "Object Type";
    public static final String FILTERS = "Filters";
    public static final String FILTER = "Filter";
    public static final String MASK = "Mask";
    public static final String CORON_FILTER = "Coron Mask/Filter";

    //Singleton
    private MiriInstrument () {
        name = "MIRI";
    }

    public static MiriInstrument getInstance() {
        return MIRI_INSTRUMENT;
    }

    @Override
    public double computeOverheadTime () {
        // TODO Auto-generated method stub
        return 0;
    }

    public static CosiConstrainedSelection<MiriDetector> makeDetectorField (JwstTemplate<? extends JwstInstrument> iIU) {
        Collection<MiriDetector> lDetectors = Arrays.asList(MiriDetector.values());
        return new CosiConstrainedSelection<MiriDetector>(iIU,DETECTOR, lDetectors, true);
```

```
    }

    public static CosiConstrainedSelection<MiriSubarray> makeSubarrayField (JwstTemplate<? extends JwstInstrument> iIU) {
        Collection<MiriSubarray> lSubarrays = Arrays.asList(MiriSubarray.values());
        return new CosiConstrainedSelection<MiriSubarray>(iIU,SUBARRAY, lSubarrays, true);
    }

    public static CosiConstrainedSelection<MiriReadPattern> makeReadoutPatternField (JwstTemplate<? extends JwstInstrument>
iIU) {
        Collection<MiriReadPattern> lReadPatterns = Arrays.asList(MiriReadPattern.values());
        return new CosiConstrainedSelection<MiriReadPattern>(iIU,READOUT_PATTERN, lReadPatterns, true);
    }

    public static CosiConstrainedSelection<MiriObjectType> makeObjectTypeField (JwstTemplate<? extends JwstInstrument> iIU) {
        Collection<MiriObjectType> lObjectTypes = Arrays.asList(MiriObjectType.values());
        return new CosiConstrainedSelection<MiriObjectType>(iIU, OBJECT_TYPE, lObjectTypes, true);
    }

    public static MiriImagingExposureTable makeMiriImagingExposureTableField (JwstTemplate<? extends JwstInstrument> iIU) {
        return new MiriImagingExposureTable(iIU, FILTERS);
    }

    public static MiriCoronExposureTable makeMiriCoronExposureTableField (JwstTemplate<? extends JwstInstrument> iIU) {
        return new MiriCoronExposureTable(iIU, FILTERS);
    }

    public static MiriDarkExposureTable makeMiriDarkExposureTableField (JwstTemplate<? extends JwstInstrument> iIU) {
        return new MiriDarkExposureTable(iIU, FILTERS);
    }

    public static MiriExposure makeMiriMrsExposure (MiriMrsTemplate iIU) {
        return new MiriMrsExposure(iIU);
    }

    public static MiriExposure makeMiriLrsExposure (MiriLrsTemplate iIU) {
        return new MiriLrsExposure(iIU);
    }

    public static CosiConstrainedSelection<MiriFlatSuite> makeFlatSuiteField (JwstTemplate<? extends JwstInstrument> iIU) {
        Collection<MiriFlatSuite> lFlats = Arrays.asList(MiriFlatSuite.values());
        return new CosiConstrainedSelection<MiriFlatSuite>(iIU,FLAT_SUITE, lFlats, true);
    }

    public static CosiConstrainedSelection<MiriWavelength> makeWavelengthField (JwstTemplate<? extends JwstInstrument> iIU) {
        Collection<MiriWavelength> lWaves = Arrays.asList(MiriWavelength.values());
        return new CosiConstrainedSelection<MiriWavelength>(iIU,WAVELENGTH, lWaves, true);
```

```
    }

    public static CosiConstrainedDouble makeRequestedExpTimeField (JwstTemplate<? extends JwstInstrument> iIU) {
        return new CosiConstrainedDouble(iIU, REQUESTED_EXPOSURE_TIME, true, 0.0, Double.MAX_VALUE);
    }

    public static CosiConstrainedInt makeNumberOfExpsField (JwstTemplate<? extends JwstInstrument> iIU) {
        CosiConstrainedInt lNumExps = new CosiConstrainedInt(iIU, NUMBER_OF_EXPS, true, 1, Integer.MAX_VALUE);
        lNumExps.set(1);
        return lNumExps;
    }

    public static CosiConstrainedInt makeNumberOfGroupsField (JwstTemplate<? extends JwstInstrument> iIU) {
        CosiConstrainedInt lNumGroups = new CosiConstrainedInt(iIU, NUMBER_OF_GROUPS, true, 1, Integer.MAX_VALUE);
        lNumGroups.set(0);
        return lNumGroups;
    }

    public static CosiConstrainedInt makeNumberOfIntsField (JwstTemplate<? extends JwstInstrument> iIU) {
        CosiConstrainedInt lNumInts = new CosiConstrainedInt(iIU, NUMBER_OF_INTS, true, 1, Integer.MAX_VALUE);
        lNumInts.set(0);
        return lNumInts;
    }

    public static CosiConstrainedSelection<MiriFilter> makeFilterField (JwstTemplate<? extends JwstInstrument> iIU) {
        return makeFilterField(iIU, FILTER);
    }

    public static CosiConstrainedSelection<MiriFilter> makeFilterField (JwstTemplate<? extends JwstInstrument> iIU, String
iName) {
        Collection<MiriFilter> lFilters = Arrays.asList(MiriFilter.values());
        return new CosiConstrainedSelection<MiriFilter>(iIU, iName, lFilters, true);
    }

    public static CosiConstrainedSelection<MiriCoronFilter> makeCoronFilterField (JwstTemplate<? extends JwstInstrument>
iIU) {
        Collection<MiriCoronFilter> lCoronFilters = Arrays.asList(MiriCoronFilter.values());
        return new CosiConstrainedSelection<MiriCoronFilter>(iIU, CORON_FILTER, lCoronFilters, true);
    }

    public static CosiConstrainedSelection<MiriMask> makeMaskField (JwstTemplate<? extends JwstInstrument> iIU) {
        Collection<MiriMask> lMasks = Arrays.asList(MiriMask.values());
        return new CosiConstrainedSelection<MiriMask>(iIU, MASK, lMasks, true);
    }

    public static CosiConstrainedDouble makeCalculatedExpTimeField (JwstTemplate<? extends JwstInstrument> iIU) {
```

```
        CosiConstrainedDouble lField =
            new CosiConstrainedDouble(iIU, MiriInstrument.CALCULATED_EXP_TIME, true, 1, Double.MAX_VALUE);
        lField.setEditable(false);
        return lField;
    }


    //Detector Definitions
    //TODO find the real names of these
    public enum MiriDetector {
        IMAGER,
        MRS,
        ALL
    }

    //Detector Readout Regions, aka Subarrays
    public enum MiriSubarray {
        FULL,
        BRIGHTSKY
    }

    //Detector Readout Patterns
    public enum MiriReadPattern {
        SLOW (new Integer[] {1,    2,    3,    4,     5,     6,     7,     8,     9,     10},
              new Double[]  {30.0, 60.0, 90.0, 120.0, 180.0, 240.0, 300.0, 600.0, 900.0, 1200.0}),

        //TODO: I arbitrarily chose this mode (as opposed to the FASTMode_Short below).  Need to confirm this assumption.
        FAST (new Integer[] {1,   2,   3,   4,    5,    6,    7,    8,    9,    10,   11,   12,   13},
              new Double[]  {3.0, 6.0, 9.0, 12.0, 15.0, 18.0, 21.0, 24.0, 27.0, 30.0, 60.0, 90.0, 120.0});

//        FASTMode_Short (new Integer[] {1},
//                        new Double[]  {3.0});
//
//        SUBARRAY1_SUBARRAY4 (new Integer[] {1,    2,    3,    4,    5,    6,    7,   8},
//                             new Double[]  {0.16, 0.33, 0.49, 0.66, 0.82, 0.98, 1.2, 1.3}),
//
//        SUBARRAY5 (new Integer[] {1,    2,   3,   4},
//                   new Double[]  {0.65, 1.3, 2.0, 4.0});

        private Integer[] fValidSamples;
        private HashMap<Integer, Double> fSampleTimeMap = new HashMap<Integer, Double>();

        private MiriReadPattern(Integer[] validSamples, Double[] sampleTimes) {
            fValidSamples = validSamples;
            for(int i=0; i<validSamples.length; i++) {
                fSampleTimeMap.put(validSamples[i], sampleTimes[i]);
```

```
        }
    }

    public Integer[] getValidSamples() {
        return fValidSamples;
    }

    public Double[] getValidIntegrationTimes() {
        Double[] lTimes = new Double[fValidSamples.length];
        for(int i=0; i<fValidSamples.length; i++) {
            lTimes[i] = fSampleTimeMap.get(fValidSamples[i]);
        }

        return lTimes;
    }

    // Given the key, return the value
    public Double getSampleTime(Integer iNumberOfSamples) {
        return fSampleTimeMap.get(iNumberOfSamples);
    }

    // Given the value, return the key
    public Integer getNumberOfSamples(Double iSampleTime) {
        for(Integer i : fSampleTimeMap.keySet()) {
            if(fSampleTimeMap.get(i).equals(iSampleTime)) {
                return i;
            }
        }
        return null;
    }
}

public enum MiriObjectType {
    FAINT,
    BRIGHT
}

public enum MiriFilter {
    F560W(5.6, 1.2, 1000),
    F770W(7.7, 2.2, 1000),
    F1000W(10.0, 2.0, 1000),
    F1065C(10.7, 0.5, 1000),
    F1130W(11.3, 0.7, 1000),
    F1140C(11.4, 0.6, 1000),
    F1280W(12.8, 2.4, 1000),
    F1500W(15.0, 3.0, 1000),
```

```
    F1550C(15.5, 0.8, 1000),
    F1800W(18.0, 3.0, 1000),
    F2100W(21.0, 5.0, 1000),
    F2300C(23.0, 4.6, 1000),
    F2550W(25.5, 4.0, 1000),
    F2550WR(25.5, 4.0, 1000),
    P750L(7.5, 0, 1000),
    FND(13.0, 10.0, 1000);

    private double centerWave;
    private double bandpass;
    private int maxInts;

    MiriFilter(double iCenterWave, double iBandpass, int iMaxInts) {
        centerWave = iCenterWave;
        bandpass = iBandpass;
        maxInts = iMaxInts;
    }

    public double getCenterWave() {
        return centerWave;
    }

    public double getBandpass() {
        return bandpass;
    }

    public int getMaxInts() {
        return maxInts;
    }
}

public enum MiriMask {
    _4QPM("4QPM"),
    LYOT("LYOT");

    MiriMask (String iName) {
        name = iName;
    }
    private String name;

    @Override
    public String toString() {
        return name;
    }
}
```

```
public enum MiriCoronFilter {
    _4QPMF1065C("4QPM/F1065C", MiriMask._4QPM, MiriFilter.F1065C),
    _4QPMF1140C("4QPM/F1140C", MiriMask._4QPM, MiriFilter.F1140C),
    _4QPMF1550C("4QPM/F1550C", MiriMask._4QPM, MiriFilter.F1550C),
    LYOTF2300C("LYOT/F2300C", MiriMask.LYOT, MiriFilter.F2300C);

    MiriCoronFilter(String iComboName, MiriMask iCoron, MiriFilter iFilter){
        name=iComboName;
        mask = iCoron;
        filter = iFilter;
    }

    private String name;
    private MiriMask mask;
    private MiriFilter filter;

    public String getComboName() {
        return name;
    }
    public MiriMask getMask() {
        return mask;
    }
    public MiriFilter getFilter() {
        return filter;
    }
    @Override
    public String toString() {
        return name;
    }
}

public enum MiriFlatSuite {
    ALL,
    PRINCIPAL,
    ONE
}

public enum MiriWavelength {
    ALL("default", 1000),
    SHORT("4.87-5.82, 7.45-8.90, 11.47-13.67, 17.54-21.10", 100),
    MEDIUM("5.62-6.73, 8.61-10.28, 13.25-15.80, 20.44-24.72", 200),
    LONG("6.49-7.76, 9.94-11.87, 15.30-18.24, 23.84-28.82", 300);

    private String ranges;
    private int maxInts;
```

```
        MiriWavelength (String iRanges, int iMaxInts) {
            ranges = iRanges;
            maxInts = iMaxInts;
        }

        public String getRanges() {
            return ranges;
        }

        public int getMaxInts() {
            return maxInts;
        }
    }
}
```

```java
package edu.stsci.jwst.apt.model.template.miri;

import java.util.List;

public class MiriImagingTemplate extends JwstTemplate<MiriInstrument> {

    public MiriImagingTemplate () {
        super(MiriInstrument.getInstance());
        Propagator.completeInitialization(this, MiriImagingTemplate.class);
    }

    private CosiConstrainedSelection<MiriSubarray> subarray = MiriInstrument.makeSubarrayField(this);
    private CosiConstrainedSelection<MiriObjectType> objectType = MiriInstrument.makeObjectTypeField(this);
    private MiriImagingExposureTable expTable = MiriInstrument.makeMiriImagingExposureTableField(this);

    private TinaField<?>[] attributes = new TinaField[] {subarray, objectType, expTable};

    @Override
    public TinaField<?>[] getAttributes() {
        return attributes;
    }

    public void setObjectType(MiriObjectType iType) {
        objectType.setValue(iType);
    }

    public void setSubarray(MiriSubarray iSubarray) {
        subarray.setValue(iSubarray);
    }

    public MiriSubarray getSubarray() {
        return subarray.getValue();
    }

    public CosiConstrainedSelection<MiriSubarray> getSubarrayField() {
        return subarray;
    }

    public MiriObjectType getObjectType() {
        return objectType.getValue();
    }

    public CosiConstrainedSelection<MiriObjectType> getObjectTypeField() {
        return objectType;
    }
```

```
    public List<MiriImagingExposureTableRow> getExposures() {
        return expTable.getValue();
    }

    public void newExposure() {
        expTable.addField();
    }

    public void addExposure(MiriImagingExposureTableRow iRow) {
        expTable.addField(iRow);
    }

    public MiriImagingExposureTable getExposureTable () {
        return expTable;
    }

    @Override
    public MiriImagingVisit createVisit(TinaField[] iProperties) {
        return new MiriImagingVisit (iProperties);
    }

    private class MiriImagingVisit extends JwstVisit {
        public MiriImagingVisit(TinaField[] iProperties) {
            setProperties(iProperties);
        }
    }
}
```

```java
package edu.stsci.jwst.apt.model.template.miri;

import edu.stsci.CoSI.Constraint;

public class MiriLrsTemplate extends JwstTemplate<MiriInstrument> {
    private MiriTargetAcq tacq = new MiriTargetAcq(this);
    private MiriLrsExposure exposure = new MiriLrsExposure(this);

    public MiriLrsTemplate () {
        super(MiriInstrument.getInstance());
        readoutPattern.setEditable(false);
        numberOfGroups.setEditable(false);
        numberOfInts.setEditable(false);
        add(tacq, false);
        Propagator.completeInitialization(this, MiriLrsTemplate.class);
    }

    private final CosiConstrainedSelection<MiriObjectType> objectType = MiriInstrument.makeObjectTypeField(this);
    private final CosiConstrainedDouble reqExpTime = MiriInstrument.makeRequestedExpTimeField(this);
    private final CosiConstrainedInt numberOfGroups = MiriInstrument.makeNumberOfGroupsField(this);
    private final CosiConstrainedInt numberOfInts = MiriInstrument.makeNumberOfIntsField(this);
    private final CosiConstrainedSelection<MiriReadPattern> readoutPattern = MiriInstrument.makeReadoutPatternField(this);

    private final CosiConstrainedDouble calculatedExpTime = MiriInstrument.makeCalculatedExpTimeField(this);
    {
        Propagator.addDelayedConstraint(new Constraint(this, "Calculated Exp Time") {
            public void run() {
                setReadoutPattern(exposure.getReadoutPattern());
                setNumberOfGroups(exposure.getNumberOfGroups());
                setNumberOfIntegrations(exposure.getNumberOfIntegrations());
                calculatedExpTime.set(exposure.getCalculatedExpTime());
            }
        });
    }
    private final TinaFieldGroup expTimeGroup = new TinaFieldGroup(this, "Exposure Time");{
        reqExpTime.setTinaFieldGroup(expTimeGroup);
        expTimeGroup.setLabel(reqExpTime, MiriInstrument.REQUESTED_EXPOSURE_TIME);
        readoutPattern.setTinaFieldGroup(expTimeGroup);
        expTimeGroup.setLabel(readoutPattern,MiriInstrument.READOUT_PATTERN);
        numberOfGroups.setTinaFieldGroup(expTimeGroup);
        expTimeGroup.setLabel(numberOfGroups,MiriInstrument.NUMBER_OF_GROUPS);
        numberOfInts.setTinaFieldGroup(expTimeGroup);
        expTimeGroup.setLabel(numberOfInts,MiriInstrument.NUMBER_OF_INTS);
        calculatedExpTime.setTinaFieldGroup(expTimeGroup);
        expTimeGroup.setLabel(calculatedExpTime,MiriInstrument.CALCULATED_EXP_TIME);
    }
```

```
private TinaField<?>[] attributes = new TinaField[] {objectType, reqExpTime,
        numberOfGroups, numberOfInts, calculatedExpTime};{
            attributes = (TinaField<?>[])ArrayUtils.addArrays(tacq.getAttributes(), attributes);
        }

@Override
public TinaField<?>[] getAttributes() {
    return attributes;
}

public MiriObjectType getObjectType() {
    return objectType.get();
}

public void setObjectType (MiriObjectType iObjectType) {
    objectType.set(iObjectType);
}

public Double getRequestedExpTime() {
    return reqExpTime.get();
}

public void setRequestedExpTime (Double iReqExpTime) {
    reqExpTime.set(iReqExpTime);
}

public Target getAcqTargetChooser() {
    return tacq.getAcqTargetChooser();
}

public void setTarget(String iID) {
    tacq.setTarget(iID);
}

public MiriFilter getAcqFilter() {
    return tacq.getAcqFilter();
}

public void setAcqFilter (MiriFilter iFilter) {
    tacq.setAcqFilter(iFilter);
}

public String getAcqFlux() {
    return tacq.getAcqFlux();
```

```
    }

    public void setAcqFlux (String iFlux) {
        tacq.setAcqFlux(iFlux);
    }

    public Integer getNumberOfGroups () {
        return numberOfGroups.get();
    }

    public void setNumberOfGroups (Integer iGroups) {
        numberOfGroups.set(iGroups);
    }

    public Integer getNumberOfIntegrations () {
        return numberOfInts.get();
    }

    public void setNumberOfIntegrations (Integer iInts) {
        numberOfInts.set(iInts);
    }

    public MiriReadPattern getReadoutPattern () {
        return readoutPattern.get();
    }

    public void setReadoutPattern (MiriReadPattern iRead) {
        readoutPattern.set(iRead);
    }

    public CosiConstrainedSelection<MiriObjectType> getObjectTypeField() {
        return objectType;
    }
}
```

```
package edu.stsci.jwst.apt.model.template.miri;

import edu.stsci.CoSI.Constraint;

public class MiriMrsTemplate extends JwstTemplate<MiriInstrument> {

    private MiriTargetAcq tacq = new MiriTargetAcq(this);
    private MiriMrsExposure exposure = new MiriMrsExposure(this);

    public MiriMrsTemplate () {
        super(MiriInstrument.getInstance());
        readoutPattern.setEditable(false);
        numberOfGroups.setEditable(false);
        numberOfInts.setEditable(false);
        add(tacq, false);
        Propagator.completeInitialization(this, MiriMrsTemplate.class);
    }

    private final CosiConstrainedSelection<MiriObjectType> objectType = MiriInstrument.makeObjectTypeField(this);
    private final CosiConstrainedSelection<MiriWavelength> wavelength = MiriInstrument.makeWavelengthField(this);
    private final CosiConstrainedDouble reqExpTime = MiriInstrument.makeRequestedExpTimeField(this);
    private final CosiConstrainedInt numberOfGroups = MiriInstrument.makeNumberOfGroupsField(this);
    private final CosiConstrainedInt numberOfInts = MiriInstrument.makeNumberOfIntsField(this);
    private final CosiConstrainedSelection<MiriReadPattern> readoutPattern = MiriInstrument.makeReadoutPatternField(this);

    private final CosiConstrainedDouble calculatedExpTime = MiriInstrument.makeCalculatedExpTimeField(this);
    {
        Propagator.addDelayedConstraint(new Constraint(this, "Calculated Exp Time") {
            public void run() {
                setReadoutPattern(exposure.getReadoutPattern());
                setNumberOfGroups(exposure.getNumberOfGroups());
                setNumberOfIntegrations(exposure.getNumberOfIntegrations());
                calculatedExpTime.set(exposure.getCalculatedExpTime());
            }
        });
    }
    private final TinaFieldGroup expTimeGroup = new TinaFieldGroup(this, "Exposure Time");{
        reqExpTime.setTinaFieldGroup(expTimeGroup);
        expTimeGroup.setLabel(reqExpTime, MiriInstrument.REQUESTED_EXPOSURE_TIME);
        readoutPattern.setTinaFieldGroup(expTimeGroup);
        expTimeGroup.setLabel(readoutPattern,MiriInstrument.READOUT_PATTERN);
        numberOfGroups.setTinaFieldGroup(expTimeGroup);
        expTimeGroup.setLabel(numberOfGroups,MiriInstrument.NUMBER_OF_GROUPS);
        numberOfInts.setTinaFieldGroup(expTimeGroup);
        expTimeGroup.setLabel(numberOfInts,MiriInstrument.NUMBER_OF_INTS);
        calculatedExpTime.setTinaFieldGroup(expTimeGroup);
```

```
        expTimeGroup.setLabel(calculatedExpTime,MiriInstrument.CALCULATED_EXP_TIME);
    }

    private TinaField<?>[] attributes = new TinaField[] {objectType, wavelength, reqExpTime,
            numberOfGroups, numberOfInts, calculatedExpTime}; {
                attributes = (TinaField<?>[])ArrayUtils.addArrays(tacq.getAttributes(), attributes);
            }

    @Override
    public TinaField<?>[] getAttributes() {
        return attributes;
    }

    public MiriObjectType getObjectType() {
        return objectType.get();
    }

    public void setObjectType (MiriObjectType iObjectType) {
        objectType.set(iObjectType);
    }

    public MiriWavelength getWavelength() {
        return wavelength.get();
    }

    public void setWavelength (MiriWavelength iWavelength) {
        wavelength.set(iWavelength);
    }

    public Double getRequestedExpTime() {
        return reqExpTime.get();
    }

    public void setRequestedExpTime (Double iReqExpTime) {
        reqExpTime.set(iReqExpTime);
    }

    public Target getAcqTargetChooser() {
        return tacq.getAcqTargetChooser();
    }

    public void setTarget(String iID) {
        tacq.setTarget(iID);
    }

    public MiriFilter getAcqFilter() {
```

```
        return tacq.getAcqFilter();
    }

    public void setAcqFilter (MiriFilter iFilter) {
        tacq.setAcqFilter(iFilter);
    }

    public String getAcqFlux() {
        return tacq.getAcqFlux();
    }

    public void setAcqFlux (String iFlux) {
        tacq.setAcqFlux(iFlux);
    }

    public Integer getNumberOfGroups () {
        return numberOfGroups.get();
    }

    public void setNumberOfGroups (Integer iGroups) {
        numberOfGroups.set(iGroups);
    }

    public Integer getNumberOfIntegrations () {
        return numberOfInts.get();
    }

    public void setNumberOfIntegrations (Integer iInts) {
        numberOfInts.set(iInts);
    }

    public MiriReadPattern getReadoutPattern () {
        return readoutPattern.get();
    }

    public void setReadoutPattern (MiriReadPattern iRead) {
        readoutPattern.set(iRead);
    }

    public CosiConstrainedSelection<MiriObjectType> getObjectTypeField() {
        return objectType;
    }
}
```

```java
package edu.stsci.jwst.apt.model.template.miri;

import java.util.List;

public class MiriCoronImagingTemplate extends JwstTemplate<MiriInstrument> {
    public MiriCoronImagingTemplate () {
        super(MiriInstrument.getInstance());
        add(tacq, false);
    }

    private MiriTargetAcq tacq = new MiriTargetAcq(this);

    private CosiConstrainedSelection<MiriObjectType> objectType = MiriInstrument.makeObjectTypeField(this);
    private MiriCoronExposureTable expTable = MiriInstrument.makeMiriCoronExposureTableField(this);

    private TinaField<?>[] attributes = new TinaField[] {objectType, expTable}; {
        attributes = (TinaField<?>[])ArrayUtils.addArrays(tacq.getAttributes(), attributes);
    }

    @Override
    public TinaField<?>[] getAttributes() {
        return attributes;
    }

    public Target getAcqTargetChooser() {
        return tacq.getAcqTargetChooser();
    }

    public void setTarget(String iID) {
        tacq.setTarget(iID);
    }

    public MiriFilter getAcqFilter() {
        return tacq.getAcqFilter();
    }

    public void setAcqFilter (MiriFilter iFilter) {
        tacq.setAcqFilter(iFilter);
    }

    public String getAcqFlux() {
        return tacq.getAcqFlux();
    }

    public void setAcqFlux (String iFlux) {
        tacq.setAcqFlux(iFlux);
```

```
}

public void setObjectType(MiriObjectType iType) {
    objectType.setValue(iType);
}

public MiriObjectType getObjectType() {
    return objectType.getValue();
}

public CosiConstrainedSelection<MiriObjectType> getObjectTypeField() {
    return objectType;
}

public List<MiriCoronExposureTableRow> getFilters() {
    return expTable.getValue();
}

public void newFilter() {
    expTable.addField();
}

public void addFilter(MiriCoronExposureTableRow iRow) {
    expTable.addField(iRow);
}

public MiriCoronExposureTable getFilterTable () {
    return expTable;
}

@Override
public void ensureVisits (JwstObservation iObs, TinaField[] iProperties) {
    //this is where I would loop through the Coron list to get the
    //subset of corons.
    iObs.add(iObs.ensureVisit(this, iProperties), true);
    iObs.add(iObs.ensureVisit(this, iProperties), true);
}

@Override
public MiriCoronImagingVisit createVisit(TinaField[] iProperties) {
    return new MiriCoronImagingVisit (iProperties);
}

private class MiriCoronImagingVisit extends JwstVisit {
    public MiriCoronImagingVisit(TinaField[] iProperties) {
        setProperties(iProperties);
```

```
        }
    }

}
```

```java
package edu.stsci.jwst.apt.model.template.miri;

import java.util.List;

public class MiriDarkTemplate extends JwstTemplate<MiriInstrument> {
    public MiriDarkTemplate () {
        super(MiriInstrument.getInstance());
    }

    private CosiConstrainedSelection<MiriDetector> detector = MiriInstrument.makeDetectorField(this);{
        List<MiriDetector> lLegalsValues = detector.getLegalValues();
        lLegalsValues.remove(MiriDetector.ALL);
        detector.setLegalValues(lLegalsValues);
    }
    private MiriDarkExposureTable expTable = MiriInstrument.makeMiriDarkExposureTableField(this);

    private TinaField<?>[] attributes = new TinaField[] {detector, expTable};

    @Override
    public TinaField<?>[] getAttributes() {
        return attributes;
    }

    public MiriDetector getDetector() {
        return detector.get();
    }

    public void setDetector(MiriDetector iDetector) {
        this.detector.set(iDetector);
    }

    public List<MiriDarkExposureTableRow> getExposures() {
        return expTable.getValue();
    }

    public void newExposure() {
        expTable.addField();
    }

    public void addExposure(MiriDarkExposureTableRow iRow) {
        expTable.addField(iRow);
    }

    public MiriDarkExposureTable getExposureTable () {
        return expTable;
    }
```

```
}
```

```
package edu.stsci.jwst.apt.model.template.miri;

import java.util.Arrays;

public class MiriImagingFlatTemplate extends JwstTemplate<MiriInstrument> {
    public MiriImagingFlatTemplate () {
        super(MiriInstrument.getInstance());
    }

    private CosiConstrainedSelection<MiriFlatSuite> flatSuite = MiriInstrument.makeFlatSuiteField(this);
    private CosiConstrainedSelection<MiriFilter> filter = MiriInstrument.makeFilterField(this);
    private CosiConstrainedInt numInts = MiriInstrument.makeNumberOfIntsField(this);

    private TinaField<?>[] attributes = new TinaField[] {flatSuite, filter, numInts};

    @Override
    public TinaField<?>[] getAttributes() {
        return attributes;
    }

    {
        Propagator.addConstraint(new Constraint(this, "Flat Properties") {
            public void run () {
                if (flatSuite.isSpecified()) {
                    Collection<MiriFilter> lLegals = new Vector<MiriFilter>();
                    switch (flatSuite.get()) {
                        case ONE:
                            lLegals = Arrays.asList(MiriFilter.values());
                            filter.setRequired(true);
                            break;
                        default:
                            filter.setRequired(false);
                            filter.set(null);
                            break;
                    }
                    filter.setLegalValues(lLegals);
                } else {
                    filter.setRequired(false);
                    filter.set(null);
                }
            }
        }
        );
        Propagator.addConstraint(new Constraint(this, "Number Of Ints Limits") {
            public void run () {
                if (filter.isSpecified()) {
```

```
                    numInts.setRequired(true );
                    numInts.setMax(filter.get().getMaxInts());
                } else {
                    numInts.setRequired(false);
                    numInts.set(null);
                }
            }
        }
        );
    }

    public MiriFlatSuite getFlatSuite() {
        return flatSuite.get();
    }

    public void setFlatSuite(MiriFlatSuite iFlatSuite) {
        this.flatSuite.set(iFlatSuite);
    }

    public MiriFilter getFilter() {
        return filter.get();
    }

    public void setFilter(MiriFilter iFilter) {
        this.filter.set(iFilter);
    }

    public int getNumInts() {
        return numInts.get();
    }

    public void setNumInts(int iNumInts) {
        this.numInts.set(iNumInts);
    }
}
```

```
package edu.stsci.jwst.apt.model.template.miri;

import java.util.Arrays;

public class MiriMrsFlatTemplate extends JwstTemplate<MiriInstrument> {
    public MiriMrsFlatTemplate () {
        super(MiriInstrument.getInstance());
    }

    private CosiConstrainedSelection<MiriFlatSuite> flatSuite = MiriInstrument.makeFlatSuiteField(this);{
        List<MiriFlatSuite> lLegalsValues = flatSuite.getLegalValues();
        lLegalsValues.remove(MiriFlatSuite.PRINCIPAL);
        flatSuite.setLegalValues(lLegalsValues);
    }
    private CosiConstrainedSelection<MiriWavelength> wavelength = MiriInstrument.makeWavelengthField(this); {
        List<MiriWavelength> lWavelengths = wavelength.getLegalValues();
        lWavelengths.remove(MiriWavelength.ALL);
        wavelength.setLegalValues(lWavelengths);
    }
    private CosiConstrainedInt numberOfInts = MiriInstrument.makeNumberOfIntsField(this);

    private TinaField<?>[] attributes = new TinaField[] {flatSuite, wavelength, numberOfInts};

    @Override
    public TinaField<?>[] getAttributes() {
        return attributes;
    }

    {
        Propagator.addConstraint(new Constraint(this, "Flat Properties") {
            public void run () {
                if (flatSuite.isSpecified()) {
                    Collection<MiriWavelength> lLegals = new Vector<MiriWavelength>();
                    switch (flatSuite.get()) {
                        case ONE:
                            lLegals = Arrays.asList(MiriWavelength.values());
                            wavelength.setRequired(true);
                            break;
                        default:
                            wavelength.setRequired(false);
                            wavelength.set(null);
                            break;
                    }
                    wavelength.setLegalValues(lLegals);
                } else {
                    wavelength.setRequired(false);
```

```
                    wavelength.set(null);
                }
            }
        }
        );
        Propagator.addConstraint(new Constraint(this, "Number Of Ints Limits") {
            public void run () {
                if (wavelength.isSpecified()) {
                    numberOfInts.setRequired(true );
                    numberOfInts.setMax(wavelength.get().getMaxInts());
                } else {
                    numberOfInts.setRequired(false);
                    numberOfInts.set(null);
                }
            }
        }
        );
    }

    public MiriFlatSuite getFlatSuite() {
        return flatSuite.get();
    }
    public void setFlatSuite(MiriFlatSuite iSuite) {
        flatSuite.set(iSuite);
    }

    public MiriWavelength getWavelength() {
        return wavelength.get();
    }

    public void setWavelength(MiriWavelength iWavelength) {
        wavelength.set(iWavelength);
    }

    public Integer getNumberOfInts() {
        return numberOfInts.get();
    }

    public void setNumberOfInts(Integer iInt) {
        numberOfInts.set(iInt);
    }

}
```

```java
package edu.stsci.jwst.apt.model.template.miri;

import edu.stsci.jwst.apt.model.instrument.MiriInstrument;

public class MiriAnnealTemplate extends JwstTemplate<MiriInstrument> {
    public MiriAnnealTemplate () {
        super(MiriInstrument.getInstance());
    }

    private CosiConstrainedSelection<MiriDetector> detector = MiriInstrument.makeDetectorField(this);

    private TinaField<?>[] attributes = new TinaField[] {detector};

    @Override
    public TinaField<?>[] getAttributes() {
        return attributes;
    }

    public MiriDetector getDetector() {
        return detector.get();
    }

    public void setDetector(MiriDetector iDetector) {
        detector.set(iDetector);
    }

}
```

```
package edu.stsci.jwst.apt.model.template.miri;

import java.util.Arrays;

public class MiriTargetAcq extends AbstractTinaDocumentElement {
    private static final String ACQ_TARGET = "AcqTarget";
    private static final String ACQ_FILTER = "AcqFilter";
    private static final String ACQ_FLUX = "AcqFlux";
    private static final String TACQ_FIELDS = "Target ACQ";

    public static final MiriFilter[] acqFilters = new MiriFilter[]{
        MiriFilter.F560W,
        MiriFilter.F1000W,
        MiriFilter.F1500W,
        MiriFilter.FND};

    //TODO ROB Have to make the target chooser keep updated
    private final TargetChooser acqTargetChooser = new TargetChooser(null, ACQ_TARGET); {
        //User cannot change this - needs to be same as science target
        acqTargetChooser.setEditable(false);

        Propagator.addDelayedConstraint(new Constraint(this, "Coron TACQ Target") {
            public void run () {
                if (getParent()!=null) {
                    acqTargetChooser.setValue(((JwstObservation)getParent()).getTarget());
                }
            }
        });
    }
    private final CosiConstrainedSelection<MiriFilter> acqFilter = MiriInstrument.makeFilterField(null, ACQ_FILTER); {
        acqFilter.setLegalValues(Arrays.asList(acqFilters));
    }
    //This is a temporary field - until we can get Fluxes from the ACQ target directly.
    private final TinaCosiStringField acqFlux = new TinaCosiStringField(null, ACQ_FLUX, true);
    private final TinaFieldGroup tacqFields = new TinaFieldGroup(null, TACQ_FIELDS); {
        acqTargetChooser.setTinaFieldGroup(tacqFields);
        tacqFields.setLabel(acqTargetChooser, ACQ_TARGET);
        acqFilter.setTinaFieldGroup(tacqFields);
        tacqFields.setLabel(acqFilter, ACQ_FILTER);
        acqFlux.setTinaFieldGroup(tacqFields);
        tacqFields.setLabel(acqFlux, ACQ_FLUX);
    }
    private JwstTemplate template;

    public MiriTargetAcq(JwstTemplate iTemplate) {
        template = iTemplate;
```

```
    acqTargetChooser.setContainer(template);
    acqFilter.setContainer(template);
    acqFlux.setContainer(template);
    tacqFields.setContainer(template);
}

private TinaField<?>[] attributes = new TinaField[] {acqTargetChooser, acqFilter, acqFlux};

public TinaField<?>[] getAttributes() {
    return attributes;
}

@Override
public String getTypeName() {
    // TODO Auto-generated method stub
    return null;
}

public Element getDomElement() {
    // TODO Auto-generated method stub
    return null;
}

public Target getAcqTargetChooser() {
    if(acqTargetChooser.getSelectedValue() == null) {
        return null;
    }

    Object lValue = acqTargetChooser.getSelectedValue();
    if(!(lValue instanceof Target)) {
        return null;
    }

    return (Target)acqTargetChooser.getSelectedValue();
}

public void setTarget(String iID) {
    acqTargetChooser.setValue(iID);
}

public MiriFilter getAcqFilter() {
    return acqFilter.get();
}

public void setAcqFilter (MiriFilter iFilter) {
    acqFilter.set(iFilter);
```

```
    }

    public String getAcqFlux() {
        return acqFlux.get();
    }

    public void setAcqFlux (String iFlux) {
        acqFlux.set(iFlux);
    }
}
```

```java
package edu.stsci.jwst.apt.model.instrument;

import java.text.DecimalFormat;

public abstract class MiriExposureTable<T extends MiriExposureTableRow> extends AbstractTinaMultiField<T> {
    {
        fDisplayEditButton = false;
    }

    public MiriExposureTable (TinaDocumentElement iContainer, String iName) {
        super(iContainer, iName);
    }

    //TODO ROB Make getTotalExpTime() a Derived property at some point
    public double getTotalExpTime() {
        double lTotal = 0.0;
        for(T lRow : getValue()) {
            Double d = lRow.getCalculatedExpTime();
            if(d != null) {
                lTotal += d;
            }
        }
        // HACK: Correct the stupid Java roundoff error (e.g. don't return
        // something like 37.23000000000000001 when adding 34.11 and 3.12)
        DecimalFormat df = new DecimalFormat("0.00");
        lTotal = Double.parseDouble(df.format(lTotal));
        return lTotal;
    }

    public abstract int getColumnCount ();

    public abstract Class<?> getColumnClass (int iColumn);

    public abstract void setValueAt (Object iValue, int iRow, int iColumn);

    public abstract String getColumnName(int iColumn);

    public abstract Object getValueAt(int rowIndex, int iColumn);

    public void configureDefaultEditors (JTable iTable) {
        iTable.setDefaultEditor(CosiConstrainedInt.class, new DefaultTinaCosiFieldEditor());
        iTable.setDefaultEditor(CosiConstrainedDouble.class, new DefaultTinaCosiFieldEditor());
        iTable.setDefaultEditor(CosiConstrainedSelection.class, new CosiConstrainedSelectionEditor());
    }

    public boolean areRowsValid() {
```

```
        for(T lRow : getValue()) {
            if(lRow.isRowValid() == false) {
                return false;
            }
        }
        return true;
    }
}
```

```java
package edu.stsci.jwst.apt.model.instrument;

import java.security.InvalidParameterException;

public class MiriImagingExposureTable extends MiriExposureTable<MiriImagingExposureTableRow> {

    public MiriImagingExposureTable (TinaDocumentElement iContainer, String iName) {
        super(iContainer, iName);
    }

    @Override
    public int getColumnCount () {
        return 6;
    }

    @Override
    public Class<?> getColumnClass (int iColumn) {
        switch (iColumn) {
            case 0: return CosiConstrainedSelection.class;
            case 1: return CosiConstrainedDouble.class;
            case 2: return CosiConstrainedSelection.class;
            case 3: return CosiConstrainedInt.class;
            case 4: return CosiConstrainedInt.class;
            case 5: return CosiDerivedProperty.class;
            default:
                throw new InvalidParameterException("Expected column less than " + getColumnCount());
        }
    }

    // BE ADVISED!!! There is some inconsistent behavior in the order properties are fired
    // between ConstrainedInt and ConstrainedSelection.  This means that when this method is
    // called, ConstrainedSelection still has its old, original value, but ConstrainedInt
    // has been updated with its new value!
    @Override
    public void setValueAt (Object iValue, int iRow, int iColumn) {
        MiriImagingExposureTableRow lField = getValue().get(iRow);

        if ((iValue != null) && (!"".equals(iValue.toString()))) {
            switch (iColumn) {
                case 0: lField.setFilter((MiriFilter)iValue); break;
                case 1: lField.setRequestedExpTime(Double.parseDouble(iValue.toString())); break;
                case 2: lField.setReadoutPattern((MiriReadPattern)iValue); break;
                case 3: lField.setNumberOfGroups(Integer.parseInt(iValue.toString())); break;
                case 4: lField.setNumberOfIntegrations(Integer.parseInt(iValue.toString())); break;
                case 5: break;//TODO ROB See if this ever needs to be settable during read....lField.setCalculatedExpTime
(Double.parseDouble(iValue.toString())); break;
```

```
                default:
                    throw new InvalidParameterException("Col count was "+iColumn+".  Expected column less than " +
getColumnCount());
            }
        }
    }

    @Override
    public String getColumnName(int iColumn) {
        switch (iColumn) {
            case 0: return MiriInstrument.FILTER;
            case 1: return MiriInstrument.REQUESTED_EXPOSURE_TIME;
            case 2: return MiriInstrument.READOUT_PATTERN;
            case 3: return MiriInstrument.NUMBER_OF_GROUPS;
            case 4: return MiriInstrument.NUMBER_OF_INTS;
            case 5: return MiriInstrument.CALCULATED_EXP_TIME;
            default:
                throw new InvalidParameterException("Col count was "+iColumn+".  Expected column less than " +
getColumnCount());
        }
    }

    @Override
    protected MiriImagingExposureTableRow newField() {
        MiriImagingExposureTableRow lRow = new MiriImagingExposureTableRow(this);
        return lRow;
    }

    @Override
    public Object getValueAt(int rowIndex, int iColumn) {
        MiriImagingExposureTableRow lField = getValue().get(rowIndex);
        switch (iColumn) {
            case 0: return lField.getFilterObject();
            case 1: return lField.getRequestedExpTimeObject();
            case 2: return lField.getReadoutPatternObject();
            case 3: return lField.getNumberOfGroupsObject();
            case 4: return lField.getNumberOfIntegrationsObject();
            case 5: return lField.getCalculatedExpTimeObject();
        }
        throw new InvalidParameterException("Col count was "+iColumn+".  Expected column less than " + getColumnCount());
    }
}
```

```
package edu.stsci.jwst.apt.model.instrument;

import java.security.InvalidParameterException;

public class MiriCoronExposureTable extends MiriExposureTable<MiriCoronExposureTableRow> {

    public MiriCoronExposureTable (TinaDocumentElement iContainer, String iName) {
        super(iContainer, iName);
    }

    @Override
    public int getColumnCount () {
        return 6;
    }

    @Override
    public Class<?> getColumnClass (int iColumn) {
        switch (iColumn) {
            case 0: return CosiConstrainedSelection.class;
            case 1: return CosiConstrainedDouble.class;
            case 2: return CosiConstrainedSelection.class;
            case 3: return CosiConstrainedInt.class;
            case 4: return CosiConstrainedInt.class;
            case 5: return CosiDerivedProperty.class;
            default:
                throw new InvalidParameterException("Expected column less than " + getColumnCount());
        }
    }

    // BE ADVISED!!! There is some inconsistent behavior in the order properties are fired
    // between ConstrainedInt and ConstrainedSelection.  This means that when this method is
    // called, ConstrainedSelection still has its old, original value, but ConstrainedInt
    // has been updated with its new value!
    @Override
    public void setValueAt (Object iValue, int iRow, int iColumn) {
        MiriCoronExposureTableRow lField = getValue().get(iRow);

        if ((iValue != null) && (!"".equals(iValue.toString()))) {
            switch (iColumn) {
                case 0: lField.setCoronFilter((MiriCoronFilter)iValue); break;
                case 1: lField.setRequestedExpTime(Double.parseDouble(iValue.toString())); break;
                case 2: lField.setReadoutPattern((MiriReadPattern)iValue); break;
                case 3: lField.setNumberOfGroups(Integer.parseInt(iValue.toString())); break;
                case 4: lField.setNumberOfIntegrations(Integer.parseInt(iValue.toString())); break;
                case 5: break;//TODO ROB See if this ever needs to be settable during read....lField.setCalculatedExpTime
(Double.parseDouble(iValue.toString())); break;
```

```
                default:
                    throw new InvalidParameterException("Col count was "+iColumn+".  Expected column less than " +
getColumnCount());
            }
        }
    }

    @Override
    public String getColumnName(int iColumn) {
        switch (iColumn) {
            case 0: return MiriInstrument.CORON_FILTER;
            case 1: return MiriInstrument.REQUESTED_EXPOSURE_TIME;
            case 2: return MiriInstrument.READOUT_PATTERN;
            case 3: return MiriInstrument.NUMBER_OF_GROUPS;
            case 4: return MiriInstrument.NUMBER_OF_INTS;
            case 5: return MiriInstrument.CALCULATED_EXP_TIME;
            default:
                throw new InvalidParameterException("Col count was "+iColumn+".  Expected column less than " +
getColumnCount());
        }
    }

    @Override
    protected MiriCoronExposureTableRow newField() {
        MiriCoronExposureTableRow lRow = new MiriCoronExposureTableRow(this);
        return lRow;
    }

    @Override
    public Object getValueAt(int rowIndex, int iColumn) {
        MiriCoronExposureTableRow lField = getValue().get(rowIndex);
        switch (iColumn) {
            case 0: return lField.getCoronFilterObject();
            case 1: return lField.getRequestedExpTimeObject();
            case 2: return lField.getReadoutPatternObject();
            case 3: return lField.getNumberOfGroupsObject();
            case 4: return lField.getNumberOfIntegrationsObject();
            case 5: return lField.getCalculatedExpTimeObject();
        }
        throw new InvalidParameterException("Col count was "+iColumn+".  Expected column less than " + getColumnCount());
    }
}
```

```
package edu.stsci.jwst.apt.model.instrument;

import java.security.InvalidParameterException;

public class MiriDarkExposureTable extends MiriExposureTable<MiriDarkExposureTableRow> {

    public MiriDarkExposureTable (TinaDocumentElement iContainer, String iName) {
        super(iContainer, iName);
    }

    @Override
    public int getColumnCount () {
        return 5;
    }

    @Override
    public Class<?> getColumnClass (int iColumn) {
        switch (iColumn) {
            case 0: return CosiConstrainedInt.class;
            case 1: return CosiConstrainedSelection.class;
            case 2: return CosiConstrainedSelection.class;
            case 3: return CosiConstrainedInt.class;
            case 4: return CosiConstrainedInt.class;
            default:
                throw new InvalidParameterException("Expected column less than " + getColumnCount());
        }
    }

    // BE ADVISED!!! There is some inconsistent behavior in the order properties are fired
    // between ConstrainedInt and ConstrainedSelection.  This means that when this method is
    // called, ConstrainedSelection still has its old, original value, but ConstrainedInt
    // has been updated with its new value!
    @Override
    public void setValueAt (Object iValue, int iRow, int iColumn) {
        MiriDarkExposureTableRow lField = getValue().get(iRow);

        if ((iValue != null) && (!"".equals(iValue.toString()))) {
            switch (iColumn) {
                case 0: lField.setNumberOfExposures(Integer.parseInt(iValue.toString())); break;
                case 1: lField.setFilter((MiriFilter)iValue); break;
                case 2: lField.setReadoutPattern((MiriReadPattern)iValue); break;
                case 3: lField.setNumberOfGroups(Integer.parseInt(iValue.toString())); break;
                case 4: lField.setNumberOfIntegrations(Integer.parseInt(iValue.toString())); break;
                default:
                    throw new InvalidParameterException("Col count was "+iColumn+".  Expected column less than " +
getColumnCount());
```

```
            }
        }
    }

    @Override
    public String getColumnName(int iColumn) {
        switch (iColumn) {
            case 0: return MiriInstrument.NUMBER_OF_EXPS;
            case 1: return MiriInstrument.FILTER;
            case 2: return MiriInstrument.READOUT_PATTERN;
            case 3: return MiriInstrument.NUMBER_OF_GROUPS;
            case 4: return MiriInstrument.NUMBER_OF_INTS;
            default:
                throw new InvalidParameterException("Col count was "+iColumn+".  Expected column less than " +
getColumnCount());
        }
    }

    @Override
    protected MiriDarkExposureTableRow newField() {
        MiriDarkExposureTableRow lRow = new MiriDarkExposureTableRow(this);
        return lRow;
    }

    @Override
    public Object getValueAt(int rowIndex, int iColumn) {
        MiriDarkExposureTableRow lField = getValue().get(rowIndex);
        switch (iColumn) {
            case 0: return lField.getNumberOfExposuresObject();
            case 1: return lField.getFilterObject();
            case 2: return lField.getReadoutPatternObject();
            case 3: return lField.getNumberOfGroupsObject();
            case 4: return lField.getNumberOfIntegrationsObject();
        }
        throw new InvalidParameterException("Col count was "+iColumn+".  Expected column less than " + getColumnCount());
    }
}
```

```java
package edu.stsci.jwst.apt.model.instrument;

import edu.stsci.jwst.apt.model.instrument.MiriInstrument.MiriObjectType;

public interface MiriExposure {
    public CosiConstrainedSelection<MiriSubarray> getSubarrayField();
    public MiriSubarray getSubarray();
    public CosiConstrainedSelection<MiriObjectType> getObjectTypeField();
    public MiriObjectType getObjectType();
    public Double getRequestedExpTime();
    public MiriReadPattern getReadoutPattern();
    public Integer getNumberOfGroups();
    public Integer getNumberOfIntegrations();
    public Double getCalculatedExpTime();
    public boolean isCommandingMode();
}
```

```
package edu.stsci.jwst.apt.model.instrument;

import java.beans.PropertyChangeEvent;

public abstract class MiriExposureTableRow<T extends MiriExposureTable>
implements TinaMultiFieldField, MiriExposure {

    private T table;
    private PropertyChangeSupport fChangeSupport = new PropertyChangeSupport(this);

    private MiriExpTimeCalculator expTimeCalculator;

    protected CosiConstrainedSelection<MiriFilter> fFilter = MiriInstrument.makeFilterField(null);
    protected CosiConstrainedDouble fRequestedExpTime = MiriInstrument.makeRequestedExpTimeField(null);

    //Derived fields
    protected final CosiConstrainedSelection<MiriReadPattern> fReadoutPattern = MiriInstrument.makeReadoutPatternField(null);
    protected final CosiConstrainedInt fNumberOfGroups = MiriInstrument.makeNumberOfGroupsField(null);
    protected final CosiConstrainedInt fNumberOfIntegrations = MiriInstrument.makeNumberOfIntsField(null);
    protected final CosiDerivedProperty<Double> fCalculatedExpTime =
        CosiDerivedProperty.createUninitializedProperty(this, 0.0, new Calculator<Double>(){
            public Double calculate() {
                double lExpTime = 0.0;
                updateExposureTime();
                if (expTimeCalculator.getCalculatedExpTime()!=null) {
                    lExpTime = expTimeCalculator.getCalculatedExpTime();
                }
                return lExpTime;
            }
        });

    public MiriExposureTableRow (T iContainer) {
        setTable(iContainer);
        if (!isCommandingMode()) {
            fReadoutPattern.setEditable(false);
            fNumberOfGroups.setEditable(false);
            fNumberOfIntegrations.setEditable(false);
        } else {
            fRequestedExpTime.setEditable(false);
        }
        //This has to be done last, so that other fields are set up properly.
        expTimeCalculator = new MiriExpTimeCalculator(this);
        Propagator.completeInitialization(this,MiriExposureTableRow.class);
    }

    public boolean isCommandingMode() {
```

```
    return false;
}

public MiriFilter getFilter() {
    return fFilter.getValue();
}

public Double getRequestedExpTime() {
    return fRequestedExpTime.getValue();
}

public MiriReadPattern getReadoutPattern() {
    return fReadoutPattern.getValue();
}

public Integer getNumberOfGroups() {
    return fNumberOfGroups.getValue();
}

public Integer getNumberOfIntegrations() {
    return fNumberOfIntegrations.getValue();
}

public Double getCalculatedExpTime() {
    return expTimeCalculator.getCalculatedExpTime();
}

protected CosiConstrainedSelection<MiriFilter> getFilterObject() {
    return fFilter;
}

protected CosiConstrainedDouble getRequestedExpTimeObject() {
    return fRequestedExpTime;
}

protected CosiConstrainedSelection<MiriReadPattern> getReadoutPatternObject() {
    return fReadoutPattern;
}

protected CosiConstrainedInt getNumberOfGroupsObject () {
    return fNumberOfGroups;
}

protected CosiConstrainedInt getNumberOfIntegrationsObject() {
    return fNumberOfIntegrations;
}
```

```
protected CosiDerivedProperty<Double> getCalculatedExpTimeObject() {
    return fCalculatedExpTime;
}

public void setFilter(MiriFilter iFilter) {
    fFilter.set(iFilter);
}

public void setRequestedExpTime(double iRequestedExpTime) {
    fRequestedExpTime.set(iRequestedExpTime);
    updateExposureTime();
}

protected void updateExposureTime() {
    //Set derived values
    if (!isCommandingMode()) {
        setReadoutPattern(expTimeCalculator.getReadoutPattern());
        setNumberOfGroups(expTimeCalculator.getNumberOfGroups());
        setNumberOfIntegrations(expTimeCalculator.getNumberOfIntegrations());
    }
}

public void setReadoutPattern(MiriReadPattern iReadoutPattern) {
    fReadoutPattern.set(iReadoutPattern);
}

public void setReadoutPattern(String iReadoutPattern) {
    try {
        setReadoutPattern(MiriReadPattern.valueOf(iReadoutPattern));
    } catch (Exception e) {
        // TODO: handle exception
    }
}

public void setNumberOfGroups(int iNumGroups) {
    fNumberOfGroups.set(iNumGroups);
}

public void setNumberOfIntegrations(Integer iIntegrations) {
    fNumberOfIntegrations.set(iIntegrations);
}

public void setParent(TinaDocumentElement iElement) {
    fFilter.setContainer(iElement);
    fRequestedExpTime.setContainer(iElement);
```

```
    fReadoutPattern.setContainer(iElement);
    fNumberOfGroups.setContainer(iElement);
    fNumberOfIntegrations.setContainer(iElement);
}

public abstract CosiConstrainedSelection<MiriSubarray> getSubarrayField ();

public abstract MiriSubarray getSubarray();

public abstract CosiConstrainedSelection<MiriObjectType> getObjectTypeField();

public abstract MiriObjectType getObjectType();

public boolean isRowValid() {
    return fFilter.isSpecified() &&
    fRequestedExpTime.getValue() != null &&
    fReadoutPattern.isSpecified() && !fReadoutPattern.isIllegalSelection() &&
    fNumberOfGroups.getValue() != null && !fNumberOfGroups.isOutOfRange() &&
    fNumberOfIntegrations.getValue() != null && !fNumberOfIntegrations.isOutOfRange();
}

public T getTable () {
    return table;
}

public void setTable (T iTable) {
    table = iTable;
}

@Override
public abstract String toString ();

public void addPropertyChangeListener (PropertyChangeListener iListener) {
    fChangeSupport.addPropertyChangeListener(iListener);
}

public void addPropertyChangeListener (String iPropName, PropertyChangeListener iListener) {
    fChangeSupport.addPropertyChangeListener(iPropName, iListener);
}

public void removePropertyChangeListener (PropertyChangeListener iListener) {
    fChangeSupport.removePropertyChangeListener(iListener);
}

public void removePropertyChangeListener (String iPropName, PropertyChangeListener iListener) {
    fChangeSupport.removePropertyChangeListener(iPropName, iListener);
```

```
    }

    public void firePropertyChange (PropertyChangeEvent iEvent) {
        fChangeSupport.firePropertyChange(iEvent);
    }
}
```

```java
package edu.stsci.jwst.apt.model.instrument;

import java.util.Arrays;

public class MiriImagingExposureTableRow extends MiriExposureTableRow<MiriImagingExposureTable>
implements TinaMultiFieldField, MiriExposure {

    public static final MiriFilter[] imagingFilters = new MiriFilter[]{
        MiriFilter.F560W,
        MiriFilter.F770W,
        MiriFilter.F1000W,
        MiriFilter.F1130W,
        MiriFilter.F1280W,
        MiriFilter.F1500W,
        MiriFilter.F1800W,
        MiriFilter.F2100W,
        MiriFilter.F2550W};

    public MiriImagingExposureTableRow (MiriImagingExposureTable iContainer) {
        super(iContainer);
        fFilter.setLegalValues(Arrays.asList(imagingFilters));
        Propagator.completeInitialization(this,MiriImagingExposureTableRow.class);
    }

    @Override
    public CosiConstrainedSelection<MiriSubarray> getSubarrayField () {
        return ((MiriImagingTemplate)getTable().getContainer()).getSubarrayField();
    }

    @Override
    public MiriSubarray getSubarray() {
        return ((MiriImagingTemplate)getTable().getContainer()).getSubarray();
    }

    @Override
    public CosiConstrainedSelection<MiriObjectType> getObjectTypeField() {
        return ((MiriImagingTemplate)getTable().getContainer()).getObjectTypeField();
    }

    @Override
    public MiriObjectType getObjectType() {
        return ((MiriImagingTemplate)getTable().getContainer()).getObjectType();
    }

    @Override
    public String toString () {
```

```
        return fFilter+" "+fRequestedExpTime+" "+fReadoutPattern+" "+fNumberOfGroups+" "+fNumberOfIntegrations;
    }
}
```

```java
package edu.stsci.jwst.apt.model.instrument;

import java.util.Arrays;

public class MiriCoronExposureTableRow extends MiriExposureTableRow<MiriCoronExposureTable>
implements TinaMultiFieldField, MiriExposure {

    private CosiConstrainedSelection<MiriCoronFilter> fCoronFilter = MiriInstrument.makeCoronFilterField(null);
    private CosiConstrainedSelection<MiriMask> fMask = MiriInstrument.makeMaskField(null);

    public static final MiriFilter[] coronagraphicFilters = new MiriFilter[]{
        MiriFilter.F1065C,
        MiriFilter.F1140C,
        MiriFilter.F1550C,
        MiriFilter.F2300C};

    {
        Propagator.addDelayedConstraint(new Constraint(this, "Filter/Mask selection"){
           public void run () {
               if (fCoronFilter.isSpecified()) {
                   fFilter.set(fCoronFilter.get().getFilter());
                   fMask.set(fCoronFilter.get().getMask());
               }
           }
        });
    }

    public MiriCoronExposureTableRow (MiriCoronExposureTable iContainer) {
        super(iContainer);
        fFilter.setLegalValues(Arrays.asList(coronagraphicFilters));
        fFilter.setEditable(false);
        fMask.setEditable(false);
        Propagator.completeInitialization(this,MiriCoronExposureTableRow.class);
    }

    protected CosiConstrainedSelection<MiriCoronFilter> getCoronFilterObject() {
        return fCoronFilter;
    }

    public MiriCoronFilter getCoronFilter() {
        return fCoronFilter.get();
    }

    protected CosiConstrainedSelection<MiriMask> getMaskObject() {
        return fMask;
    }
```

```
public MiriMask getMask() {
    return fMask.get();
}

public void setCoronFilter(MiriCoronFilter iCoronFilter) {
    fCoronFilter.set(iCoronFilter);
}

public void setMask(MiriMask iMask) {
    fMask.set(iMask);
}

@Override
public void setParent(TinaDocumentElement iElement) {
    fCoronFilter.setContainer(iElement);
    fMask.setContainer(iElement);
    super.setParent(iElement);
}

@Override
public CosiConstrainedSelection<MiriSubarray> getSubarrayField () {
    return null;
}

@Override
public MiriSubarray getSubarray() {
    return null;
}

@Override
public CosiConstrainedSelection<MiriObjectType> getObjectTypeField() {
    return ((MiriCoronImagingTemplate)getTable().getContainer()).getObjectTypeField();
}

@Override
public MiriObjectType getObjectType() {
    return ((MiriCoronImagingTemplate)getTable().getContainer()).getObjectType();
}

@Override
public boolean isRowValid() {
    return  fCoronFilter.isSpecified() &&
    fMask.isSpecified() &&
    super.isRowValid();
}
```

```
    @Override
    public String toString () {
        return fCoronFilter+" "+fMask+" "+fFilter+" "+fRequestedExpTime+" "+fReadoutPattern+" "+fNumberOfGroups+"
"+fNumberOfIntegrations;
    }
}
```

```java
package edu.stsci.jwst.apt.model.instrument;

import edu.stsci.CoSI.Propagator;

public class MiriDarkExposureTableRow extends MiriExposureTableRow<MiriDarkExposureTable> implements TinaMultiFieldField,
MiriExposure {
    private CosiConstrainedInt numExps;

    public MiriDarkExposureTableRow (MiriDarkExposureTable iContainer) {
        super(iContainer);
        fReadoutPattern.setEditable(true);
        fNumberOfGroups.setEditable(true);
        fNumberOfIntegrations.setEditable(true);
        numExps  = MiriInstrument.makeNumberOfExpsField(getTemplate());
        Propagator.completeInitialization(this,MiriDarkExposureTableRow.class);
    }

    public MiriDarkTemplate getTemplate() {
        MiriDarkExposureTable lTable = getTable();
        MiriDarkTemplate lObs = (MiriDarkTemplate)lTable.getContainer();
        return lObs;
    }

    @Override
    public MiriObjectType getObjectType() {
        return null;
    }

    @Override
    public CosiConstrainedSelection<MiriObjectType> getObjectTypeField() {
        return null;
    }

    @Override
    public MiriSubarray getSubarray() {
        return null;
    }

    @Override
    public CosiConstrainedSelection<MiriSubarray> getSubarrayField() {
        return null;
    }

    public void setNumberOfExposures (int iNumExps) {
        this.numExps.set(iNumExps);
    }
```

```
    public int getNumberOfExposures() {
        return numExps.get();
    }

    public CosiConstrainedInt getNumberOfExposuresObject() {
        return numExps;
    }

    @Override
    public String toString () {
        return fReadoutPattern+" "+fNumberOfGroups+" "+fNumberOfIntegrations;
    }
}
```

```java
package edu.stsci.jwst.apt.model.instrument;

import edu.stsci.CoSI.Calculator;

public class MiriLrsExposure implements MiriExposure {

    private MiriLrsTemplate template;
    private MiriExpTimeCalculator expTimeCalculator;

    private final CosiDerivedProperty<Double> calculatedExpTime =
        CosiDerivedProperty.createUninitializedProperty(this, 0.0, new Calculator<Double>(){
            public Double calculate() {
                return expTimeCalculator.getCalculatedExpTime();
            }
        });

    public MiriLrsExposure(MiriLrsTemplate iTemplate) {
        template = iTemplate;
        expTimeCalculator = new MiriExpTimeCalculator(this);
        Propagator.completeInitialization(this, MiriLrsExposure.class);
    }

    public Integer getNumberOfGroups() {
        return expTimeCalculator.getNumberOfGroups();
    }

    public Integer getNumberOfIntegrations() {
        return expTimeCalculator.getNumberOfIntegrations();
    }

    public MiriObjectType getObjectType() {
        return template.getObjectType();
    }

    public CosiConstrainedSelection<MiriObjectType> getObjectTypeField() {
        return template.getObjectTypeField();
    }

    public MiriReadPattern getReadoutPattern() {
        return expTimeCalculator.getReadoutPattern();
    }

    public Double getRequestedExpTime() {
        return template.getRequestedExpTime();
    }
```

```
    public MiriSubarray getSubarray() {
        return null;
    }

    public CosiConstrainedSelection<MiriSubarray> getSubarrayField() {
        return null;
    }

    public Double getCalculatedExpTime() {
        return calculatedExpTime.get();
    }

    public boolean isCommandingMode () {
        return false;
    }
}
```

```java
package edu.stsci.jwst.apt.model.instrument;

import edu.stsci.CoSI.Calculator;

public class MiriMrsExposure implements MiriExposure {

    MiriMrsTemplate template;
    private MiriExpTimeCalculator expTimeCalculator;

    private final CosiDerivedProperty<Double> calculatedExpTime =
        CosiDerivedProperty.createUninitializedProperty(this, 0.0, new Calculator<Double>(){
            public Double calculate() {
                return expTimeCalculator.getCalculatedExpTime();
            }
        });

    public MiriMrsExposure(MiriMrsTemplate iTemplate) {
        template = iTemplate;
        expTimeCalculator = new MiriExpTimeCalculator(this);
        Propagator.completeInitialization(this, MiriMrsExposure.class);
    }

    public Integer getNumberOfGroups() {
        return expTimeCalculator.getNumberOfGroups();
    }

    public Integer getNumberOfIntegrations() {
        return expTimeCalculator.getNumberOfIntegrations();
    }

    public MiriObjectType getObjectType() {
        return template.getObjectType();
    }

    public CosiConstrainedSelection<MiriObjectType> getObjectTypeField() {
        return template.getObjectTypeField();
    }

    public MiriReadPattern getReadoutPattern() {
        return expTimeCalculator.getReadoutPattern();
    }

    public Double getRequestedExpTime() {
        return template.getRequestedExpTime();
    }
```

```
    public MiriSubarray getSubarray() {
        return null;
    }

    public CosiConstrainedSelection<MiriSubarray> getSubarrayField() {
        return null;
    }

    public Double getCalculatedExpTime() {
        return calculatedExpTime.get();
    }

    public boolean isCommandingMode () {
        return false;
    }
}
```

```
package edu.stsci.jwst.apt.model.instrument;

import java.text.DecimalFormat;


public class MiriExpTimeCalculator {
    private static final double SHORT_TGROUP = 0.69;
    private static final double FAINT_SUB_MAX = 30.0;
    private static final double SLOW_THRESHOLD_TIME = 120.0;
    private static final double GROUP_MAX_THRESHOLD = 106.0;
    private static final double FULL_FAST_GROUP = 2.64;
    private static final double FULL_FAST_MAX = 105.6;
    private static final double FULL_SLOW_GROUP = 26.4;
    private static final double FULL_SLOW_MAX = 1020.0;

    //derived parameters
    private final CosiDerivedProperty<MiriReadPattern> readoutPattern =
        CosiDerivedProperty.createUninitializedProperty(this, null, new Calculator<MiriReadPattern>(){
            public MiriReadPattern calculate() {
                MiriSubarray lSubarray = getSubarray();
                if (lSubarray==null) {
                    lSubarray = MiriSubarray.FULL;
                }
                switch (lSubarray) {
                    case BRIGHTSKY:
                        return MiriReadPattern.FAST;
                    case FULL:
                    default:
                        if (getObjectType()!=null) {
                            switch (getObjectType()) {
                                case BRIGHT:
                                    return MiriReadPattern.FAST;
                                case FAINT:
                                    if ((getRequestedExpTime()!=null &&
                                            getRequestedExpTime()<SLOW_THRESHOLD_TIME)) {
                                        return MiriReadPattern.FAST;
                                    } else {
                                        return MiriReadPattern.SLOW;
                                    }
                                default:
                                    break;
                            }
                        }
                }
                return null;
            }
```

```
    });
private final CosiDerivedProperty<Double> groupTime =
    CosiDerivedProperty.createUninitializedProperty(this, 0.0, new Calculator<Double>(){
        public Double calculate() {
            MiriSubarray lSubarray = getSubarray();
            if (lSubarray==null) {
                lSubarray = MiriSubarray.FULL;
            }
            switch (lSubarray) {
                case BRIGHTSKY:
                    return SHORT_TGROUP;
                case FULL:
                default:
                    if (getObjectType()!=null) {
                        switch (getObjectType()) {
                            case BRIGHT:
                                return FULL_FAST_GROUP;
                            default:
                                break;
                        }
                    }
            }
            if ((getRequestedExpTime()!=null &&
                    getRequestedExpTime()<GROUP_MAX_THRESHOLD)) {
                return FULL_FAST_GROUP;
            } else {
                return FULL_SLOW_GROUP;
            }
        }
    });
private final CosiDerivedProperty<Double> maxTime =
    CosiDerivedProperty.createUninitializedProperty(this, 0.0, new Calculator<Double>(){
        public Double calculate() {
            MiriSubarray lSubarray = getSubarray();
            if (lSubarray==null) {
                lSubarray = MiriSubarray.FULL;
            }
            switch (lSubarray) {
                case BRIGHTSKY:
                    if (getObjectType()!= null) {
                        switch (getObjectType()) {
                            case BRIGHT:
                                return getGroupTime();
                            case FAINT:
                                return FAINT_SUB_MAX;
                            default:
```

```
                                break;
                            }
                        }
                        break;
                    case FULL:
                    default:
                        if (getObjectType()!= null) {
                            switch (getObjectType()) {
                                case BRIGHT:
                                    return getGroupTime();
                                case FAINT:
                                    break;
                            }
                        }
                }
                if ((getRequestedExpTime()!=null &&
                        getRequestedExpTime()<GROUP_MAX_THRESHOLD)) {
                    return FULL_FAST_MAX;
                } else {
                    return FULL_SLOW_MAX;
                }
            }
        });
    private final CosiDerivedProperty<Integer> numberOfIntegrations =
        CosiDerivedProperty.createUninitializedProperty(this, 0, new Calculator<Integer>(){
            public Integer calculate() {
                if (getRequestedExpTime()!=null && getMaxTime()!=null && getMaxTime()!=0.0) {
                    return (int)Math.ceil(getRequestedExpTime()/getMaxTime());
                }
                return 0;
            }
        });
    private final CosiDerivedProperty<Integer> numberOfGroups =
            CosiDerivedProperty.createUninitializedProperty(this, 0, new Calculator<Integer>(){
                public Integer calculate() {
                    if (getRequestedExpTime()!=null &&
                            getNumberOfIntegrations()!=null && getNumberOfIntegrations()!=0 &&
                            getGroupTime()!=null && getGroupTime()!=0) {
                        return (int)Math.ceil((getRequestedExpTime()/getNumberOfIntegrations())/getGroupTime());
                    }
                    return 0;
                }
            });
    private final CosiDerivedProperty<Double> calculatedExpTime =
                CosiDerivedProperty.createUninitializedProperty(this, 0.0, new Calculator<Double>(){
                    public Double calculate() {
```

```
                    double lExpTime = 0.0;
                    if (getNumberOfIntegrations()!=null &&
                            getNumberOfGroups()!=null &&
                            getGroupTime()!=null) {
                        //This has to be done in this order to avoid inadvertent upcasting during multiplication.
                        lExpTime =  getGroupTime()*getNumberOfIntegrations()*getNumberOfGroups();
                        // HACK: Correct the stupid Java roundoff error (e.g. don't return
                        // something like 37.23000000000000001 when adding 34.11 and 3.12)
                        DecimalFormat df = new DecimalFormat("0.00");
                        lExpTime = Double.parseDouble(df.format(lExpTime));
                    }
                    return lExpTime;
                }
            });

private MiriExposure exposureSpec;

public MiriExpTimeCalculator (MiriExposure iExposureSpec) {
    exposureSpec = iExposureSpec;
    Propagator.completeInitialization(this, MiriExpTimeCalculator.class);
}

//Accessors: Getters/Setters
public Double getRequestedExpTime() {
    return exposureSpec.getRequestedExpTime();
}
public MiriSubarray getSubarray() {
    return exposureSpec.getSubarray();
}
public MiriObjectType getObjectType() {
    return exposureSpec.getObjectType();
}
public MiriReadPattern getReadoutPattern() {
    return readoutPattern.get();
}
public Double getGroupTime() {
    return groupTime.get();
}
public Double getMaxTime() {
    return maxTime.get();
}
public Integer getNumberOfIntegrations() {
    return numberOfIntegrations.get();
}
public Integer getNumberOfGroups() {
    return numberOfGroups.get();
```

```
    }
    public Double getCalculatedExpTime() {
        return calculatedExpTime.get();
    }
}
```