

Click on “Sign up” on the right upper side of the web page.



Signup

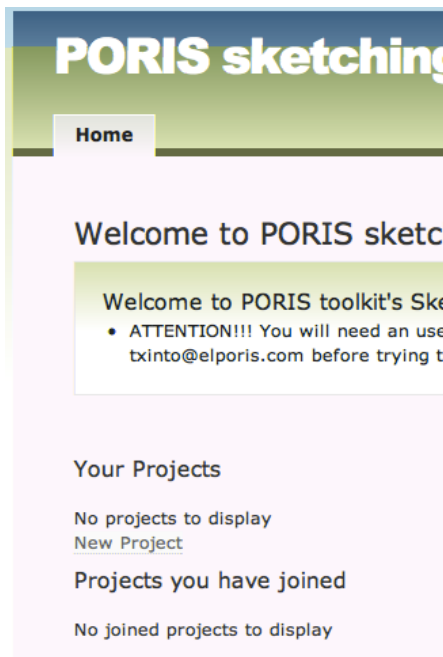
Name	<input type="text" value="Txinto Vaz"/>
Abbrev	<input type="text" value="txinto"/>
Email Address	<input type="text" value="txinto@elporis.com"/>
Password	<input type="password" value="....."/>
Password Confirmation	<input type="password" value="....."/>

or

Fill the info...



Check that you are actually logged in.



This is your welcome page(left):

The first thing we must do is to create a new project to work in, so click on “New Project” and give it a name and abbrev. In the web the project name and project abbrev must be unique (like in a company), so use your abbrev as prefix to avoid colliding with other users’ tutorials. The resulting name will be “[Prefix]Test”.

Home

New Project  
For: Txinto Vaz

Name	<input type="text" value="TxTest"/>
Abbrev	<input type="text" value="txtest"/>

Now you have a project that you own. The members of the project can access it, so you can give access to your project to anyone on the web using the “New Membership” link.

**Abbrev** txtest

**Members**

**Project Membership**

No members in this project  
[New Membership](#)

**Libraries**

**Library** **Pa**

No libraries in this project  
[New Library in TxTest](#)

For this tutorial we will add ourselves to the project in the role of a system engineer. We will also check the contributor field to allow us to modify the project. If this is not checked, then the person is added only as observer. The owner of the project can also access and modify it, but it is a good practice to add the owner to the project in the role he/she acts.

**New Project Membership**  
For: TxTest

**Contributor** ☒

**Role** Systems Engineer

**User** Txinto Vaz

[Create Project Membership](#)

We click back to our project's home page by clicking the "<<[Prefix]Test" link.

The project contains systems, but they are inside libraries. The libraries allow us to group several systems in a kind of "folders" to keep them organized. Let's create a library called PrefixTestLib to host our TestSystem using the "New Library in [Prefix]Test" project. Do not choose any super or sublibs.

**Libraries**

**Library**

No libraries in this project  
[New Library in TxTest](#)

**New Library**

**Name** TxTestLib

**Super Libs** [Add Super Lib](#)

**Sub Libs** [Add Sub Lib](#)

In the next screen we will see the details of the library. We can see it is linked to [Prefix] Test project. In the right side of the page we can see the library icon as a root of a tree.

The library was created successfully


**TxTestLib**

---

The library was created successfully

<b>Name</b>	TxTestLib
<b>Node Type</b>	Library
<b>Project</b>	TxTest

[Edit Library](#)

 TxTestLib

We will create a [Prefix]TestInstrument in that library using the “New System” link.

**Systems**

---

No systems to display  
[New System in TxTestLib](#)

Just give it a name ([Prefix]TestInstrument), select the “Node Type” as “System” and press the “Create SubSystem” link.

**New Sub System**

---

**Name**

**Default Mode**

---

**Modes**

**Node Type**

**Libraries**

---

**Sub Systems**

**SuperSystems to include in:**

**Values in this sub system:**

or

Please note that, in concept, systems, subsystems and parameters are the same. The only difference between them is the emphasis on the semantics :

- We will call it system when we want to emphasize that that subsystem is the target of our development (the instrument in our case).
- We will call it parameter when does not contain any other subsystem inside it, normally it will only take a value.

- We will call it subsystem in the rest of the cases: contain other subsystems and it is not the target of our development.

Normally, the project will contain libraries, and the libraries will contain systems, and the system will contain other subsystems as a tree architecture. When a node does not contain other subsystem and takes a value it is called a parameter. Take these ideas in consideration:

- Depending on the “node type” you have chosen, the web editor will display an icon or another in the system tree in the right side of the web page.
- If you include a system inside another bigger one, the only thing you will need is to change the “node type” to obtain the correct icon in your tree.
- If you include a new subsystem into a parameter the only thing you must do is to change the “node type” to obtain the correct icon in your tree.

The next screen will show to you the [Prefix]TestInstrument details.

The sub system was created successfully

TxTestInstrument

[Edit Sub System](#)

The sub system was created successfully

<b>Node Type</b>	<a href="#">System</a>
<b>Project</b>	<a href="#">TxTest</a>
<b>Default Mode</b>	(Not Available)

**Labels**

No labels to display

[New Label for TxTestInstrument](#)


**Libraries**

**TxTestLib**

SubLibraries: None

Systems: [TxTestInstrument](#)

**Super Systems**



TxTestInstrument

You will see the icon on the right side of the screen. Why is not the library there? Because you are looking at the instrument level, that is a level below the library. We can see “Libraries” section in this page and can find the [Prefix]TestLib library there.

If we click on “[Prefix]TestLib” link, you will arrive to the Library details page and see the tree on the right side.

TxTestLib

Edit Library



TxTestLib



TxTestInstrument

If you click on the '+' sign near its icon it will open showing to you the [Prefix] TestInstrument inside of it. You can use the tree to navigate to the instrument again to continue this tutorial.

TxTestInstrument

Edit Sub System

TxTestInstrument

Node Type

System

Project

TxTest

Default Mode

(Not Available)

Labels

No labels to display  
New Label for TxTestInstrument

Libraries

TxTestLib

SubLibraries: None

Systems: TxTestInstrument

Super Systems

No super systems to display

Sub Systems

No sub systems to display  
New Sub System in TxTestInstrument

Our instrument will have 4 main subsystems:

- Masks
- Filters
- Dispersion
- Detector

So we will use the “New Subsystem” link to create the first one:

## New Sub System

Name

TxMasks

Default Mode

First add some modes

Modes

Add Mode

Node Type

SubSystem

Libraries

Add Library

Sub Systems

Add Sub System

SuperSystems to include in:

TxTestInstrument

Add Super System

Values in this sub system:

Add Value

Create Sub System

The sub system was created successfully

## TxMasks

[Edit Sub System](#)

The sub system was created successfully

Node Type

SubSystem

Project

TxTest

Default Mode

(Not Available)

Labels

No labels to display

[New Label for TxMasks](#)


Libraries

No libraries to display

Super Systems

**TxTestInstrument**

Values: None

 TxMasks

Now we will click on “Super Systems” section, in the [Prefix]TestInstrument link to return to the instrument...

**TxTestInstrument**

[Edit Sub System](#)

**Node Type**      [System](#)

**Project**          [TxTest](#)

**Default Mode**    (Not Available)

Labels

No labels to display

[New Label for TxTestInstrument](#)

Libraries

**TxTestLib**

SubLibraries: None

Systems: [TxTestInstrument](#)

Super Systems

No super systems to display

Sub Systems

**TxMasks**



Values: None

Values: None

Modes: None

SubSystems: None

[New Sub System in TxTestInstrument](#)

 **TxTestInstrument** 

... and add the second one using the “new SubSystem” link:

### New Sub System

<b>Name</b>	<input type="text" value="TxFilters"/>
<b>Default Mode</b>	<input type="text" value="First add some modes"/>
<b>Modes</b>	<input type="text" value="Add Mode"/>
<b>Node Type</b>	<input type="text" value="SubSystem"/>
<b>Libraries</b>	<input type="text" value="Add Library"/>
<b>Sub Systems</b>	<input type="text" value="Add Sub System"/>
<b>SuperSystems to include in:</b>	<input type="text" value="TxTestInstrument"/>
	<input type="text" value="Add Super System"/>
<b>Values in this sub system:</b>	<input type="text" value="Add Value"/>

[Create Sub System](#)

We must repeat these steps creating the [Prefix]Dispersion and [Prefix]Detector subsystems always at the [Prefix]TestInstrument level. We will say the instrument is the parent of these four subsystems and the subsystems are children of it. If we go to the Instrument details and go to the library details we can see the whole tree on the right side of the web page:

### TxTestInstrument

[Edit Sub System](#)

<b>Node Type</b>	System
<b>Project</b>	TxTest
<b>Default Mode</b>	(Not Available)

Labels

```

graph TD
    TxTestInstrument --> TxMasks
    TxTestInstrument --> TxFilters
    TxTestInstrument --> TxDispersion
    TxTestInstrument --> TxDetector
  
```

Now we will generate our first automated product: a GraphML representation of our instrument, so we will navigate into the [Prefix]TestInstrument using the tree (you can also use the -> arrow in the tree to open a new window at that level) and we will go at the bottom of that page, right-clicking in the “Generate GraphML Code...” link in the Code Generation Tools section. We will choose “Save link as...” and store it with the “instrument.graphml” file name.





Now you can open the file using the yed editor (you can find it at [www.yworks.com/en/products\\_yed\\_download.html](http://www.yworks.com/en/products_yed_download.html) web page). It will show a big box called [Prefix] TestInstrument and four other boxes inside. You can try to re-arrange them to fit your requirements (for printing or to be a figure in a document). Since there is not a GraphML importer already developed, the GraphML diagrams are only for illustration purposes, the changes you make on them are not managed by PORIS. In fact we will overwrite this file during our tutorial.



Now we will add some values for the Masks subsystem, we will add these values: 0.6, 1.0, 2.0. They are discrete values, so we will navigate to the [Prefix]Masks subsystem and use “New Value” link to create them.

**Sub Systems**

No sub systems to display  
New Sub System in TxM

**Values**

No values to display  
New Value in TxMasks  
New ValueString in TxM  
New ValueDoubleRange  
New ValueDateRange in

**New Value**

**Name** Tx0.6

**Modes to include in:** First choose some super systems

**Node Type** Value

**Systems to include in:** TxMasks

**Value Formatter** (No Value Formatter)

The value was created successfully

(Not Available)

**Tx0.6**

<b>Name</b>	Tx0.6
<b>Node Type</b>	Value
<b>Project</b>	TxTest
<b>Value Formatter</b>	(Not Available)

**Systems**

**TxMasks**

Values: Tx1.0, Tx2.0, Tx0.6

Values: None

Modes: None

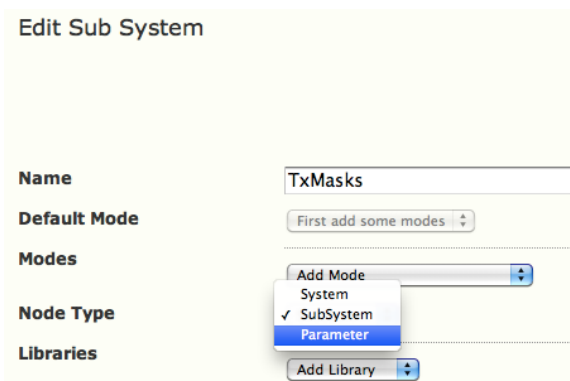
SubSystems: None

Do not choose any ValueFormatter.

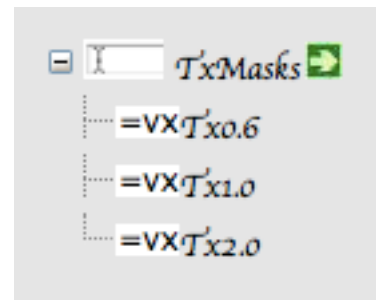
Once three values have been created, if you look the [Prefix] Masks detail page you will see the three possible values hanging from that node in the tree view.



Let's remember the "node type" concept: we see a subsystem that will not have more subsystems and the only thing can do is to take a value from three choices. We will edit it to change the "node type" from "subsystem" to "parameter" to have a more semantic tree representation on it. This can be done by clicking on the "Edit Subsystem" in the [Prefix] Masks detail page.



After that change, the icon have been changed to a more representative one.



Now we will navigate using the "Super Systems" section links until we arrive at [Prefix] TestInstrument, and let's convert to parameter node type the [Prefix]Dispersion.



Then, navigate to the [Prefix]Dispersion detail view to add the following values:

- R500
- R1000
- R2000

New Value

Name: TxR500

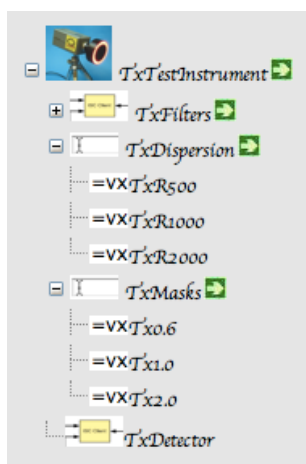
Modes to include in: First choose some s

Node Type: Value

Add System

Value Formatter: (No Value Format)

If you navigate to the [Prefix]TestInstrument details page and expand the tree you will see something similar to:



Let's introduce the two filter values into the [Prefix]Filter subsystem, and let's change it to a parameter node type.

New Value

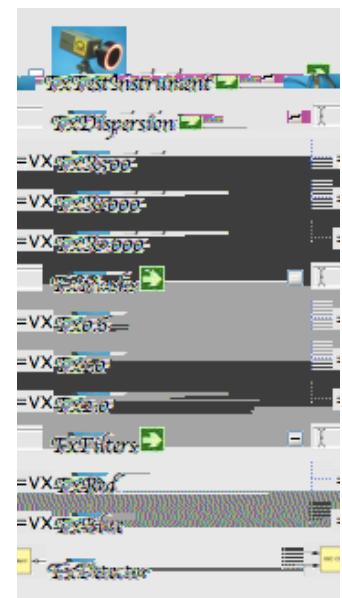
Name: TxRed

Modes to include in: First choose some s

Node Type: Value

Add System

Value Formatter: (No Value Format)



Now let's sketch the detector. Detector has two parameters: Binning and expTime. The first one must be built as a subsystem of [Prefix]Detector called [Prefix]Binning, that will have the "Parameter" node type, and that must contain the 1x1,2x1,1x2,2x2 values.

The steps are these:

- Go to the [Prefix]Detector details page.
- Click on “New SubSystem”
- Give it the [Prefix]Binning name.
- Choose “Parameter” in “Node Type” field selector.
- Click “Create SubSystem”.
- Go to the [Prefix]Binning details page (if you are not already there).
- Click on “new value”, give it the name 1x1
- Click “Create Value”.
- Repeat these last three steps with the values 2x1, 1x2 and 2x2.

### New Value

**Name**

**Modes to include in:**

**Node Type**

**Systems to include in:**

### New Sub System

**Name**

**Default Mode**

**Modes**

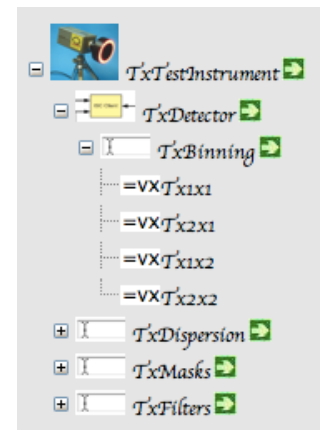
**Node Type**

**Libraries**

**Sub Systems**

**SuperSystems to include in:**

**Values in this sub system:**



After that steps you will have this instrument tree:

Now go to the [Prefix]Detector subsystem and add a subsystem called [Prefix]ExpTime, which node type would be Parameter.

The sub system was created successfully

## TxExpTime

[Edit Sub System](#)

Now we must add the expTime value. This is a new kind of value, it is not a discrete list of possible values, is a continuous range. So we will use the “New ValueDoubleRange” link to add it. Name it [Prefix]ExpTimeRange, and set rangemin to 0.0, default float to 0.01 and rangemax to 1000.0. We can choose the ValueFormatter s, that mean seconds.

### Values

No values to display

[New Value in TxExpTime](#)

[New ValueString in TxExpTime](#)

[New ValueDoubleRange in TxExpTime](#)

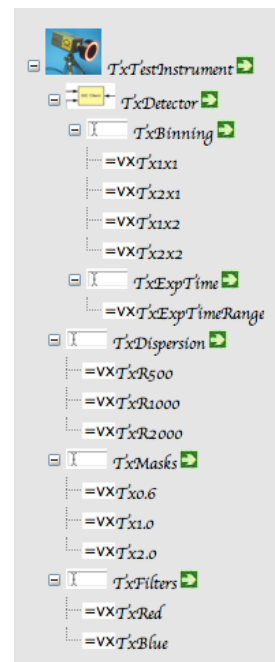
[New ValueDateRange in TxExpTime](#)

## New Value Double Range

Name	TxExpTimeRange
Rangemin	0.0
Default Float	0.01
Rangemax	1000.0
Modes	First choose some super sy
Node Type	First add some node types
Systems	TxExpTime
	Add System
Value Formatter	s

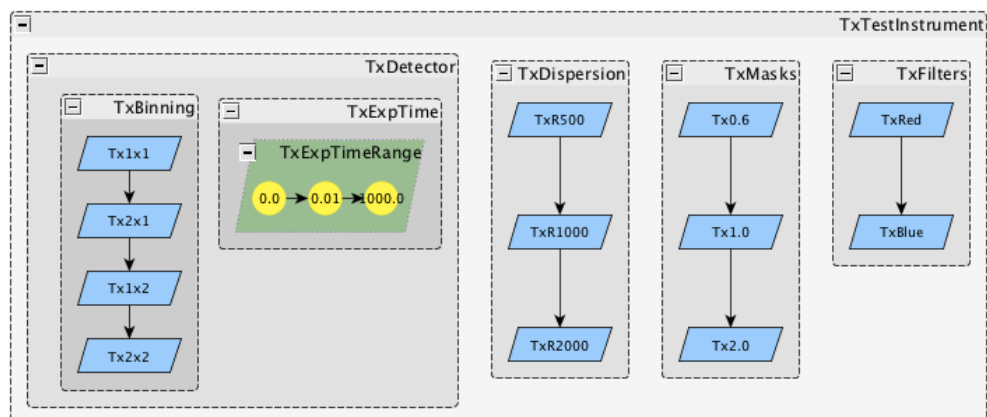
If you navigate to [Prefix]TestInstrument you will see a tree similar to this:

Let's generate the GraphML file again (right-click on "Generate GraphML ..." link and select "Save link as") and save it as instrument.graphml, overwriting the previous one.



If you open it in yEd you will find something similar to this (see the long figure on the left).

So you can arrange the figures to achieve a prettier look:



Ok!! So now you have some kind of "instrument architecture sketch". Anyone can read this diagram as:

*"An instrument has four subsystems: masks, filters, detector and dispersion. Masks can take one of the following values: 0.6, 1.0 or 2.0. Filters can take these: Red or Blue. Dispersion can take R500, R1000 or R2000. Detector is itself composed of two smaller subsystems: Binning (that can take the values 1x1, 2x1, 1x2 or 2x2) and ExpTime, which can take any value between 0.0 and 1000.0. The default value for ExpTime is 0.01."*

You can even read this:

*"The values of Masks have this natural order: 0.6, 1.0, 2.0. The filters have this natural order Red, Blue. The dispersion ones have this: R500, R1000 and R2000. And the Binning ones: 1x1, 2x1, 1x2, 2x2. The 'natural order' means that if no mode impose any other sequence, the values available for the users will be offered to them in this order. One can say that, normally, TestInstrument is a system,*

*and that Masks, Filters, Dispersion, Binning and ExpTime will be called 'parameters', while Detector will be called 'subsystem', but this is only a convention."*

In the GraphML diagram there is no difference between a parameter and a subsystem or a system, all they three are boxes.

We are very happy with our architecture but it is static. In the architecture we can not find nothing that tells us the expected behavior of the system, or the restrictions on the parameters depending on the configuration. We need to implement modes!!!

One can say that a mode of a subsystem makes two things:

- restrict the values that this subsystem can take.
- restrict the modes in that this subsystem's children can be.

This is very simple, but if every subsystem (including systems and parameters) have a mode, we can build a network of nodes linking the system mode choice to every parameter value availability.

First we will add the masks "All" mode. Let's go to the [Prefix]Masks details page and create a new mode called [Prefix]MasksAll using the "New Mode ..." link. In the "Values in this mode" selector we will add 0.6, then 1.0, and then 2.0.

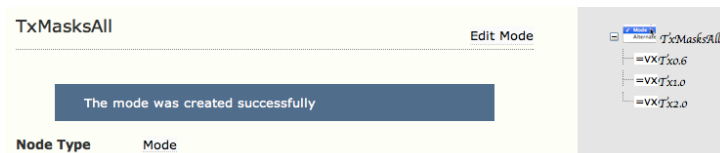
## Modes

No modes to display  
New Mode in TxMasks

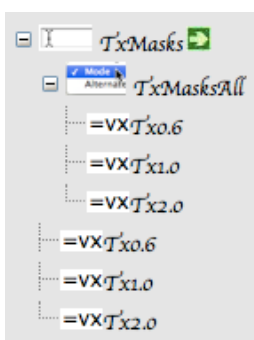
## New Mode

Name	<input type="text" value="TxMasksAll"/>
Node Type	<input type="text" value="Mode"/>
Default Value	<input type="text" value="First choose some values"/>
Default Mode	<input type="text" value="First add some sub_mod"/>
Sub Modes	<input type="text" value="First choose some system"/>
Super Modes	<input type="text" value="First choose some super"/>
Systems to include in:	<input type="text" value="TxMasks"/>
	<input type="button" value="Add System"/>
Values in this mode:	<div><input checked="" type="checkbox"/> Add Value <input type="checkbox"/> Tx0.6 <input type="checkbox"/> Tx1.0 <input type="checkbox"/> Tx2.0</div>

After that is done, let's click on "Create Mode". We will see the [Prefix]MasksAll mode details page, and we can expand the mode icon and see the values inside it.



If we navigate to the [Prefix]Masks parameter details page, we can see the tree:



So we can read "[Prefix]Masks has three possible values to take: 0.6, 1.0 and 2.0. It has only one mode available called [Prefix]MasksAll, which enables the values: 0.6, 1.0 and 2.0."

We will do an interesting step now: we will generate the GraphML only for the [Prefix]Masks subsystem, not the whole instrument. In the [Prefix]Masks details page right-click on "Generate GraphML..." link and select "Save As" and use the masks.graphml file name. Open it



on yEd and you can see this (figure on the right).

Let's repeat the process for the Dispersion parameter:

- Go to [Prefix]Dispersion details page.
- Select "New Mode..."
- Name it [Prefix]DispersionAll.
- Add R500, R1000 and R2000.

New Mode

Name: TxDispersionAll

Node Type: Mode

Default Value: First choose some values

Default Mode: First add some sub\_mode

Sub Modes: First choose some system

Super Modes: First choose some super s

Systems to include in: TxDispersion

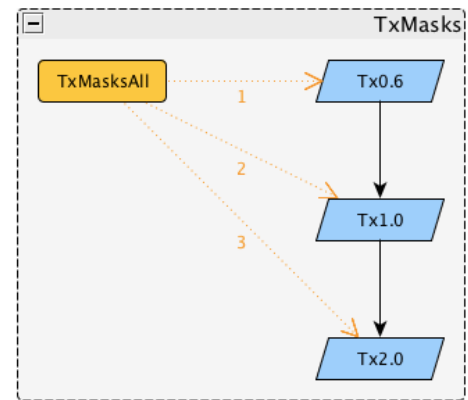
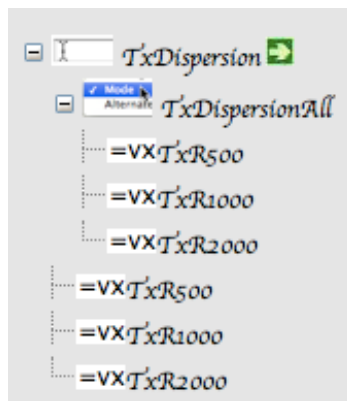
Add System

Values in this mode: TxR500, TxR1000

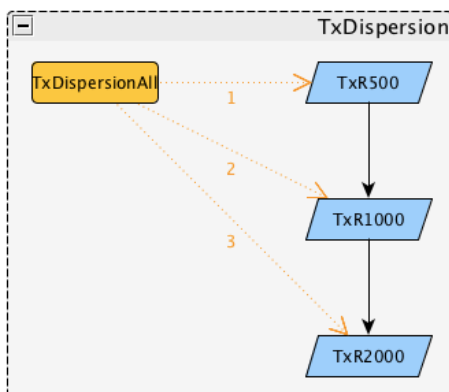
Add Value

TxR500, TxR1000, TxR2000

Check at [Prefix] Dispersion details page that the tree looks like this:



You can also go to the [Prefix]Dispersion details page and generate the GraphML diagram with the dispersion.graphml file name. Check if it looks like this:



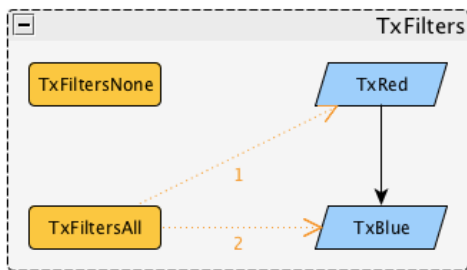
Ok, now let's continue doing real funny things: a parameter that has more than one mode. Go to the [Prefix]Filters details page and click on "New Mode". Call the new mode "[Prefix]FiltersAll" and add Red and Blue values (the same thing you did with masks and dispersion).

After that, return to the [Prefix]Filters details page and add another mode called "[Prefix]FiltersNone". Do not add any value to that mode, just click "create mode". If you go to the [Prefix]Filter details page you will find a tree like this:

This can be read as: Filters has these possible values: Red and Blue. It also has two modes: All (that enables Red and Blue values) and None, that enables nothing".

Let's generate the GraphML diagram for [Prefix]Filters parameter and save it as filters.graphml. You will obtain something similar to this:



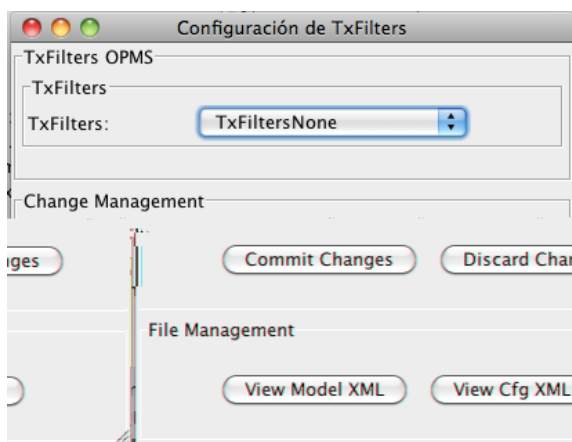


And now the amazing step where we will generate a configuration panel without coding a line!

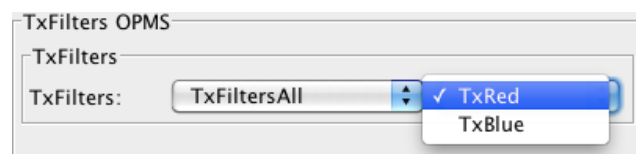
Download the PorisPlayer utility from <http://www.elporis.com/downloads> , unzip it to a folder and localize the gatATACPoris.jar file.

In the [Prefix]Filters details page right-click on “Generate Tree Code...” and name it “instrument.xml”.

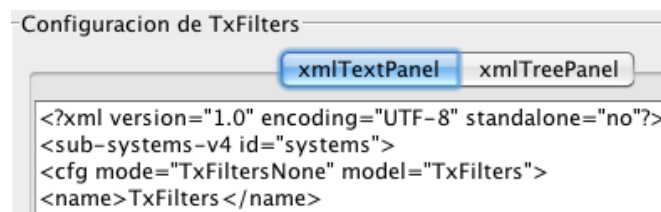
Put this instrument.xml file in the same directory where gatATACPoris.jar is and execute it using the command “java -jar gatATACPoris.jar”. It will look for the instrument.xml file and, if you click the ‘Open it!’ button it show you a panel like this:



Try to choose between [Prefix]FiltersAll and [Prefix]FiltersNone and see how the Red/Blue choices appear and disappear.



Choose None and press on “ViewConfigXML”, you will see a XML showing the current status of the panel. Only one <cfg> tag exists and it has the mode=”[Prefix]FiltersNone” attribute.



If you choose All you will see this <cfg> change to have two attributes: mode=”[Prefix]FiltersAll” and value=”Red”. If you choose Blue from the value selector then you will read mode=”[Prefix]FiltersAll” and value=”Blue”.

```
<sub-systems-v4 id="systems">
<cfg mode="TxFiltersAll" model="TxFilters" value="TxRed">
```

```
<sub-systems-v4 id="systems">
<cfg mode="TxFiltersAll" model="TxFilters" value="TxBlue">
```

But, don’t you think [Prefix]FiltersAll is an ugly name for the user? Let’s use Labels to fix this. First of all, close the gatATACPoris.jar application.



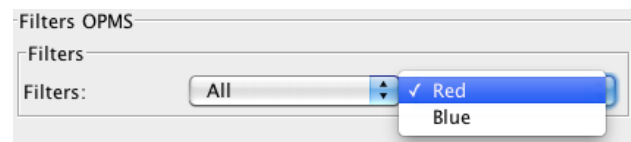
- Go to [Prefix]Filters and add a label called “Filters”, by clicking “New Label ...” link, typing the label in the name field and selecting OPMS scope kind from the selector. OPMS is not the best name, but for historical reasons this name is still present in the project and in the future it will be changed to a more suitable one.
- Go to [Prefix]FiltersAll and add a label called “All” as you did in the step before.
- Go to [Prefix]FiltersNone and add a label called “None” in the same way.
- Go to [Prefix]Red and add a label called “Red”.
- Go to [Prefix]Blue and add a label called “Blue”.

New Label  
For: TxFilters

Name   
Scope Kind

Now you can navigate to the [Prefix]Filters details page and generate the tree code again (right-clicking, as before, the “Generate Tree Code...” link and saving it as instrument.xml in the same directory the gatATACPoris.jar file is present. Restart that application using the “java -jar gatATACPoris.jar”.

You will see that your configuration panel gets friendlier.



NOTE: If you try to generate a new version of GraphML diagram you will find no differences between the one before and the one after adding labels. Labels are not considered part of the instrument architecture, are considered as modifiers for the automatic code generators.

Here you can spend some time making another OPMS labels before continuing:

- [Prefix]TestInstrument -> “Test Instrument”.
- [Prefix]Masks -> “Masks”
- [Prefix]MasksAll -> “All”
- [Prefix]0.6 -> “0.6 arcsecs”
- [Prefix]1.0 -> “1.0 arcsecs”
- [Prefix]2.0 -> “2.0 arcsecs”
- [Prefix]Dispersion -> “Dispersion”
- [Prefix]DispersionAll -> “All”
- [Prefix]R500 -> “R500”
- [Prefix]R1000 -> “R1000”
- [Prefix]R2000 -> “R2000”
- [Prefix]Detector -> “Detector”
- [Prefix]Binning -> “Binning”
- [Prefix]1x1 -> “1x1”
- [Prefix]2x1 -> “2x1”
- [Prefix]1x2 -> “1x2”
- [Prefix]2x2 -> “2x2”
- [Prefix]ExpTime -> “Exposure Time”

Until now, we have only used the first feature of the mode concept: it restricts the values that the parameter may take. Let’s explore the second one: it restricts the modes that the children subsystems may take. The next step is: build the modes of [Prefix]Binning, [Prefix]ExpTime, [Prefix]Detector and link them.

- Go to [Prefix]Binning and add a new mode call '[Prefix]BinningAll'. Add all the binning values to that mode: 1x1, 2x1, 1x2, 2x2.
- Add an OPMS label to [Prefix]BinningAll called "All".
- Go to [Prefix]ExpTime and add a new mode called "[Prefix]ExpTimeNormal". Add the [Prefix]ExpTimeRange value to it.
- Add an OPMS label to [Prefix]ExpTimeNormal called "Normal".
- Go to [Prefix]Detector and add a new mode called "[Prefix]DetectorNormal". Add the sub-modes [Prefix]ExpTimeNormal and [Prefix]BinningAll to it.
- Add an OPMS label to [Prefix]DetectorNormal called "Normal".

**New Mode**

**Name** TxDetectorNormal

**Node Type** Mode

**Default Value** First choose some values

**Default Mode** First add some sub\_modes

**Sub Modes** TxExpTimeNormal

**Super Modes**

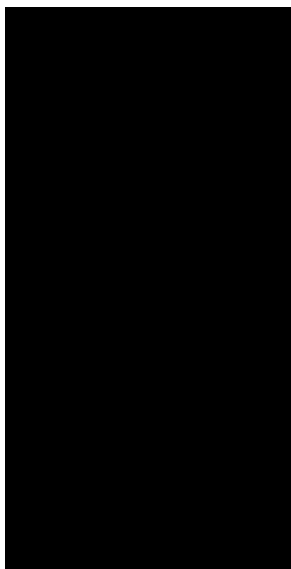
**Systems to include in:** TxDetector

**Values in this mode:** First choose some super sys

Add Sub Mode  
TxBinningAll  
TxExpTimeNormal

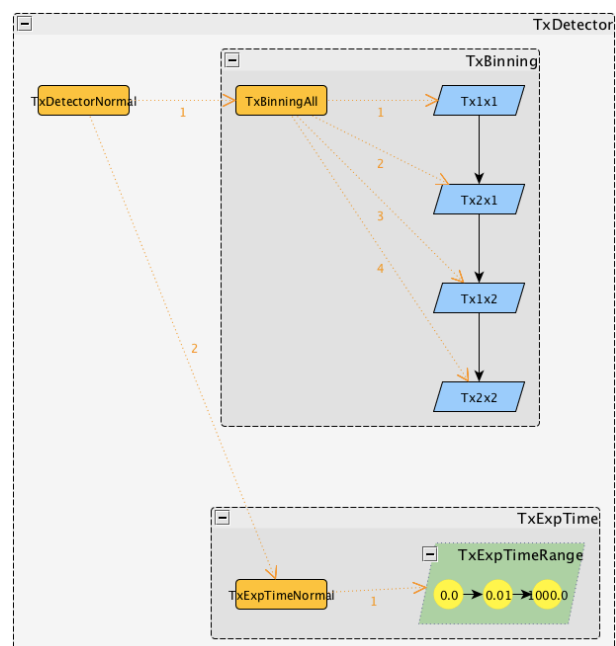
Add System

Go to the [Prefix]Detector details page and check if your tree looks like this:



This can be read as: "Detector has two parameters, binning and exptime. Binning has a mode called All, and ExpTime has a mode called Normal. Both modes enable all the possible values of their parameters. Detector has a mode called Normal that enables Binning's All and ExpTime's normal."

If you generate the GraphML representation for [Prefix]Detector into detector.graphml file you must obtain something similar to this (right figure).



And if you generate the tree code of [Prefix]Detector, name it instrument.xml and re-execute the gatATACPoris application you will obtain these panels:

**Detector OPMS**

Detector

Binning: 1x1

Exposure Time(s): 0.01

But, why can't I choose the detector, binning or exposure time modes?

- Detector: you have only one mode possible, so detector is already in "Normal" mode (see the XML Cfg window) and it will not show a selector if there is no choice possible.

- Binning: DetectorNormal forces Binning to be in All mode, so as no choice is possible no selector is shown. You can check the XML Cfg window to see it is already in this mode.
- ExpTime: the same reason.

In the SPIE's PORIS article (link to [article](#)), in figure 8 we can see some modes inside the Detector's architecture:

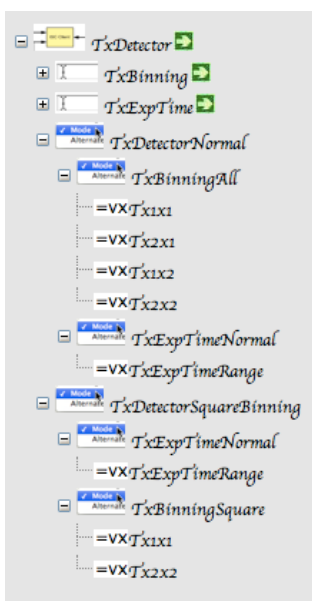
- Binning as an "Square" mode that only enables 1x1 and 2x2 values.
- Detector has a new "Square Binning" mode that enables the ExpTimeNormal mode and the BinningSquare mode.

Before introducing these changes, let's stop for a moment and try to read and understand the behavior this figure is showing to us:

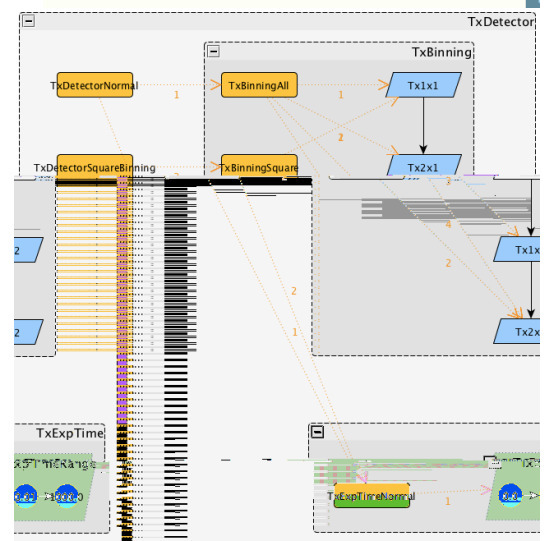
*"At the detector level we will be able to choose between Normal or Square modes. If we choose Normal, then the expTime range will be normal ([0.0...0.01...1000.0]) and the binning possible values will be 1x1, 2x1, 1x2 and 2x2. We will not be able to select submodes neither in ExpTime nor in Binning, because each Detector mode forces only one mode option for each subsystem. DetectorNormal forces BinningAll and ExpTimeNormal and DetectorSquare forces BinningSquare and ExpTimeNormal."*

- Go to [Prefix]Binning, add a new mode called "[Prefix]BinningSquare" and add the 1x1 and 2x2 values to it.
- Add an OPMS label to [Prefix]BinningSquare called "Square".
- Go to [Prefix]Detector, add a new mode called "[Prefix]DetectorSquareBinning", and add the submodes [Prefix]ExpTimeNormal and [Prefix]BinningSquare.
- Add an OPMS label to [Prefix]DetectorSquareBinning called "Square Binning".

If you go to [Prefix]Detector details page the tree must look like this (left figure).

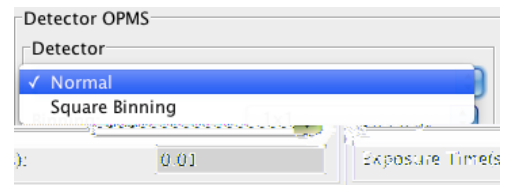


And after generating the GraphML diagram you must obtain one similar to this:



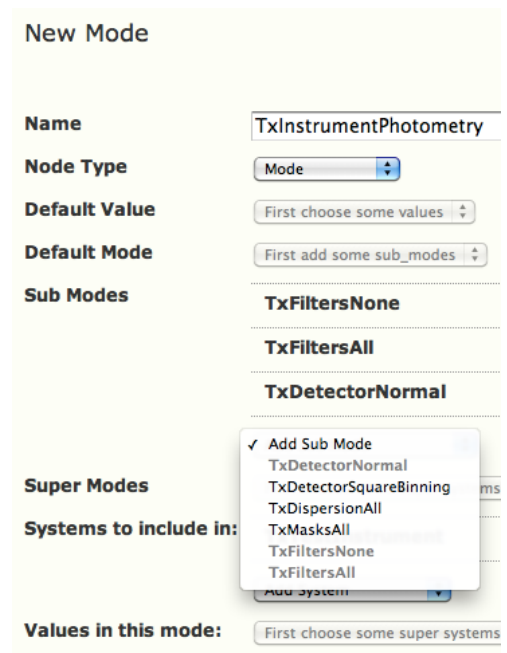
The configuration panel looks generated using “Generate Tree Code ...” looks like this:

If you choose the Normal mode, then you can choose the four values for binning, but if you choose the Square Binning mode only 1x1 and 2x2 are available.



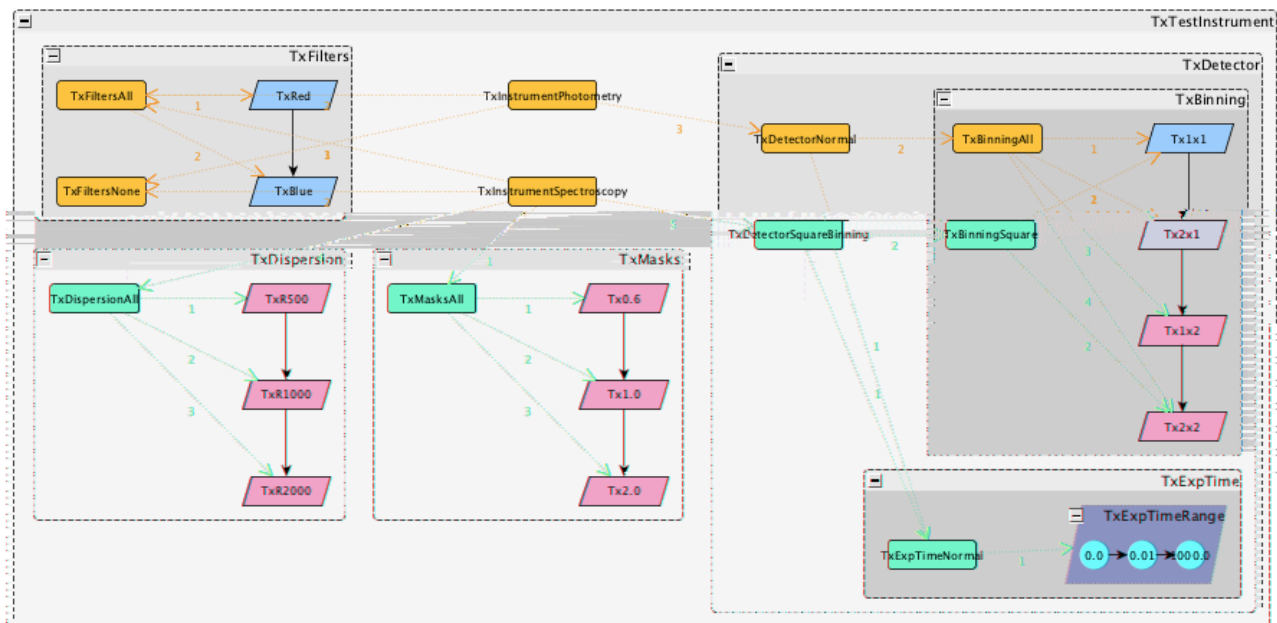
Let’s have another look of SPIE article’s figure 8 (link to [article](#)). The steps needed to obtain that system totally modelled in our web editor are:

- Go to the [Prefix]TestInstrument details page.
- Add a mode called [Prefix]InstrumentPhotometry, and add the following submodes: [Prefix]FiltersNone, [Prefix]FiltersAll, [Prefix]DetectorNormal. Dispersion and Masks will not be enabled in PhotometryMode, so we will not add any of their modes. The ExpTime and Binning modes are not available for adding, because those subsystems are not children of Instrument, they are two levels below. The modes can only affect children subsystems’ modes.
- Add an OPMS label to [Prefix]InstrumentPhotometry and call it Photometry.
- Add a mode to [Prefix]TestInstrument called [Prefix]InstrumentSpectroscopy, and add to it the following submodes: [Prefix]MasksAll, [Prefix]FiltersNone, [Prefix]FiltersAll, [Prefix]DispersionAll, [Prefix]DetectorSquareBinning.
- Add an OPMS label to [Prefix]InstrumentSpectroscopy called “Spectroscopy”.



Let’s stop again and note one interesting fact: both instrument modes are enabling more than one Filters’ modes! How will this affect to the instrument’s behavior? The answer is: when a subsystem’s mode enables more than one mode of the same child subsystem, a selector will appear at child’s level allowing the user to select between the enabled child modes.

Let’s continue to see that: if you go to [Prefix]TestInstrument and generate a GraphML diagram you must obtain something similar to this (we have manually arranged the boxes before making the snapshot):



And if you generate the instrument.xml file from the [Prefix]TestInstrument details page the configuration panels must look like this:

Congratulations!!! You have your first instrument sketch fully operative and now you can send to your team an interactive configuration panel to play with it!!!

Next steps: Read the article's section 5.4 and try to modify your instrument to reflect these changes. Use a clock to measure the time you spend for implementing that modification and be able to send a new feedback to your team.

Thanks for your attention.

Tx.