

# Dynamic UI Generation Guide

---

**Version 3.3**

PSL Team

11/29/2011

## Dynamic UI Generation

### Contents

Purpose .....	3
Introduction .....	3
Details .....	4
Examples .....	8
1) Tune Receiver.....	8
2) Chart Recorder .....	11
3) Spectral Line Display .....	20
4) Pulsar Display .....	26

## Purpose

This document contains the details about how to generate UI dynamically just by modifying few xml files. It illustrates an example each wherever dynamic UI has been generated in CMS application.

## Introduction

For generating a UI dynamically we basically need 2 files

1. The template xml file which contains the UI elements. In most cases only this file needs to be changed to add new control on UI.

2.

Following is the list of templates present in CMS for various screens

a) Receiver setup files

- ✓ continuum.xml
- ✓ spectral-line.xml
- ✓ pulsar.xml
- ✓ planetary.xml
- ✓ sun-moon.xml

b) ChartRecorder.xml – Template xml to generate the UI for chart recorder.

c) servoTrendPlot.xml – Template xml to generate the UI for the trend plot

d) SpectralLineDisplay.xml – Template xml to generate the UI for the spectral plot

e) PulsarDisplay.xml – Template xml to generate the UI for the pulsar plot

3. The xsl file which is used to transform template xml into HTML format. This file will not need any change most of the time. Server restart is needed for change in this file to reflect.

Following is the list of xsl files already used in CMS to generate UI

a) htmlgenerator.xsl – xsl used to generate Tune Receiver UI

b) ChartRecorder.xml – xsl used to process ChartRecorder.xml file and generate UI for chart recorder.

c) servoTrendPlot.xml – xsl used to process ChartRecorder.xml file and generate UI for trend Plot preferences

d) SpectralLineDisplay.xml – xsl used to process SpectralLineDisplay.xml file and generate UI for Spectral Line Display preferences

e) PulsarDisplay.xml – xsl used to process PulsarDisplay.xml file and generate UI for Pulsar Display preferences

## Details

Let us start explaining the generation of dynamic UI with the help of an example.

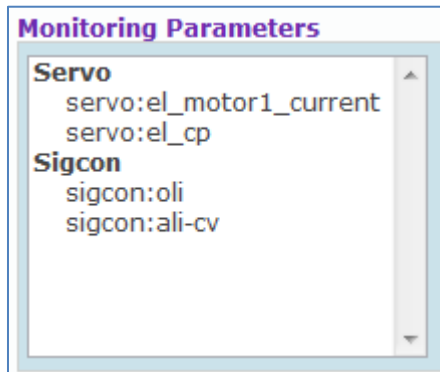


Figure 1

The figure shown above contains a select box. For generating this UI dynamically we have to make a template xml file of the UI elements and an xsl file which will transform the template xml file into the HTML format.

## Dynamic UI Generation

The template xml for the above UI will look like this:

```
<group name="Monitoring Parameters">
  <controls>
    <row>
      <column>
        <control>
          <name>MoniParam_multipleSelct</name>
          <type>MultipleSelectList</type>
          <colspan>4</colspan>
          <cmdid>MoniParam_multipleSelct</cmdid>
          <maxLength>10</maxLength>
          <size>10</size>
          <style>WIDTH:200px</style>
          <label></label>
          <optgroup>
            <label>Servo</label>
            <option>
              <key>servo:el_motor1_current</key>
              <value>servo:el_motor1_current</value>
              <selected>>false</selected>
            </option>
            <option>
              <key>servo:el_cp</key>
              <value>servo:el_cp</value>
              <selected>>false</selected>
            </option>
          </optgroup>
          <optgroup>
            <label>Sigcon</label>
            <option>
              <key>sigcon:oli</key>
              <value>sigcon:oli</value>
              <selected>>false</selected>
            </option>
            <option>
              <key>sigcon:ali-cv</key>
              <value>sigcon:ali-cv</value>
              <selected>>false</selected>
            </option>
          </optgroup>
        </control>
      </column>
    </row>
  </controls>
</group>
```

Figure 2

## Dynamic UI Generation

Let us start explaining each node one by one.

1. **Groups** – It contains the list of groups present under a section. The attribute available with node is 'templateName' where user can mention the name of the template. It's useful in some cases however; it's not used as of now.
2. **Group** – It groups various HTML elements to form a section, so to add a new group related to new subsystem, one need to add a new group section to template xml. The supported attribute is on this node is "name".
3. **Controls** – One section can contain various HTML controls. The list of these controls is provided in this. It contains 4 attributes i.e. type, cssClass, sectionname, rowspan.
4. **Row** - Represents a line on GUI, if user need to add a new control on new line then this tag should be added. This is synonymous <TR> element in HTML.
5. **Column** - Represents a cell on UI, this acts as a container of actual UI elements (e.g. Textbox, radio button etc.). This is synonymous <TD> element in HTML.
6. **Control** - Represents actual UI element on the screen, user can configure various properties for control depending upon the type of control chosen, following are various configurable properties of this element.
  - a. **Disabled** - Used to disable a control on UI, so that by default user can't change default values for a preconfigured template.
  - b. **type** - Used to configure type of element to be added on UI, following are various values
    - i. **radio** – adds a radio button control on UI
    - ii. **checkbox** – adds a checkbox element on UI
    - iii. **Text** – adds a textbox element on UI
    - iv. **Dropdown** – adds a dropdown element on UI
    - v. **MultipleSelectList** – adds a multiple select dropdown element on UI
  - c. **name** – Actual name of the control corresponding to element on UI.
  - d. **cmdid** – This is same as "name" field mentioned and is kept their for future use (if needed), however, this might be removed in future
  - e. **label** – Used to specify the label of the UI element (e.g. – APERTURE)
  - f. **value** – Used to specify default value for UI element.
  - g. **subsystemName** – Used to specify the name of the subsystem associated with element
  - h. **options** – For controls which have range of valid values (e.g. position values) we use dropdown control. For such controls we use Dropdown on UI. This property is used to configure various values.
  - i. **checked** – Used to specify whether checkbox element on UI should be checked by default
  - j. **maxLength** – It is used to specify the maximum number of characters allowed in that UI element.
  - k. **onClick** – This property is used for buttons. It contains the name of the javascript function which is to be called on click of that button.
  - l. **onChange** – This property is used for dropdowns. It contains the name of the method which is to be called on change of the dropdown option.
  - m. **Tooltip** – It contains the data to be shown in the tooltip.

## Dynamic UI Generation

- n. Style – It is used to specify the style, such as height, width, background color, alignment, etc.
- o. Size – it is used to specify the number of the parameter shown in multiple select box without scroll, if a value is 5 than at-least 5 elements displayed.

So in the above UI all the controls fall under the section named 'Monitoring Parameters'. Here the controls are arranged in 2 rows. The first row contains the control of type dropdown and the second row contains 2 controls of type button arranged in 2 columns.

## Examples

Following are few examples about how to configure the UI at various places in CMS.

### 1) Tune Receiver

Following is the command configuration section for receiver setup for SIGCON system before adding a new parameter.

We will add a configuration parameter named “NEW”, which has 2 channels.

Note that NEW here is used to just indicate a new command/configuration parameter name, but it can be any command name like move, track etc.

SIGCON			
	POL1		POL2
IF PRE-ATTN:	<input type="text" value="10"/>	dB	<input type="text" value="5"/> dB
LOCAL OSC:	<input type="text" value="300"/>	MHz	<input type="text" value="400"/> MHz
SIGCON FILTER:	<input type="text" value="100"/> ▾	MHz	<input type="text" value="100"/> ▾ MHz
IF POST-ATTN:	<input type="text" value="10"/>	dB	<input type="text" value="14"/> dB
AGC:	<input checked="" type="checkbox"/>		<input type="checkbox"/>

Figure 3



## Dynamic UI Generation

Let's add following lines in continuum.xml at the end of sigcon section.

```
<row>
  <column>
    <control disabled="true">
      <name>NEW__POL1</name>
      <type>Text</type>
      <cmdid>NEW__POL1</cmdid>
      <maxLength>5</maxLength>
      <label>NEW:</label>
      <value>10</value>
      <subSystemName>sigcon</subSystemName>
      <options/>
      <unit>dB</unit>
      <metaData>>false</metaData>
    </control>
  </column>
  <column>
    <control disabled="true">
      <name>NEW__POL2</name>
      <type>Text</type>
      <cmdid>NEW__POL2</cmdid>
      <maxLength>5</maxLength>
      <label></label>
      <value>5</value>
      <subSystemName>sigcon</subSystemName>
      <unit>dB</unit>
      <metaData>>false</metaData>
    </control>
  </column>
</row>
```

Figure 4

Please notice the contents of "name" tag above, it has following format X\_\_Y,

X here represents the command name, whereas Y indicates the parameter name. Both of these should be separated by '\_\_' (double underscore). If a command has multiple parameters then one can just repeat entries of X\_\_Y1, X\_\_Y2 and so on in xml file under "column/control" nodes.

## Dynamic UI Generation

Following is effect of adding these lines on tune receiver UI.

SIGCON			
	POL1		POL2
IF PRE-ATTN:	<input type="text" value="10"/>	dB	<input type="text" value="5"/> dB
LOCAL OSC:	<input type="text" value="300"/>	MHz	<input type="text" value="400"/> MHz
SIGCON FILTER:	<input type="text" value="100"/> ▼	MHz	<input type="text" value="100"/> ▼ MHz
IF POST-ATTN:	<input type="text" value="10"/>	dB	<input type="text" value="14"/> dB
AGC:	<input checked="" type="checkbox"/>		<input type="checkbox"/>
NEW:	<input type="text" value="10"/>	dB	<input type="text" value="5"/> dB

Figure 5

In order to execute this newly added command we also need to add new command in sigcon\_commands.xml –

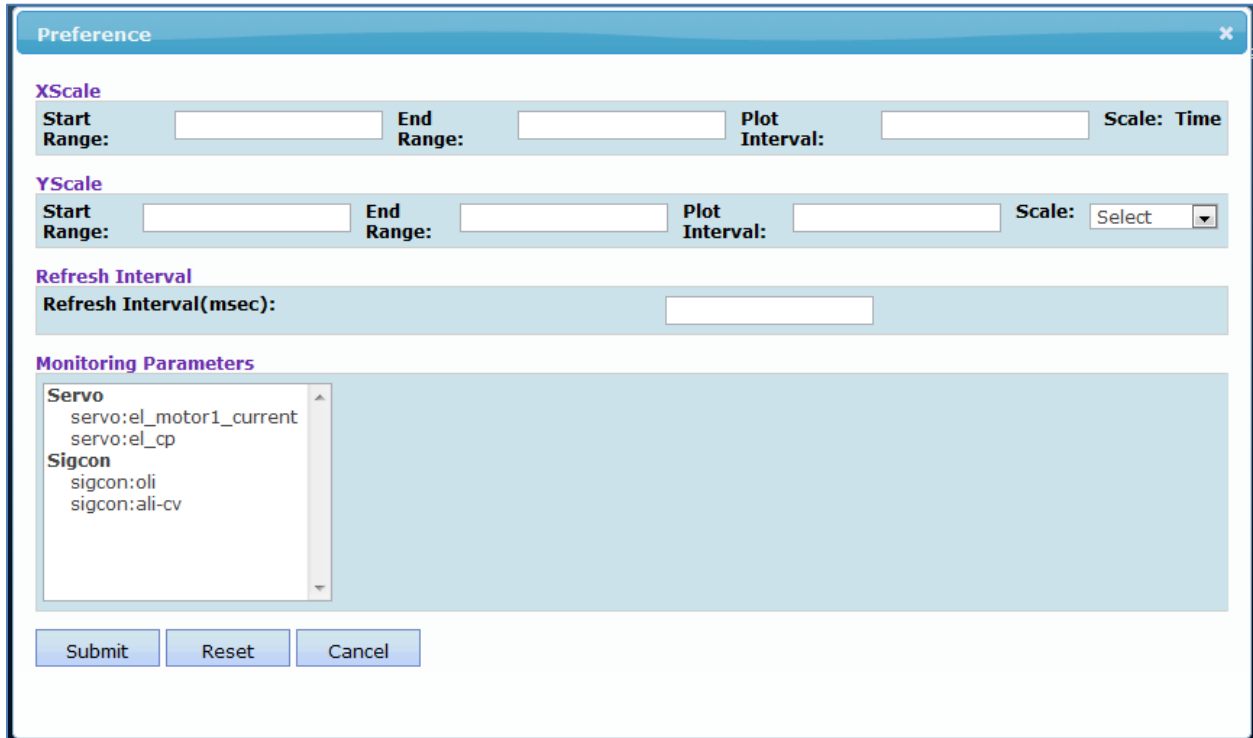
```
<command>
  <name>NEW</name>
  <id>280</id>
  <syntax>POL1,POL2</syntax>
  <sample>1,2</sample>
  <timeout>120000</timeout>
  <params>
    <param required="true">
      <paramname>POL1</paramname>
      <type>float</type>
      <validation>
        <range>
          <min>0</min>
          <max>15</max>
        </range>
      </validation>
    </param>
    <param required="true">
      <paramname>POL2</paramname>
      <type>float</type>
      <validation>
        <range>
          <min>0</min>
          <max>15</max>
        </range>
      </validation>
    </param>
  </params>
</command>
```

Figure 6

## 2) Chart Recorder



Figure 7



**Preference**

**XScale**

Start Range:  End Range:  Plot Interval:  Scale: Time

**YScale**

Start Range:  End Range:  Plot Interval:  Scale:

**Refresh Interval**

Refresh Interval(msec):

**Monitoring Parameters**

Servo  
servo:el\_motor1\_current  
servo:el\_cp  
Sigcon  
sigcon:oli  
sigcon:ali-cv

Submit Reset Cancel

Figure 8

The chart recorder Preference screen shot is dynamically generated. The files which are used for generating chart recorder Preference dynamically is ChartRecorder.xml (template xml file) and ChartRecorder.xsl (xsl file).

User can add a new section similar to that of Monitoring Parameters or filter just by changing the template xml file. Steps for adding a new section named 'New Monitoring Parameters' are as follows.

1. Copy the complete 'controls' node of Monitoring Parameters from ChartRecorder.xml file shown in Figure 2.
2. Paste it just below the closing tag of 'controls' node of 'MoniParam\_multipleSelect' as shown in Figure 9.

## Dynamic UI Generation

```

<control>
  <name>New_MoniParam_multipleSelect</name>
  <type>MultipleSelectList</type>
  <colspan>4</colspan>
  <cmdid>MoniParam_multipleSelct</cmdid>
  <maxLength>10</maxLength>
  <size>10</size>
  <style>WIDTH:200px</style>
  <label></label>
  <optgroup>
    <label>Sigcon</label>
    <option>
      <key>sigcon:oli</key>
      <value>sigcon:oli</value>
      <selected>>false</selected>
    </option>
    <option>
      <key>sigcon:ali-cv</key>
      <value>sigcon:ali-cv</value>
      <selected>>false</selected>
    </option>
  </optgroup>
</control>
</column>
</row>
</controls>
</group>

```

Figure 9

3. Populate various values for newly added "New\_MoniParam\_multipleSelect", we need to add corresponding entry in ChartRecorder.xml file as well. For this just copy existing "<control>" node of 'MoniParam\_multipleSelect' and change few attributes highlighted below in figure 10. Following are the nodes to be modified
  - a. <Name> – Set name as any name like New\_MoniParam\_multipleSelect
  - b. <Size> - At a time how many parameters you want to display without scroll bar.
  - c. <Style> - Here user can set the width, height, background color etc.
  - d. <optgroup> - This very important section which contains multiple <options> where you can specify the monitoring parameters, here configuring two monitoring parameters, sigcon:oli, sigcon:ali-cv.

# Dynamic UI Generation

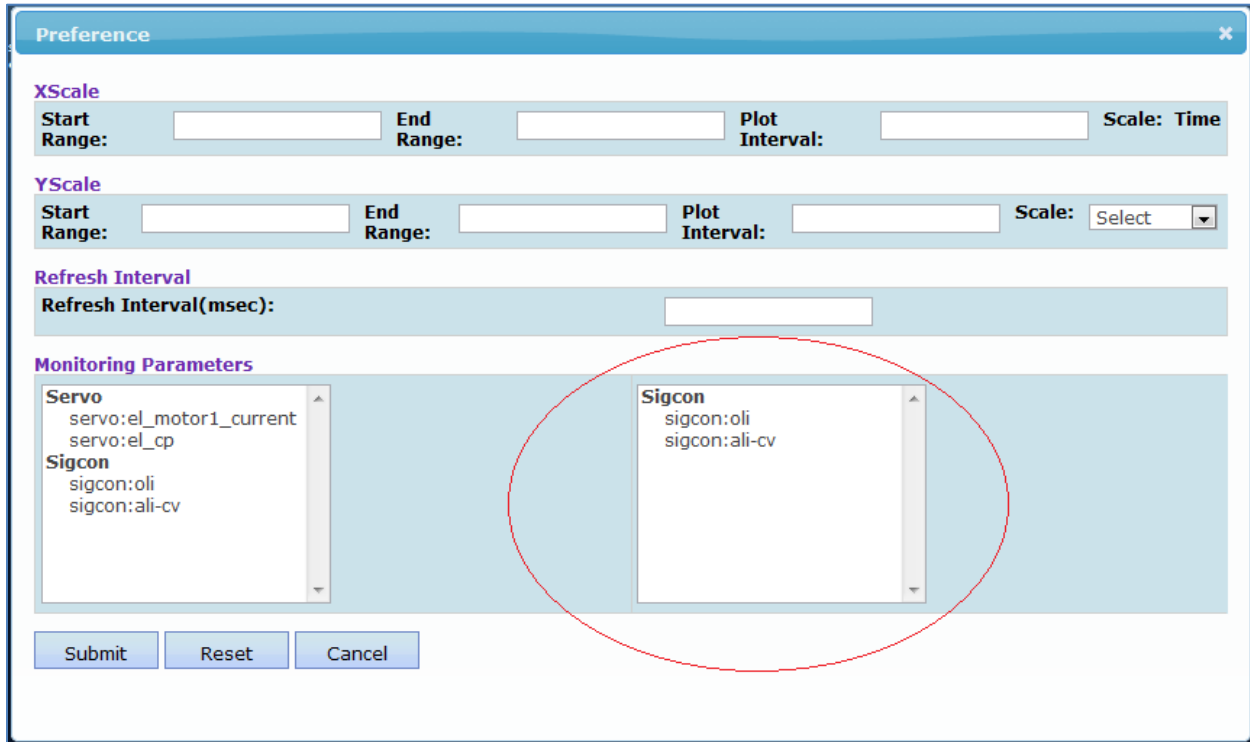
```

<control>
  <name>New_MoniParam_multipleSelct</name>
  <type>MultipleSelectList</type>
  <colspan>4</colspan>
  <cmdid>MoniParam_multipleSelct</cmdid>
  <maxLength>10</maxLength>
  <size>10</size>
  <style>WIDTH:200px</style>
  <label></label>
  <optgroup>
    <label>Sigcon</label>
    <option>
      <key>sigcon:oli</key>
      <value>sigcon:oli</value>
      <selected>false</selected>
    </option>
    <option>
      <key>sigcon:ali-cv</key>
      <value>sigcon:ali-cv</value>
      <selected>false</selected>
    </option>
  </optgroup>
</control>
</column>
</row>
</controls>
</group>

```

Figure 10

4. With above changes ui looks like,



**Preference**

**XScale**  
Start Range:  End Range:  Plot Interval:  Scale: Time

**YScale**  
Start Range:  End Range:  Plot Interval:  Scale:

**Refresh Interval**  
Refresh Interval(msec):

**Monitoring Parameters**

**Servo**  
servo:el\_motor1\_current  
servo:el\_cp

**Sigcon**  
sigcon:oli  
sigcon:ali-cv

**Sigcon**  
sigcon:oli  
sigcon:ali-cv

Submit Reset Cancel

Figure 11

5. Now Add new label and Text box into existing UI
6. Open ChartRecorder.xml now look into refresh interval section as show in below

```

<group name="Refresh Interval">
  <controls>
    <row>
      <column>
        <control>
          <name>Refresh_Interval_Label</name>
          <type>LABEL</type>
          <cmdid>Refresh_Interval_Label</cmdid>
          <label>Refresh Interval (msec) :</label>
        </control>
      </column>
      <column>
        <control class="int">
          <name>Refresh_Interval_Value</name>
          <type>Text</type>
          <cmdid>Refresh_Interval_Value</cmdid>
          <maxLength>5</maxLength>
        </control>
      </column>
    </row>
  </controls>
</group>

```

Figure 12

- a. Copy above both <control> and pest it below it look like,



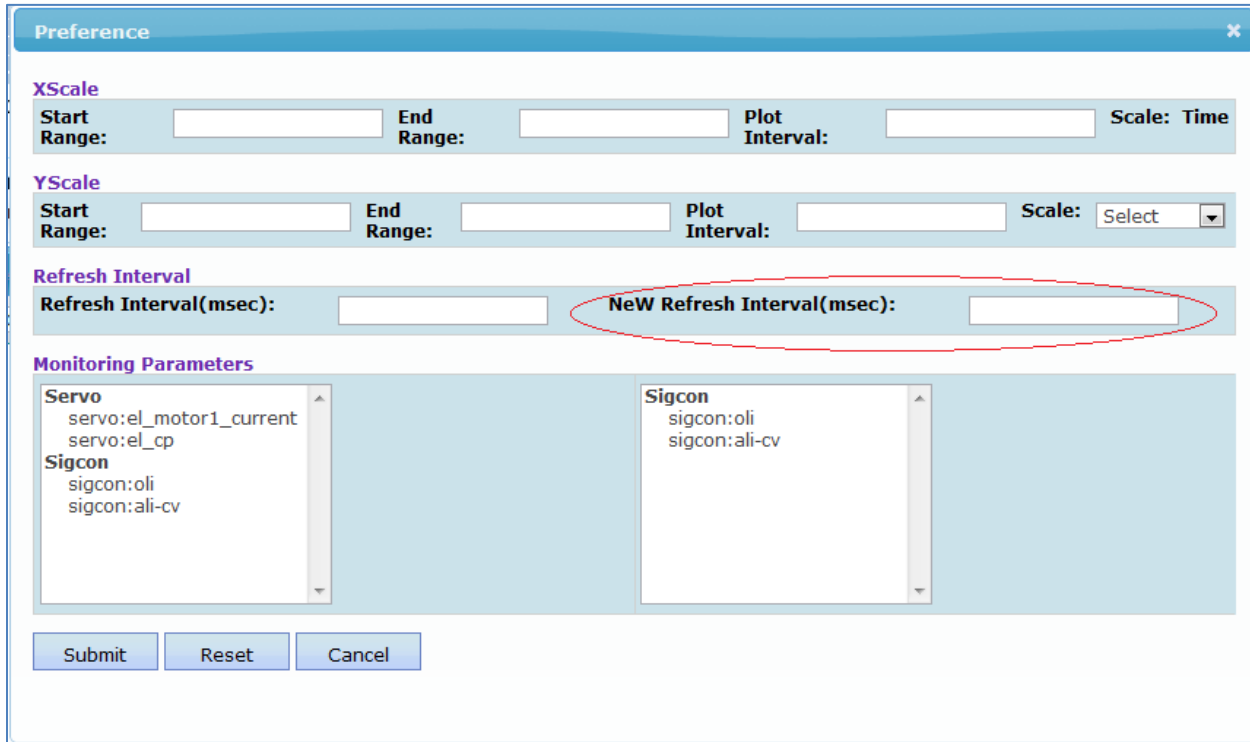


```
<group name="Refresh Interval">
  <controls>
    <row>
      <column>
        <control>
          <name>Refresh_Interval_Label</name>
          <type>LABEL</type>
          <cmdid>Refresh_Interval_Label</cmdid>
          <label>Refresh Interval (msec) :</label>
        </control>
      </column>
      <column>
        <control class="int">
          <name>Refresh_Interval_Value</name>
          <type>Text</type>
          <cmdid>Refresh_Interval_Value</cmdid>
          <maxLength>5</maxLength>
        </control>
      </column>
    </row>
    <row>
      <column>
        <control>
          <name>RefreshIntervalLabel</name>
          <type>LABEL</type>
          <cmdid>RefreshIntervalLabel</cmdid>
          <label>New Refresh Interval (msec) :</label>
        </control>
      </column>
      <column>
        <control class="int">
          <name>NewRefreshIntervalValue</name>
          <type>Text</type>
          <cmdid>NewRefreshIntervalValue</cmdid>
          <maxLength>5</maxLength>
        </control>
      </column>
    </row>
  </controls>
</group>
```

Figure 13

- b. Now notice the first <control> type is LABEL and second control type is Text, change the attributes <name> and <cmdid> attributes.

c. Now update UI looks like as below,



The image shows a 'Preference' dialog box with the following sections:

- XScale**: Contains fields for 'Start Range:', 'End Range:', 'Plot Interval:', and 'Scale: Time'.
- YScale**: Contains fields for 'Start Range:', 'End Range:', 'Plot Interval:', and 'Scale: Select' (with a dropdown arrow).
- Refresh Interval**: Contains fields for 'Refresh Interval(msec):' and 'NeW Refresh Interval(msec):'. The 'NeW Refresh Interval(msec):' field is circled in red.
- Monitoring Parameters**: Contains two lists:
  - Servo**: servo:el\_motor1\_current, servo:el\_cp
  - Sigcon**: sigcon:oli, sigcon:ali-cv

At the bottom, there are three buttons: 'Submit', 'Reset', and 'Cancel'.

Figure 14

**Note & Assumptions:**

- Existing ChartRecorder.xml contains 4 section as below,
  - o XScale –
    - All the values Start Range, End Rand, Plot Interval, Scale name must be start with Xscale\_
    - If dynamically add any component than please start <name> <cmdid> values must be start with Xscale\_
    - <name> and <cmdid> nodes values are used in java script and other java implementation so don't change these value if user is changing these node values than need to modify the code according.
  - o YScale –
    - All the values Start Range, End Rand, Plot Interval, Scale name must be start with Yscale\_
    - If dynamically add any component than please start <name> <cmdid> values must be start with Yscale\_

## Dynamic UI Generation

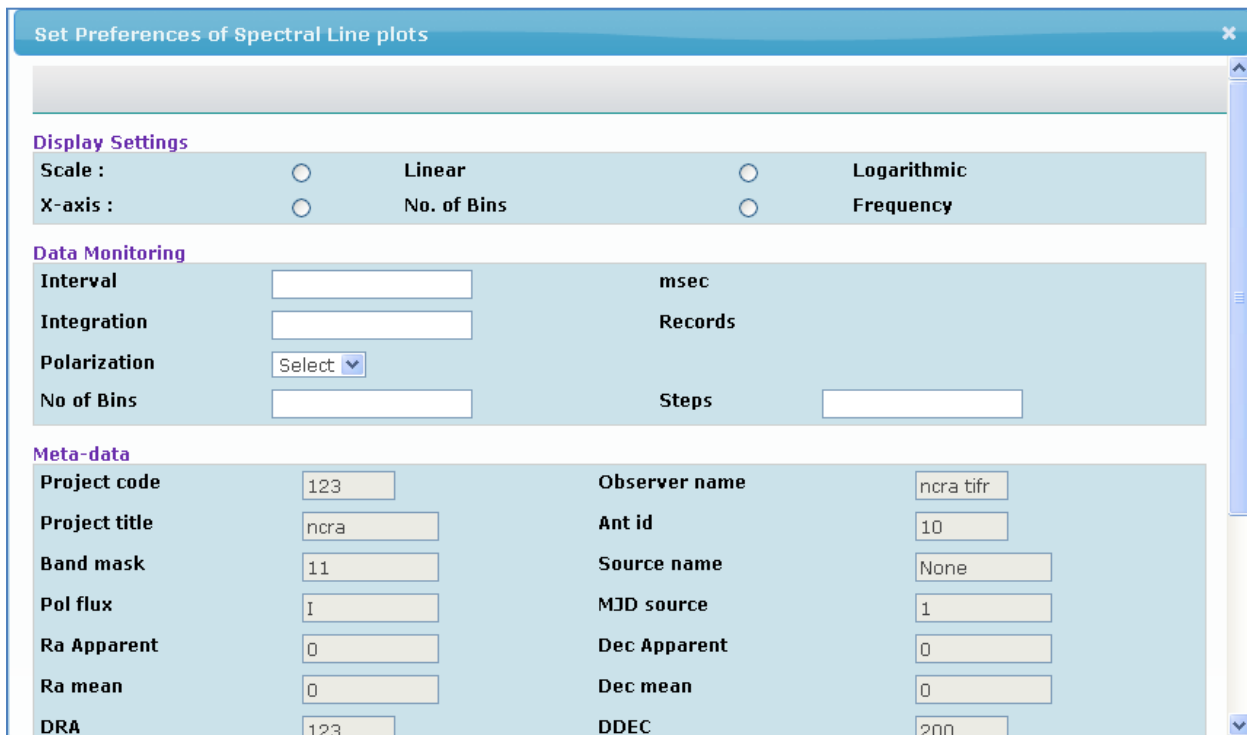
- <name> and <cmdid> nodes values are used in java script and other java implementation so don't change these value if user is changing these node values than need to modify the code according.
- Monitoring Parameters –
  - If dynamically add any component than please start <name> <cmdid> values must be start with MoniParam\_

### 3) Spectral Line Display



Figure 15

## Dynamic UI Generation



**Set Preferences of Spectral Line plots**

**Display Settings**

Scale : ☐ Linear ☐ Logarithmic

X-axis : ☐ No. of Bins ☐ Frequency

**Data Monitoring**

Interval  msec

Integration  Records

Polarization

No of Bins  Steps

**Meta-data**

Project code	<input type="text" value="123"/>	Observer name	<input type="text" value="ncra tifr"/>
Project title	<input type="text" value="ncra"/>	Ant id	<input type="text" value="10"/>
Band mask	<input type="text" value="11"/>	Source name	<input type="text" value="None"/>
Pol flux	<input type="text" value="1"/>	MJD source	<input type="text" value="1"/>
Ra Apparent	<input type="text" value="0"/>	Dec Apparent	<input type="text" value="0"/>
Ra mean	<input type="text" value="0"/>	Dec mean	<input type="text" value="0"/>
DRA	<input type="text" value="123"/>	DDEC	<input type="text" value="200"/>

Figure 16

The spectral line display Preference screen shot is dynamically generated. The files which are used for generating chart recorder Preference dynamically is SpectralLineDisplay.xml (template xml file) and SpectralLineDisplay.xsl (xsl file).

User can add new parameters by adding the controls in template xml and making the entry for the same parameter in backend-commands.xml.

- Steps to add a control say radio button name **"new\_scale"** with values say **"new\_linear"** and **"new\_logarithmic"** in "Display Settings" section of template.xml.

Example:

User wants to add two radio controls.

Following conventions must be followed while setting the attributes of radio button.

- <type>**: Type will be **"radio"**

## Dynamic UI Generation

- b. **<name>** : Name of all radio buttons in a single radio group must be same. In this example we have added two new radio buttons. Name of radio button must start with **“sendspectralplottingdata\_\_”** followed by its name.  
Example : sendspectralplottingdata\_\_new\_scale
- a. **<cmdid>**: Cmdid of the control must start with **“sendspectralplottingdata \_\_”** followed by its name.  
Example : sendspectralplottingdata\_\_Linear
- c. **<label>** : Label can be any valid name.  
Example : New Linear
- d. **<value>**: Value will be the one which user needs to send to backend system when that particular radio button will be selected.  
Example : new\_linear
- e. **<checked>**: true/false depending upon user wants the radio button to be by default checked/unchecked.



```
<row>
  <column>
    <control>
      <name>Spectral_Scale</name>
      <type>LABEL</type>
      <cmdid>Spectral_Scale</cmdid>
      <label>New Scale :</label>
    </control>
  </column>
  <column>
    <control>
      <type>radio</type>
      <name>sendspectralplottingdata__new_scale</name>
      <cmdid>sendspectralplottingdata__Linear</cmdid>
      <label>New Linear</label>
      <value>new_linear</value>
      <checked>>false</checked>
    </control>

    <control>
      <name>Spectral_linear</name>
      <type>LABEL</type>
      <cmdid>Spectral_linear</cmdid>
      <label>Linear</label>
    </control>
  </column>
  <column>
    <control>
      <type>radio</type>
      <name>sendspectralplottingdata__new_scale</name>
      <cmdid>sendspectralplottingdata__Logarithmic</cmdid>
      <label>Logarithmic</label>
      <value>new_logarithmic</value>
      <checked>>false</checked>
    </control>

    <control>
      <name>Spectral_logarithmic</name>
      <type>LABEL</type>
      <cmdid>Spectral_logarithmic</cmdid>
      <label>New Logarithmic</label>
    </control>
  </column>
</row>
```

Figure 17

2. User also needs to make entry of the added radio control in “**backend\_commands.xml**” in “**sendspectralplottingdata**” command in following way.

```

<command>
  <name>sendspectralplottingdata</name>
  <id>106</id>
  <syntax>new_scale,axis,interval,integration,polarization
  <sample>new_linear,linear,noofbins,100,1,both,100,1,123,e
  <timeout>120000</timeout>
  <params>
    <param required="true">
      <paramname>new_scale</paramname>
      <type>string</type>
      <validation>
        <values>
          <value>new_linear</value>
          <value>new_logarithmic</value>
        </values>
      </validation>
    </param>
    <param required="true">
      <paramname>scale</paramname>
      <type>string</type>
      <validation>
        <values>
          <value>linear</value>
          <value>logarithmic</value>
        </values>
      </validation>
    </param>
  </params>

```

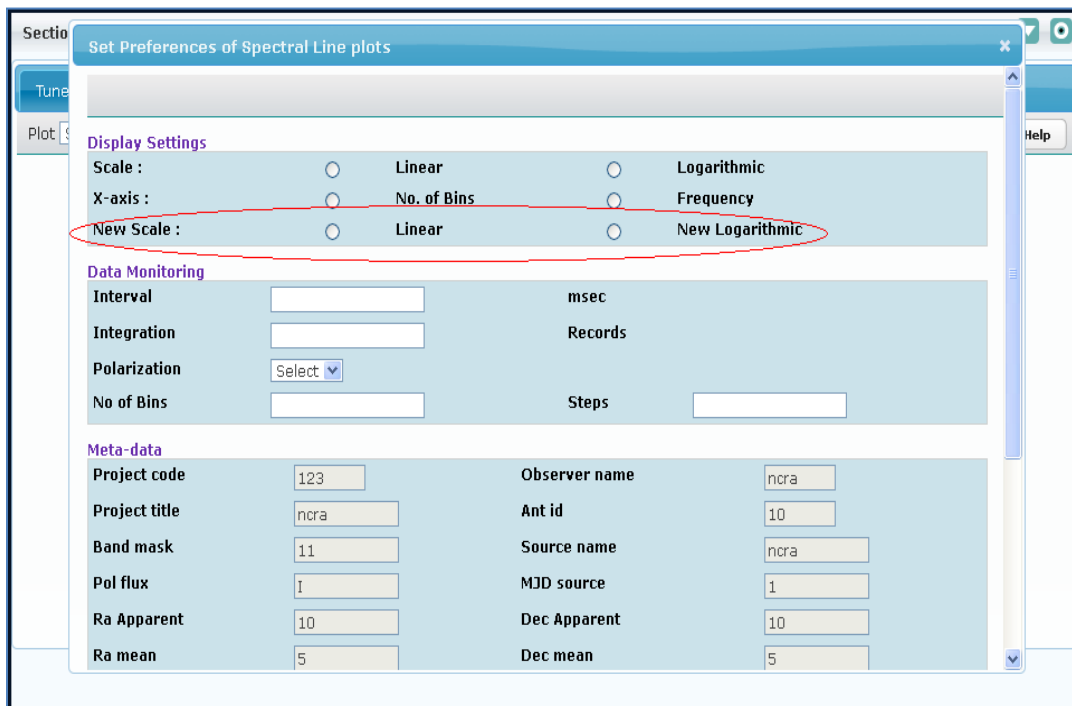
Figure 18

- a. **<syntax>** : When the radio control was added ,its name was “sendspectralplottingdata\_\_new\_scale” (refer above). Here the name of the parameter will be only “new\_scale” i.e. “sendspectralplottingdata\_\_” must be removed and the parameter must be added in the syntax.
- b. **<param>**: Add a param tag with values as mentioned in “value” of radio buttons. In this case they will be “new\_linear” and “new\_logarithmic”.
- c. **<sample>**: Add one of the values as sample value. In this case it will be “new\_linear”.



## Dynamic UI Generation

User will be able to see the new radio control added on the preferences dialog of Spectral line display.



Section: Set Preferences of Spectral Line plots

**Display Settings**

Scale : ☐ Linear ☐ Logarithmic

X-axis : ☐ No. of Bins ☐ Frequency

**New Scale :** ☐ Linear ☐ New Logarithmic

**Data Monitoring**

Interval  msec

Integration  Records

Polarization

No of Bins  Steps

**Meta-data**

Project code	<input type="text" value="123"/>	Observer name	<input type="text" value="ncra"/>
Project title	<input type="text" value="ncra"/>	Ant id	<input type="text" value="10"/>
Band mask	<input type="text" value="11"/>	Source name	<input type="text" value="ncra"/>
Pol flux	<input type="text" value="1"/>	MJD source	<input type="text" value="1"/>
Ra Apparent	<input type="text" value="10"/>	Dec Apparent	<input type="text" value="10"/>
Ra mean	<input type="text" value="5"/>	Dec mean	<input type="text" value="5"/>

Figure 19

### **Note & Assumptions:**

- Existing SpectralLineDisplay.xml contains 3 sections Display Settings, Data Monitoring and Meta data section.
- Names of all the input fields must start with “**sendspectralplottingdata\_\_**”
- <name> and <cmdid> nodes values are used in java script and other java implementation so don't change these value .If user is changing these node values than need to modify the code according.
- Meta-data section fields are all read-only and they correspond to Global parameter” bean data. User can change this data through “Expert ” or “Tune Receiver”

## 4) Pulsar Display

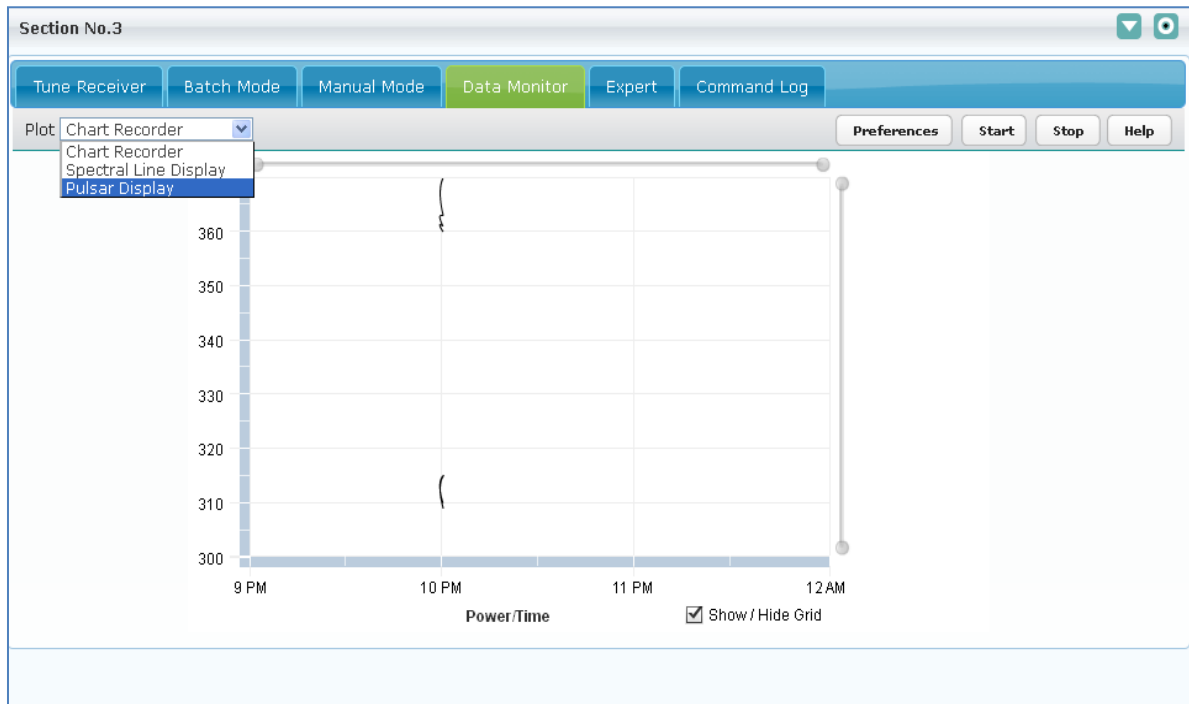


Figure 20

## Dynamic UI Generation

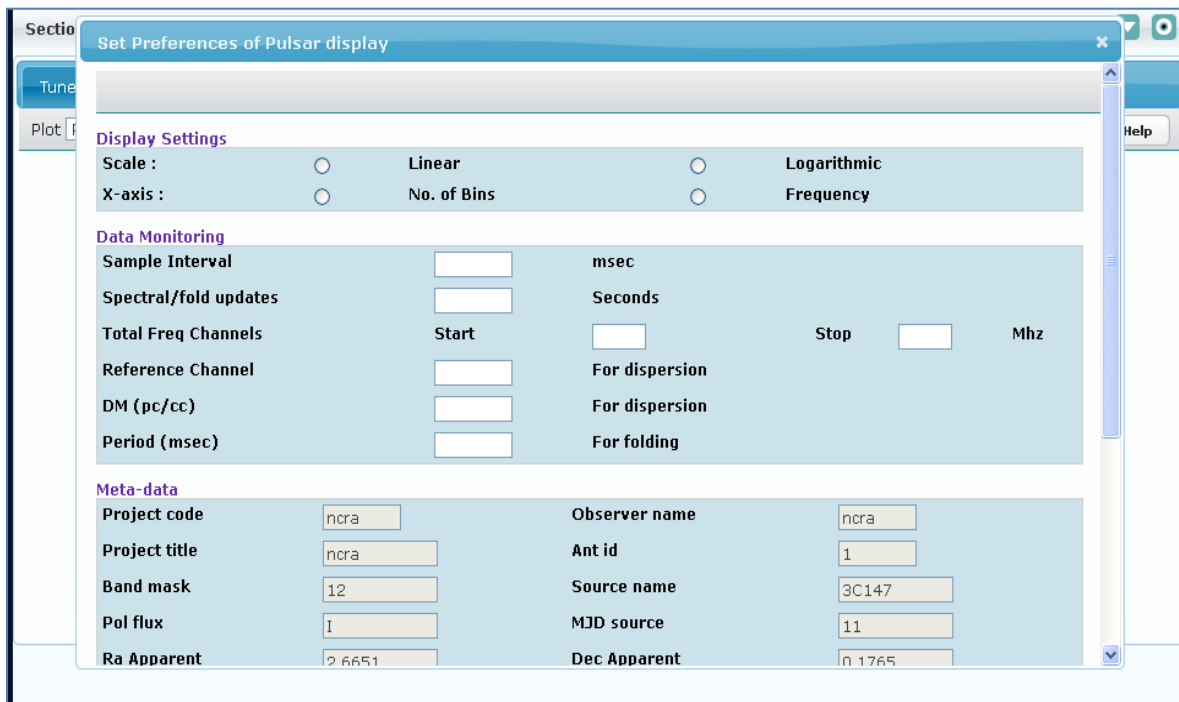


Figure 21

The pulsar display Preferences dialog is dynamically generated. The files which are used for generating chart recorder Preference dynamically is PulsarDisplay.xml (template xml file) and PulsarDisplay.xsl (xsl file).

User can add new parameters by adding the controls in template xml and making the entry for the same parameter in backend-commands.xml.

- Steps to add a control say drop down name **"sample"** with dropdown values say **"sample1"** and **"sample2"** in "Display Settings" section of template.xml .

Example:

Following conventions must be followed while setting the attributes of dropdown.

- <type>** : Type will be **"Dropdown"**
- <name>** : Name of the control must start with **"sendpulsarplottingdata\_\_"** followed by its name.  
Example : sendpulsarplottingdata\_\_sample

## Dynamic UI Generation

- d. **<cmdid>** : Cmdid of the control must start with “**sendpulsarplottingdata\_\_**” followed by its name.  
Example : sendpulsarplottingdata\_\_sample
- e. **<colspan>** : If required colspan can be specified for that particular column  
Example: <colspan>2</colspan>
- f. **<options>** : Specifies the options that will be shown in the dropdown.
- g. **<option>** : User needs to specify the key and value for each option in option tag.
- h. **<key>** : Key is the one that is displayed to user on the dropdown.
- i. **<value>** : Value is the one that is sent to backend after a particular option in dropdown is selected.  
Example : <options>  
                  <option>  
                    <key>sample1</key>  
                    <value>sample1</value>  
                  </option>  
          </options>



```
<row>
  <column>
    <control>
      <name>Sample</name>
      <type>LABEL</type>
      <cmdid>Sample</cmdid>
      <label>Sample</label>
    </control>
  </column>
  <column>
    <control class="combobox_small">
      <type>DropDown</type>
      <colspan>2</colspan>
      <cmdid>sendpulsarplottingdata__sample</cmdid>
      <name>sendpulsarplottingdata__sample</name>
      <options>
        <option>
          <key>sample1</key>
          <value>sample1</value>
        </option>
        <option>
          <key>sample2</key>
          <value>sample2</value>
        </option>
      </options>
    </control>
  </column>
</row>
```

Figure 22

2. User also needs to make entry of the added dropdown control in “**backend\_commands.xml**” in “**sendpulsarplottingdata**” command in following way.
  - a. **<syntax>** : When the dropdown control was added ,its name was “**sendpulsarplottingdata \_\_sample**” (refer above). Here the name of the parameter will be only “**sample**” i.e. “**sendpulsarplottingdata \_\_**” must be removed and the parameter must be added in the syntax.
  - b. **<param>**: Add a param tag with values as mentioned in “**value**” of dropdown options. In this case they will be “**sample1**” and “**sample2**”.
  - c. **<sample>**: Add one of the values as sample value. In this case it will be “**sample1**”.

## Dynamic UI Generation

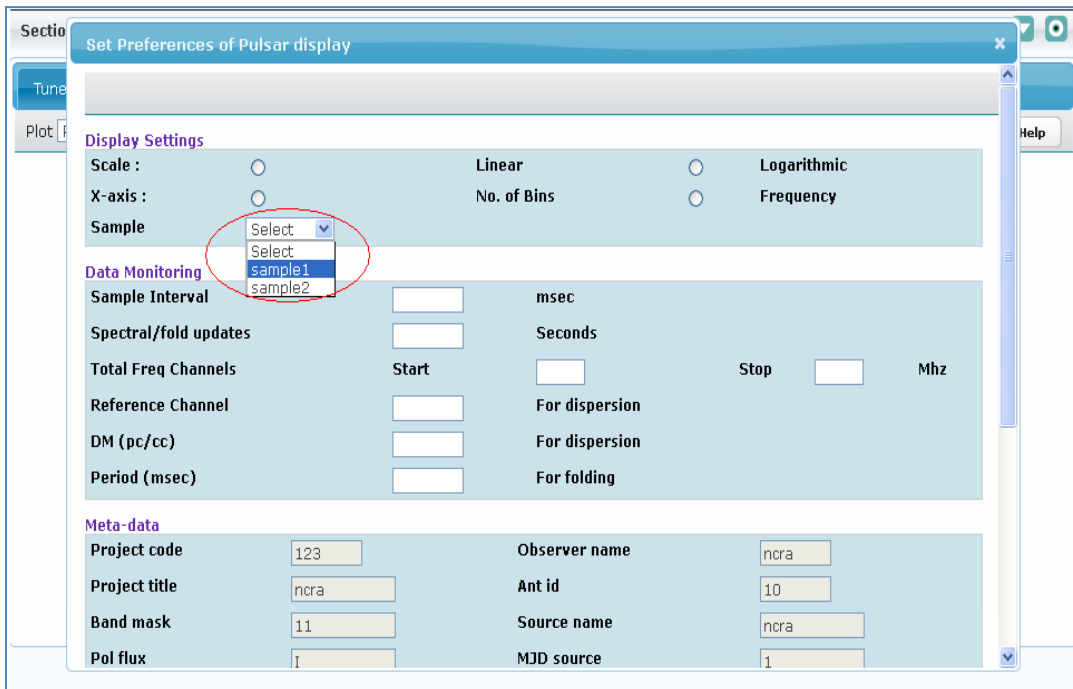
```

<command>
  <name>sendpulsarplottingdata</name>
  <id>107</id>
  <syntax>sample,scale,xaxis,interval,updates,start_chnl
  <sample>sample1,linear,noofbins,10,1,10,20,1,19.56,530
  <timeout>120000</timeout>
  <params>
    <param required="true">
      <paramname>sample</paramname>
      <type>string</type>
      <validation>
        <values>
          <value>sample1</value>
          <value>sample2</value>
        </values>
      </validation>
    </param>
  </params>
</command>

```

Figure 23

User will be able to see the new dropdown control added on the preferences dialog of Pulsar display.



The screenshot shows a 'Set Preferences of Pulsar display' dialog box. It has a sidebar with 'Tune' and 'Plot' buttons. The main area is divided into three sections:

- Display Settings:** Includes radio buttons for 'Scale' (Linear, Logarithmic), 'X-axis' (No. of Bins, Frequency), and a 'Sample' dropdown menu. The 'Sample' menu is open, showing 'Select', 'sample1', and 'sample2'.
- Data Monitoring:** Includes input fields for 'Sample Interval' (msec), 'Spectral/fold updates' (Seconds), 'Total Freq Channels' (Start, Stop, Mhz), 'Reference Channel' (For dispersion), 'DM (pc/cc)' (For dispersion), and 'Period (msec)' (For folding).
- Meta-data:** Includes input fields for 'Project code' (123), 'Project title' (ncra), 'Band mask' (11), 'Pol flux' (1), 'Observer name' (ncra), 'Ant id' (10), 'Source name' (ncra), and 'MJD source' (1).

Figure 24

**Note & Assumptions:**

- Existing Pulsar.xml contains 3 sections Display Settings, Data Monitoring and Meta data section.
- Names of all the input fields must start with “**sendpulsarplottingdata \_\_**”.
- <name> and <cmdid> nodes values are used in java script and other java implementation so don't change these value .If user is changing these node values than need to modify the code according.
- Meta-data section fields are all read-only and they correspond to Global parameter” bean data. User can change this data through “Expert” or “Tune Receiver”