Parameter Panel Level

This is the second level of the UI and is described by the abstract class **ParameterPanel**. A panel is usually implemented as a 'tab'. Implementing classes must override a few methods and are described in the javadoc. The important members of the Panel API are listed below:

Parameter Panel members	Description
public String getPanelKey()	This method returns a unique identifier for the Panel within a Panel Collection.
public String getName()	This is used to provide a name to the Panel or tab.
public String getParameterPanelClassName()	This returns a CSS class to be applied to the Panel. This should be overridden as the default implementation returns null.
public List<ParameterSection> getSections()	Implementations should return a list of ParameterSection objects within this Panel.
public String getDynamicKey() public void setDynamicKey(String dynamicKey)	These methods are used for working with Dynamic Parameters, which is discussed in the section on Helper Objects.
public GeneralPanelOptions getGeneralOptions()	General Options are discussed in the section on Helper Objects. These options control how the UI is rendered.
public List<ParameterDisplayRule> getDisplayRules()	Display Rules are discussed in the section on Helper Objects. They are used to determine whether a panel should be displayed or not based on user input.
public Map<String, ?> getData()	As with ParameterPanelCollection, the default implementation returns the member variable data. Subclasses may override to modify it. This is used by the ParameterPanelCollection's toJSON() method to construct its data object.
public JSONObject toJSON()	This method converts everything in the Panel Collection to an **org.json.JSONObject** (Jackson) object. The main attributes of this object are name, description, panelKey, parameterPanelClassName, sections, displayRules, dynamicKey and generalOptions. ○ **sections** holds a JSONArray of JSONObjects returned by each ParameterSection. ○ **displayRules** holds a list of ParameterDisplayRule, the details of which are discussed in the section on Helper Objects. ○ **generalOptions** holds an instance of GeneralPanelOptions, discussed in the section on Helper Objects.

Implementation

Yellowfin ships with an implementation of ParameterPanel for use with Data Transformation steps. **ETLStepConfigPanel** is easy to use, and contains a method called **addSection()** for adding **ParameterSection** objects. Its implementation of **getGeneralOptions()** includes an "Apply" button to the bottom of the panel, using standard styling from Data Transformations.

```
String stepName = getETLStepBean().getStepName();
ETLStepConfigPanel panel = new ETLStepConfigPanel(PANEL_KEY, stepName);
panel.addSection(section);
```