# Complex UI Set Up

If you require your UI to be dynamic, UIP includes the following features for more complex UI interactions such as dependent parameters, optional parameters, and button callbacks.



## public String buttonPressed(String key)

Button press callback function. This function will be called when the button parameter specified by ‘key’ is pressed. Return a message to show the user in a modal popup.

**Example:**
This example will make a message reading "You pressed the button" pop up when a button is pressed.

```
@Override
protected void setupParameters() {
   Parameter p = new Parameter();
   p1.setUniqueKey("MESSAGE");
   p1.setDisplayType(DISPLAY_BUTTON);
   ...
   addParameter(p);
}

@Override
public String buttonPressed(String key) {
   if ("MESSAGE".equals(key)) {
      return "You pressed the button!";
   }
}
```

top

---

## public boolean isParameterRequired(String key)

Returns true if the specified parameter is required to be displayed. Subclasses can override this method to disable parameters that are not required based on the values of other parameters. Default implementation returns true.

Example:
In the following example, the parameter will only be shown when another parameter has some value which requires extra configuration. For example, you might only want to show some sub-configuration when it is required which helps de-clutter the UI for this plugin.

```
@Override
public boolean isParameterRequired(String key) {
   Object otherValue = getParameterValue("OTHER_KEY");
   if (parameterIsRequiredBasedOnOtherValue(key, otherValue)) {
      return true;
   }
   return false;
}
```

top

---

## public boolean hasDependentParameters(String key)

Returns true if there are other parameters that are dependent on the specified parameter. Subclasses can override this method to set up dependencies between parameters. Default implementation returns false.

By setting up a plug-in like this, whenever the “CHOOSEDOG” parameter is changed (checked or unchecked), then the UI will be reloaded and the sub-configuration of the “DOGCHOICE” parameter will be shown or not shown.

Example of this method, along with isParameterRequired:

```
@Override
protected void setupParameters() {
   // p1 is a toggle which controls whether or not we are choosing a dog
   Parameter p1 = new Parameter();
   p1.setUniqueKey("CHOOSEDOG");
   p1.setDisplayType(DISPLAY_CHECKBOX);
   ...

   // p2 is a dropdown for which dog we will choose, only to be displayed when p1 is checked
   Parameter p2 = new Parameter();
   p1.setUniqueKey("DOGCHOICE");
   p1.setDisplayType(DISPLAY_SELECT);
   p1.addOption("LAB", "Labrador");
   p1.addOption("HUSKY", "Husky");
   ...

    addParameter(p);
}

@Override
public boolean isParameterRequired(String key) {
   if ("DOGCHOICE".equals(key)) {
      Boolean choiceVal = (Boolean)getParameterValue("CHOOSEDOG");
        if (choiceVal == null || choiceVal != true) {
         return false; // don't show this parameter unless the other parameter is true
      }
   }
   return true;
}

@Override
public boolean hasDependentParameters(String key) {
   if ("CHOOSEDOG".equals(key)) {
      return true;
   }
   return false;
}
```

top

**Previous topic:** Basic UI set up
**Next topic:** Appendix