IRIS Explorer User's Guide
Release 3

Document Number NP2872
ISBN 1−85206−110−3

Chapter 4

Editing Control Panels and Functions

This chapter describes how to use two of the IRIS Explorer editors. They are:

- the Control Panel Editor, which you can use to create and modify module control panels.
- the Parameter Function Editor, which you can use to create relationships between parameters in linked modules.

These two editors allow users to modify modules and increase their power without using the Module Builder.

4.1. Overview

IRIS Explorer provides the user with the capacity to edit work in progress at many different stages in the creative process. In the Map Editor, the user edits the sequence of modules in a map. In the Module Builder, the user edits the module resources. IRIS Explorer also provides editors for modifying aspects of the modules themselves.

The Control Panel Editor enables you to create new control panels for modules, groups, and applications, and modify the appearance of existing control panels. You can define new menus on the menu bar and add or rearrange widgets on the control panel.

The Parameter Function Editor lets you establish relationships between parameters in adjacent modules in a map. The parameter value in the downstream module is then expressed as a function of the upstream parameter values.

This chapter describes how to use both these editors.

4.2. Using the Control Panel Editor

The Control Panel Editor (see **Figure 4−1**) lets you modify a module control panel or create a new one by editing the parameter widgets and menu items. You may change a widget's type, position and resize them, and change their limits and default values.

There are three places the Control Panel Editor can be used from:

- To invoke the Control Panel Editor in the Map Editor, you must be creating a group and the Group Editor must be open. Select "Control Panel" from the Edit menu in the Group Editor.
- To create a module control panel in the Module Builder, click on the "Control Panels" push−button in the Module Builder main window.
- In the DataScribe, select "Control Panel" from the View menu when you are creating a DataScribe module.

The Control Panel Editor consists of an Editor window and a Preview control panel window. You stipulate the attributes of the panel in the Editor window, and the result is displayed in the Preview window.

Figure 4−1 The Control Panel Editor

When you create a module, using *mbuilder*, or a group via the Map Editor, you can use the Control Panel Editor to change or create a new widget for each of the input parameters. If you are editing a group, you can only change previously defined widgets, i.e. you can not add new widgets to input parameters which do not already have a controlling widget.

When you edit a control panel, you can:

- Select a widget for each parameter.
- Arrange the widgets as you wish in the Preview window.
- Set the initial value and the range of each widget.
- Use the Menu Bar Editor to design the menus.
- Set a widget attribute for each menu bar item.

To save the menu and control panel configurations, click the *OK* or *Apply* buttons at the bottom of each window.

4.2.1. Choosing Parameter Widgets

The widgets on the control panel allow you to control the value of the module parameters directly.

Parameter Types

An IRIS Explorer parameter may be one of three types of scalar quantity, each of which can be represented as a widget or a menu item on the module, group, or application control panel. A parameter can be:

- a long integer (32−bit)
- a double precision floating point (64−bit)
- a character string

For more information on parameters, see **"Understanding IRIS Explorer Data Types"** in Chapter 2.

Widget Types

The widget types are listed on the Type option menu (see **Figure 4−2**) and described in detail in **"Setting Module Parameters"** in Chapter 2. All widgets must have a valid type selected, including those you place on a menu. The widget attributes are listed in the rest of the Widget Attributes pane.

The properties of the selected widget are shown in the widget properties pane on the right. To display the widget properties, double−click on the widget in the preview control panel.

To select a widget from the Control Panel Editor:

1. Click on the name of the parameter in the "Parameters" pane.
2. Select a widget from the Widget Type option menu in the Widget Attributes pane (see **Figure 4−2**). The widget appears in the Preview window.

Figure 4−2 Selecting a Widget Type

In **Figure 4−2**, for example, the parameter *scale* is highlighted in the Parameters pane. The widget type selected for it on the Type menu is a "Horz Slider" (horizontal slider). The widget is highlighted in the preview window. Its properties are listed to the right under "Slider Properties".

Figure 4−3 shows a selection of widgets, as well as the control panel "decorations." Decorations are vertical and horizontal lines, frames and labels, available from the Decorations menu (see **Figure 4−4**). You can size them and move them around in the same way as you manipulate widgets.

Figure 4−3 A Selection of Widgets

In general:

- dials are used for double precision parameters and sliders are used for long integer parameters. Text type−ins can be substituted for dials and sliders, with less functionality.
- Radio buttons and option menus can be interchanged.
- You can use text type−in slots where file browsers are used, although with more limited scope. File browsers are specifically for locating files, whereas you can use text type−in widgets for any parameter type.
- The scrolled list lets you make a selection from a list of items. Depending on how the list is defined, the user may be able to select only one, or several objects at a time.

Widget similarities are shown in **Figure 4−1**.

Widget	Similar to
Dials	Sliders, Text type−in slots
Radio buttons	Option menus

Text type–in slots	File browsers

Table 4–1 Similar Widgets

You can change the type of widget associated with any parameter by clicking on the parameter name and selecting a new widget type.

Hiding Widgets

The *Initially Hidden* button (see **Figure 4–2**) lets you hide widgets under certain circumstances. These are:

- When you have allocated a parameter associated with a file browser or text type–in slot to a menu in an application.

  This process is explained in detail in **"Creating Menu Items."**
- When you are building a module in the Module Builder and you want a widget to be visible or hidden according to the context the module is in.

  For example, you may have placed a channel selector on the module control panel. If the module is receiving data on one channel only, you may wish to hide the channel selector. If there is more data, you may want the channel selector displayed.

  For more information, see *cxInWdgtHide* in the *IRIS Explorer Module Reference Pages* or the on–line *man* pages.

Widget Decorations

Use the *Decorations* menu to select frames and dividers for your control panel. Once you have selected a decoration, it is listed in the Parameters pane (see **Figure 4–4**).

Figure 4–4 The Decorations Menu

Editing a Scrolled List

Four selection modes are available for the scrolled list widget (see **Figure 4–5**). You can choose which mode suits the kind of parameter you have.

The options are:

Single Select	Only one item at a time can be selected from the list, but the module user need not select any.
Browse Select	The user must select one item from the list.
Multiple Select	Several adjacent items can be selected from the list at one time.
Extended Select	Several items, not necessarily adjacent, can be selected from the list at one time.

Figure 4–5 Scrolled List Properties

4.2.2. Moving and Resizing Widgets

Click on the widget to activate it in both the Preview window and the Editor window. An active widget is enclosed by a "handlebox" of eight black squares (see **Figure 4–6**).

Figure 4–6 Widget Handlebox

- To change the location of the widget in the Preview window, click on the widget and drag it to the desired location. A solid box appears around the widget as you move it.

The Widget Attributes pane in the Editor contains the absolute location of the active widget. The origin of the control panel is in the upper left corner. As you move the widget, the X and Y starting locations change. You can type values directly into these text slots if you wish.

To resize a widget, you can:

- Type values into the width and height slots in the Editor window. This is useful if you have an exact dimension or location for the widget in mind.

A typed–in entry takes effect in the control panel when you press **<Enter>**.

- To change the size of the widget interactively, drag on any of the "handlebox" squares.

For really precise adjustments, use the Grid menu to activate the grid. You can size and position widgets exactly by snapping them to a grid.

The Grid menu contains a list of pixel resolutions that govern the location and size of widgets. You can select any one of these options. The default snaps to a grid size of 5 pixels.

When the "snap to grid" option is in effect, all subsequent widget moves and resizes occur only in multiples of the grid size. If you resize a widget in this mode, it becomes a factor of the grid's width or height, depending on the direction in which you resize it.

4.2.3. Setting Widget Properties

The Properties pane contains information specific to the widget, including the scalar type for dials and sliders and label information for option menus and radio buttons. A plain button has the port name as its label.

The primitive type is important for the proper operation of the module and should not be changed from its default value, which is what the underlying module is expecting. However, you can change it under special circumstances.

To change the widget primitive type, select an alternative from the Properties pane of the Control Panel Editor.

4.2.4. Setting Widget Values and Limits

To set the value of a widget, you must first select it. To select a widget for editing, double–click on it in the Preview window with the left mouse or click on it in the Parameters pane.

To select all the widgets in the control panel at once, use "Select All" on the Edit menu of the Control Panel Editor. To de–select them, click on the background of the control panel.

You can change the value of dial and slider widgets by typing a new value into the upper text slot or moving the widget pointer. You can also set minimum and maximum values by typing them into the range text slots. You must press **<Enter>** for settings to take effect.

You can also change the number of options in the Properties pane for radio buttons and option menus. This might not have the desired effect, however, if you add labels beyond those expected in the module itself.

A good example of this is the *Contour* module. It has three sets of radio buttons that control contouring in the I, J, or K directions. Each radio button has two options: On or Off. If you add a third option, such as Half On, to one of these radio buttons, the module will not process it.

Caution: You can seriously confuse the Map Editor if you switch a module widget type to one that is incompatible with what the module expects.

To dismiss the Control Panel Editor click the *OK* button.

The *Apply* button saves the layout information without closing the Editor window.

The *Cancel* button dismisses the Editor without implementing your changes.

4.3. Creating Menu Items

You can link a parameter with a menu item on the module menu bar instead of giving it a widget on the control panel. For example, instead of having two file browser widgets on the control panel for opening and saving files, you can set up a File menu with the menu items "Open" and "Close," each linked to a file selector.

The Preview control panel has a default menu bar with an empty "Help" menu already in place (see **Figure 4–3**).

Note: For each parameter linked to the menu bar, you need to specify a widget attribute in the Control Panel Editor as well. Use the "Initially Hidden" button to keep them from appearing on the module control panel.

4.3.1. Setting up the Menu Bar

To display the Menu Bar Editor, select "Menu Bar" from the Edit menu on the Control Panel Editor.

The Menu Bar Editor appears (see **Figure 4–7**). You can create up to eight different menus, with up to sixteen items on each menu, depending on the parameters you want to associate with a menu item.

Note: It will help the users of your modules if you follow the conventions for menu placement; that is, File menu on the far left, then Edit menu, then your other menus.

Figure 4–7 The Menu Bar Editor

Set up your menus by typing the name of each menu in the text slots in the Menu Bar Editor, for example, "File." To save them, click on *Apply*.

You can set up menu "stubs" for parameters that currently do not exist and add the items to the menu at a later date if you wish.

4.3.2. Setting up Menu Items

To create the menu options for each item on the menu bar, click on the "Menu" push–button under the menu name. The Menu Editor is displayed (see **Figure 4–8**).

You can create entries for the current menu. For example, reader modules or groups that have reader functions will usually have "New" and "Open" entries in their File menus.

Figure 4–8 The Menu Editor

From the option menu next to the item name, you can set the action to be taken when the user selects this item from the menu bar of the completed module control panel.

The action items are:

None	No action to be taken (useful as a placeholder).
Set param	Sets the value of a widget by passing a specific value to a parameter in the module or group.
File selector	Brings up a Motif file selector. It comes up in the current directory by default.
Script command	Carries out a Skm scripting command.
System command	Carries out a UNIX command and displays the result (if any) in the shell window where IRIS Explorer is running.
Quit	Destroys the module or quits IRIS Explorer when IRIS Explorer is running in application mode.

If you choose *File selector*, a button appears in the next pane, allowing you to define the file selector parameter.

If you choose *Set param*, the button appears again, with a text slot for entering the name of the parameter.

If you choose *Script command*, or *System command*, a text slot appears for entering the name of the command.

4.4. Menus in Group Control Panels

When you create a group or application control panel, you may wish to promote a menu item from an individual module to the group control panel. Unlike other widgets, which are promoted to the group panel along with their associated parameters, the actual menu entry is not promoted. Only the parameter passes to the group level. You must, therefore, create a new menu for the group control panel and redefine the parameter as a menu item at the group level.

4.5. Using the Parameter Function Editor

When a number of connections fan into the same parameter input port, you may wish to express the value on that port, as a function of some or all of the fan in connections. For example you may wish to take the average value of all the connections into the port.

The Parameter Function(P−Func) Editor (see **Figure 4−9**) lets you set up relationships between the fan in values on an input parameter port, so that the value of the parameter is expressed as a function of one or more parameter values from the upstream modules.

To do this, you define each parameter in the P−Func Editor and then create a mathematical expression that defines their relationship. Any change in the value of the upstream parameter (or parameters) is then reflected in the value of the input parameter.

Figure 4−9 The Parameter Function Editor

4.5.1. Running the Parameter Function Editor

You run the P−Func Editor from the module containing the parameter you want to control. Before you can write a parameter function for an input parameter, you must have one or more connections, from modules upstream, wired to its port.

Figure 4−10 Module Pop−up Menu

To display the P−Func Editor window:

1. Select a module that has at least one input parameter connected to an upstream module.
2. Click on the title bar of the module control panel and select "P−Func Editor" from the pop−up menu (see**Figure 4−10**).

The title bar of the P−Func Editor (see **Figure 4−9**) displays the name of the module for which you are creating the parameter function. The window also has these components:

- The Input Parameter option menu displays all the input parameters to the module. If the module has more than one parameter, you can select a different one to work with from the option menu.
- The Connections box lists all the connections to this input parameter from other modules in the network.
- The Variables box shows a short, unique alias for each incoming connection. The alias simplifies the task of entering the parameter function into the P−Func Editor.
- The Parameter Function box shows the expression defining the relationship between the upstream parameters and the selected parameter.
- The Message box displays a message after you click on *Check* or *Apply*. It will tell you whether you have entered a parameter function with incorrect syntax.

4.5.2. Defining a Parameter Expression

The P−Func Editor provides a simple set of expressions for creating compound relationships among two or more input parameters in a module. You can use all, or any subset of, the variables listed in the Variables box to construct your function. For example, if you have two connections, with variables A and B, you can type `A + 2*B`, or `A/B`, or `(A + B)/2` (the average of two inputs).

Table 4−2 lists the expressions accepted by IRIS Explorer.

Type	Expressions
Arithmetic	*, /, % (mod), +, −

Functions	min(A, B), max(A, B), exp(A), log(A), log10(A), pow(A, B), abs(A), ceil(A), floor(A), rint(A), cos(A), sin(A), tan(A), acos(A), asin(A), atan(A), atan2(A, B), cosh(A), sinh(A), tanh(A), length (returns the string length in characters)
Typecast	(long), (double), (string)
Relational	==, !=, >, >=, <, <=
Boolean	&& or AND, || or OR, ! or NOT
Ternary	A?B:C (`if A then B else C`) Multiple *if then* clauses followed by a terminal *else*

Table 4–2 Parameter Expressions

The expressions are identical to the C language functions. For more information, refer to the C language manual pages for each function.

Typecasting Parameters

IRIS Explorer accepts incoming parameters and turns them into floats by default. If you want the parameter to have a different type, you can use one of the typecast expressions (see **Table 4–2**).

For example, if you have parameters A and B coming in as doubles and you require them in integer form, you can cast them to type (long) as follows:

```
(long)A
(long)B
```

Using Strings

You can feed string parameter values to the P–Func Editor. Strings are groups of alphanumeric characters The only operator you can use on strings is +. The expression

```
(string)A + (string)B
```

indicates concatenation of the two terms. You can also create an expression of the form:

```
(string)(long)(3.3 + 8.9)
```

in which case, IRIS Explorer returns the a string of the value of "12".

You can add like terms only. For example, the expression:

```
A + (string)B
```

will produce an error.

If a string is not a number, it goes to zero. If it is a number, it returns a value. For example,

```
(double) (7+3)
```

returns a double.

Combining Functions

You can create more complex expressions using several functions. Here are two examples:

```
min(sin(A * 3.14159/180.0)
cos(B*3.14159/180.0))
```

and

```
If sin(A) < 0 then this
If sin(A) > 0 then that
else the other
```

Linking Independent Parameters

The upstream connections, supplying the values for the P–Func variables, need not be related, i.e. they can come from unrelated modules or the same module, it does not matter. If they come form different modules then there is no relationship between `A` and `B`; the parameter function relates `A` and `B` to the input parameter independently of each other.

4.5.3. Creating a Parameter Function

In this example (see below), a relationship is created between parameters from two different modules. The upstream module is *IsosurfaceLat*. It has a "Threshold" parameter (shown as a dial), which will be wired to the minimum and maximum level inputs of *Contour*, the downstream module (see **Figure 4–11**).

The parameter function is applied only when a upstream parameter (one corresponding to a variable in the P–Func expression) is changed, "Threshold" in this example. It is not applied if you change the subject parameter, Contour's "Min Level" or "Max Level" parameters, in this example. The change must be in the upstream parameter.

Example 4–1 Creating Simple Functions

This example shows how to link parameters from different modules, and then define the relationships between them. Launch the map named *simple*, consisting of *ReadLat* connected to *IsosurfaceLat* connected to *Render*.

To make a parameter connection (see **Figure 4–11**):

1. Launch a *Contour* module. Connect up *Contour*'s "Input – – Lattice" input port to *ReadLat*'s "Output – –Lattice" port.
2. Connect up *Contour*'s "Contours – –Geometry" port to *Render*'s "Input – –Geometry (Opt)" port.
3. Create a parameter connection by wiring *IsosurfaceLat*'s "Threshold" output port to *Contour*'s "Min Level" input port.
4. Now turn the dial on the *IsosurfaceLat* module. You'll see both *IsosurfaceLat* and *Contour* fire.

Figure 4–11 Connections for a Parameter Function

To create a parameter function:

1. Connect *IsosurfaceLat*'s "Threshold" output to *Contour*'s "Max Level" input port.
2. Bring up the P–Func Editor for *Contour* (see **Figure 4–12**). Select the "Min Level" parameter from the Input Parameter list.
3. A connection for *IsosurfaceLat* Threshold appears in the Connections box. This connection is defined as `A` in the Variables box.
4. Select the Parameter Function box by clicking in it, then type:

   ```
   A * 1.3
   ```

   Click *Apply* at the bottom of the Editor window to implement your modification.
5. Similarly, select the "Max Level" parameter from the option menu and type:

   ```
   A * 1.8
   ```

   Click *Apply* again.

Figure 4–12 Parameter Functions for a Module

Close the P–Func Editor and save the modified map under a new name.

Now when you move the dial on *IsosurfaceLat*, *Contour* fires as well. Set the number of levels in *Contour* to 2 or 3 and see what happens. The image should resemble that in **Figure 4−13**.

The values of the contour lines, therefore, are always 30% to 80% higher than the isosurface threshold.

Figure 4−13 Simple Example with a Parameter Function

If you click on the *Check* button, the P−func Editor tests the function you have entered to make sure that it is valid and prints a message in the Message pane. The Editor window stays open. For example, the second expression in the Parameter Function pane in **Figure 4−14** has been entered incorrectly.

Figure 4−14 Messages in the P−Func Editor

4.5.4. Summarizing the Process

This is a summary of the steps you take to create a parameter function in the P−Func Editor:

1. Select the input parameter for which you want to establish a parameter function from the Input Parameter option menu. It must have at least one connection to an upstream module, as shown in the Connections and Variables boxes.
2. If you wish to change the default variable names (`A, B, C, ...`), click on the variable name you wish to change, in the Variables box, see **Figure 4−12**.
3. Type a new name for the variable and press **`<Enter>`** to have the new name accepted. The default names are `A, B, C,` and so on, but you can type in any name you like to replace them.

   Note: The P−Func Editor is case−insensitive.

4. In the Parameter Function box, enter an expression that defines the relationship you want between the variables in the Variable box, if any.

Saving the Parameter Functions

Use the four buttons at the bottom of the Editor to save or reject modifications to the input parameter. The buttons function as follows:

OK	Accepts the changes you have made, implements them, and closes the Editor window.
Apply	Accepts the changes you have made, implements them, and keeps the Editor window open.
Check	Checks the validity of the function and sends a message to the Message pane.
Cancel	Closes the Editor window without accepting or implementing any changes on the screen.

Example 4−2 Creating a Complex Function

This example illustrates how to create a more complex parameter function by establishing a relationship between three parameters. The parameters are *Contour*'s Min Level and Max Level and *IsosurfaceLat*'s Threshold.

First, create a map by launching a *GenLat* module, a *Contour* module, an *IsosurfaceLat* module, and a *Render* module and wiring them together as shown in **Figure 4−15**.

To create an object in *Render* like the one in **Figure 4−15**, open *GenLat*'s full−scale control panel and set these widgets as follows:

- *Coord Type*: curv torus
- *Coord Representation*: curvilinear
- *Function*: sines
- *Data Type*: float

You want the threshold value of the isosurface to lie directly in the midrange of the contour levels generated by the *Contour* module. To do this, you connect two output ports from *Contour* to an input port on *IsosurfaceLat*, and set up a relationship between the two output parameters and the input parameter.

Figure 4–15 A Parameter Function Using Two Variables

To make these connections:

1. Connect *Contour*'s "Min Level" output parameter into *IsosurfaceLat*'s "Threshold" input parameter.
2. Connect *Contour*'s "Max Level" output parameter into *IsosurfaceLat*'s "Threshold" input parameter.
3. Bring up the P–Func Editor from *IsosurfaceLat*. The parameter of interest on *IsosurfaceLat*, "Threshold," is highlighted in the Input Parameters pane. The Connections area has two listings, *Contour.Min Level* and *Contour.Max Level* aliased as `A` and `B`.
4. In the Parameter Function box, type in the expression: **`(A + B) / 2.0`** This averages the two inputs that are fanned into "Threshold".
5. Click on the *Apply* button to establish the relationship.
6. Set *Contour*'s Min Level dial to 0.0 and the Max Level dial to 1.0. Set the number of contour levels to 4 or 5.

From now on, when you turn *Contour*'s Min and Max Level dials, the Map Editor sends these values to IRIS Explorer's parameter interpreter. It evaluates the expression and feeds the result to *IsosurfaceLat*'s "Threshold" parameter.

You can see the calculated value reflected in the position of *IsosurfaceLat*'s "Threshold" slider. There are now contour levels outside the isosurface.

You can change the transparency of the isosurface in *Render* so that you can see the contour values inside the surface as well. For more information, see **"Visualizing Data"** in Chapter 3, "Using Modules."