NAME

editres – a dynamic resource editor for X Toolkit applications

SYNTAX

editres [*-toolkitoptions...*]

OPTIONS

Editres accepts all of the standard X Toolkit command line options (see **X11**(7)). The order of the command line options is not important.

DESCRIPTION

Editres is a tool that allows users and application developers to view the full widget hierarchy of any X Toolkit client that speaks the **editres** protocol. In addition **editres** will help the user construct resource specifications, allow the user to apply the resource to the application and view the results dynamically. Once the user is happy with a resource specification editres will append the resource string to the user's X Resources file.

USING EDITRES

Editres provides a window consisting of the following four areas:

Menu Bar	A set of popup menus that allow you full access to editres's features.
Panner	The panner allows a more intuitive way to scroll the application tree display.
Message Area	Displays information to the user about the action that editres expects of the her.
Application Widget Tree	This area will be used to display the selected client's widget tree.

To begin an editres session select the **Get Widget Tree** menu item from the command menu. This will change the pointer cursor to cross hair. You should now select the application you wish look at by clicking on any of its windows. If this application understands the editres protocol then editres will display the client's widget tree in its tree window. If the application does not understand the editres protocol editres will inform you of this fact in the message area after a few seconds delay.

Once you have a widget tree you may now select any of the other menu options. The effect of each of these is described below.

COMMANDS

Send Widget Tree
: Allows the user to click on any client that speaks the editres protocol and receive its widget tree.

Refresh Widget Tree
: Editres only knows about the widgets that exist at the present time. Many applications create and destroy widgets "on-the-fly". Selecting this menu item will cause editres to ask the application to resend its widget tree, thus updating its information to the new state of the application.

Example
: Xman only creates the widgets for its *topbox* when it starts up. None of the widgets for the manual page window are created until the user actually clicks on the *Manual Page* button. If you retrieved xman's widget tree before the the

manual page is active, you may wish to refresh the widget tree after the manual page has been displayed. This will allow you to also edit the manual page's resources.

Select Widget in Client
: This menu item allows you to select any widget in the application, editres will then highlight the corrosponding element the widget tree display. Once this menu item is selected the pointer cursor will again turn to a crosshair, and you must click any pointer button in the widget you wish to have displayed. Since some widgets are fully obscured by their children, it is not possible to get to every widget this way, but this mechanism does give very useful feedback between the elements in the widget tree and those in the actual client.

Dump Widget Tree to a File
: For documenting applications it is often useful to be able to dump the entire application widget tree to an ascii file. This file can then be included in the manual page. When this menu item is selected a popup dialog is activated. Type the name of the file in this dialog, and either select *okay*, or type a carriage-return. Editres will now dump the widget tree to this file. To cancel the file dialog just select the *cancel* button.

Show Active Widgets
: This command is the inverse of the **Select Widget in Client** command, it will show the user each widget that is currently selected in the widget tree, by flashing the corresponding widget in the application *numFlashes* (three by default) times in the *flashColor*.

Show Resource Box
: This command will popup a resource box for the current client. This resource box (described in detail below) will allow the user to see exactly which resources can be set for the widget that is currently selected in the widget tree display. Only one widget may be currently selected, if greater or fewer are selected editres will refuse to pop up the resource box, and put an error message in the **Message Area**.

Show Author
: Pops up message telling you who wrote this application.

Quit
: Exits editres.

TREE COMMANDS

The **Tree** menu contains several commands that allow operations to be performed on the widget tree.

Select All
Unselect All
Invert All
: These functions allow the user to select, unselect, or invert all widgets in the widget tree.

Select Children
Select Parents

These functions select the immediate parent or children of each of the currently selected widgets.

Select Descendants
Select Ancestors
These functions select all parents or children of each of the currently selected widgets. This is a recursive search.

Show Widget Names
Show Class Names
Show Widget Windows
When the tree widget is initially displayed the labels of each widget in the tree correspond to the widget names. These functions will cause the label of **all** widgets in the tree to be changed to show the class name, IDs, or window associated with each widget in the application. The widget IDs, and windows are shown as hex numbers.

In addition there are keyboard accelerators for each of the Tree operations. If the input focus is over an individual widget in the tree, then that operation will only effect that widget. If the input focus is in the Tree background it will have exactly the same effect as the corresponding menu item.

The translation entries shown may be applied to any widget in the application. If that widget is a child of the Tree widget, then it will only affect that widget, otherwise it will have the same effect as the commands in the tree menu.

Key	**Option**	**Translation Entry**
space	Unselect	Select(nothing)
s	Select	Select(all)
i	Invert	Select(invert)
c	Select Children	Select(children)
d	Select Descendants	Select(descendants)
p	Select Parent	Select(parent)
a	Select Ancestors	Select(ancestors)
N	Show Widget Names	Relabel(name)
C	Show Class Names	Relabel(class)
I	Show Widget IDs	Relabel(id)
W	Show Widget Windows	Relabel(window)

USING THE RESOURCE BOX

The resource box contains five different areas. Each of the areas, as they appear on the screen, from top to bottom will be discussed.

The Resource Line
This area at the top of the resource box shows the current resource entry exactly as it would appear if you were to save it to a file.

The Widget Names and Classes
This area allows you to select exactly which widgets this resource will apply to.

The area contains three lines, the first contains the name of the selected widget and all its ancestors, and the more restrictive dot (**.**) separator. The second line contains less specific the Class names of each widget, and well as the less restrictive star (*) separator. The last line contains a set of special buttons called **Allow Any Widget** which will completely compress out this level of the widget hierarchy, replacing it with just the star separator.

The initial state of this area is the most restrictive, using the resource names and the dot separator. By selecting the other buttons in this are you can ease the restrictions to allow more and more widgets to match the specification. The extreme case is to select all the **Allow Any Widget** buttons, which will match every widget in the application. As you select different buttons the tree display will update to show you exactly which widgets will be effected by the current resource specification.

Normal and Constrain Resources
: The next area allows you to select the name of the normal or constraint resources you wish to set. Some widgets may not have constraint resources, so that area will not appear.

Resource Value
: This next area allows you to enter the resource value. This value should be entered exactly as you would type a line into your resource file. Thus it should contain no unescaped new-lines. There are a few special character sequences for this file:

\n - This will be replaced with a newline.

\### - Where # is any octal digit. This will be replaced with a single byte that contains this sequence interpreted as an octal number. For example, a value containing a NULL byte can be stored by specifying \000.

\<new-line> - This will compress to nothing.

\\ - This will compress to a single backslash.

Command Area
: This area contains several command buttons that I will describe in this section.

Set Save File
: This button allows the user to modify file that the resources will be saved to. This button will bring up a dialog box that will ask you for a filename, once the filename has been entered, either hit carriage-return or click on the *okay* button. To popdown the dialog box without changing the save file, click the *cancel* button.

Save
: This button will append the **resource line** described above to the end of the current save file. If no save file has been set the **Set Save File** dialog box will be popped up to prompt the user for a filename.

Apply
: This button attempts to perform a XtSetValues call on all widgets that match the

resource line described above. The value specified is applied directly to all matching widgets. This behavior is an attempt to give a dynamic feel to the resource editor. Since this feature allows users to put an application in states it may not be willing to handle, a hook has been provided to allow specific clients block these SetValues requests (see **Blocking Editres Requests** below).

Unfortunately due to design constraints imposed on the widgets by the X Toolkit and the Resource Manager, trying to coerce an inherently static system into dynamic behavior can cause strange results. There is no guarantee that the results of an apply will be the same as what will happen when you save the value, and restart the application. This functionality is provided to try to give you a rough feel for what your changes will accomplish, and the results obtained should be considered suspect at best. Having said that, this is one of the neatest features of editres, and I strongly suggest that you play with it, and see what it can do.

Save and Apply
: This button combines the Save and Apply actions described above into one button.

Popdown Resource Box
: This button will remove the resource box from the display.

BLOCKING EDITRES REQUESTS

The editres protocol has been built into the Athena Widget set. This allows all application that are linked against Xaw to be able to speak to the resource editor. While this provides great flexability, and is a useful tool, it can quite easily be abused. It is therefore possible for any Xaw client to specify a value for the **editresBlock** resource described below, to keep editres from divulging information about its internals, or to disable the **SetValues** part of the protocol.

editresBlock (Class **EditresBlock)**
: Specifies which type of blocking this client wishes to impose on the editres protocol.

The accepted values are:

all	Block all requests.
setValues	Block all setvalues request, this is the only editres request that actually modifies the application, this is in effect stating that the applicaion is read-only.
none	Allow all editres requests.

Remember that these resources are set on any Xaw client, **not editres**. They allow individual clients to keep all or some of the requests editres makes from ever succeeding. Of course, editres is also an Xaw client, so it may also be viewed and modified by editres (rather recursive, I know), these commands can be blocked by setting the **editresBlock** resource on editres itself.

RESOURCES

For **editres** the available application resources are:

numFlashes (Class **NumFlashes)**
Specifies the number of times the widgets in the client application will be flashed when the **Show Active Widgets** command in invoked.

flashTime (Class **FlashTime)**
Amount of time between the flashes described above.

flashColor (Class **flashColor)**
Specifies the color used to flash client widgets. A bright color should be used that will immediately draw your attention to the area being flashed, such as red or yellow.

saveResourcesFile (Class **SaveResourcesFile)**
This is the file the resource line will be append to when the **Save** button activated in the resource box.

WIDGETS

In order to specify resources, it is useful to know the hierarchy of the widgets which compose **editres**. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
Editres  editres
        Paned  paned
                Box  box
                        MenuButton  commands
                                SimpleMenu  menu
                                SmeBSB  sendTree
                                SmeBSB  refreshTree
                                SmeBSB  showClientWidget
                                SmeLine  line
                                SmeBSB  dumpTreeToFile
                                SmeBSB  flashActiveWidgets
                                SmeBSB  getResourceList
                                SmeLine  line
                                SmeBSB  quit
                        MenuButton  treeCommands
                                SimpleMenu  menu
                                SmeBSB  selectAll
                                SmeBSB  unselectAll
                                SmeBSB  invertAll
                                SmeLine  line
                                SmeBSB  selectChildren
                                SmeBSB  selectParent
                                SmeBSB  selectDescendants
                                SmeBSB  selectAncestors
                                SmeLine  line
                                SmeBSB  showWidgetNames
```

```
                SmeBSB  showClassNames
                SmeBSB  showWidgetIDs
                SmeBSB  showWidgetWindows
        Paned  hPane
            Panner  panner
            Label  userMessage
            Grip  grip
        Porthole  porthole
            Tree  tree
                Toggle  <name of widget in client>
                .
                .
                .
                TransientShell  resourceBox
                Paned  pane
                Label  resourceLabel
                Form  namesAndClasses
                Toggle  dot
                Toggle  star
                Toggle  any
                Toggle  name
                Toggle  class
                  .
                  .
                  .
                Label  namesLabel
                List  namesList
                Label  constraintLabel
                List  constraintList
                Form  valueForm
                Label  valueLabel
                Text  valueText
                Box  commandBox
                Command  setFile
                Command  save
                Command  apply
                Command  saveAndApply
                Command  cancel
                Grip  grip
        Grip  grip
```

ENVIRONMENT

DISPLAY
to get the default host and display number.

XENVIRONMENT

to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

FILES

/usr/openwin/lib/app-defaults/Editres - specifies required resources

SEE ALSO

X11(7), **xrdb**(1), Athena Widget Set

COPYRIGHT

AUTHOR

Chris D. Peterson, formerly MIT X Consortium