Applescripting the Unscriptable: Using the GUI Scripting Beta

Ted Stevko
Stevko Studios

O'Reilly Mac OS X Convention
October 27-30, 2003

O'REILLY®
Mac OS X
Conference

Notes about this presentation

- This is about half-10.2.8, half-10.3, so forgive me if there's any mistakes or issues; some of this changed between the beta and the release in 10.3.
- I dropped the final Applescript file into SubEthaEdit for perusal.
- Comments about my poor Applescript techniques are welcomed, encouraged, and hoped for; send them to me at my e-mail, ted@stevko.com
- If you've got questions, shout out.
- I'd like to thank Matt Neuburg for allowing me to borrow the name, and Rogue Amoeba for allowing me to use their program as an example.

Why? It’s a long story...

- This all started out because I listen to headphones all day
- Audio streams aren't always convenient or stoppable.
- To fix, I got Audio Hijack Pro... but I can’t run it on my Wintel machine at work.
- Audio Hijack Pro has no Applescript support
- So I had to find another way to do this...

GUI Scripting

- End users benefit from GUI scripting by being able to script those programs which don't have Applescript support
- Programmers who cannot build in Applescript support directly can support GUI scripting, and gain additional benefit from doing so.
- Goal is **not** to discourage building in Applescript support -- lots of discussion on Apple's Applescript user list about this.

Goals of this presentation

- To go through what's been introduced in the GUI Scripting
- Touch on how GUI scripting works on the back end for Cocoa/Carbon programmers
- Show how to build a GUI script for a program that has no built-in Applescript
- Once these goals are met, I'll show the script that let me achieve my goal of being able to ignore the world for days at a time.

What can you do with GUI scripting?

- Can access any standard GUI element's attributes and actions
- This currently includes all elements built into Cocoa & Carbon, as well as element in Java apps that use the Cocoa/Java bridge.
- Claims it works with Java, but I'm not finding that true for Swing apps. Haven't tested AWT.

What can you do with GUI scripting?

- Actions include clicking radio & checkboxes, adding text to text areas, opening and selecting elements from menus, etc.
- Also includes key commands and x/y coordinate clicking (sort of)
- A full listing of all elements is available in the System Events dictionary; we'll cover the major ones.

How's it work?

- Information on GUI Scripting from Apple http://www.apple.com/applescript/GUI/
- Beta for 10.2 Released in December 2002, final 1.0 in Panther
- The beta was a download consists of an update to the System Events program and a UI Element Inspector
- The final is included in Panther directly, with the UI Element Inspector a seperate download.
- Sample scripts are available at the above address for the 1.0 release only now (and there were changes).
- Sample scripts in the scripts menu

But how do I script these elements, dammit?

- OK, now the obligatory “where can I find more” page is past, how do you script these things?
- It’s pretty simple. You tell the System Events application, to tell the application you want to script, to either get info about the GUI elements or tell it’s GUI elements to do an action.
- This leaves us a few problems, though
  - What are the basic GUI elements?
  - How do I tell Applescript which GUI elements I want?
  - What info can I get from the GUI element?
  - What can I do to the GUI element?

Finding The Basic Applescript Classes and Commands

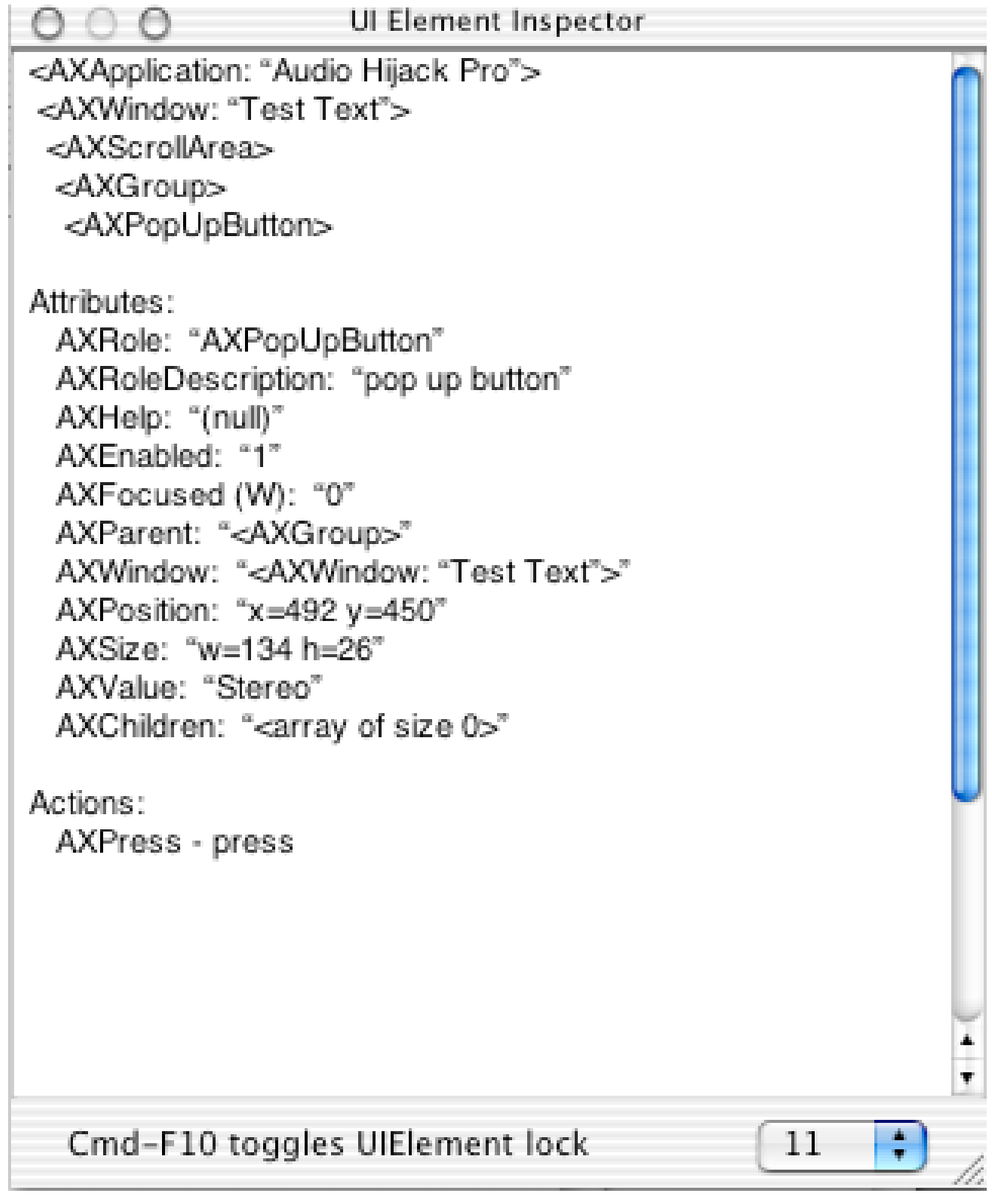
- The classes and commands for these are found in the System Events application
- System Events is a background app that allows you to system level things, like create clicks, attach scripts to folders, etc.
- In the beta the classes were added to "System Events Suite" in the downloaded System Events
- Panther contains these updates with the standard install, this time inside of “Process Suite”
- Other than knowing that this is the application you select to find the GUI scripting dictionary, this can be nicely ignored.

Finding Out Things About An Application You Want To Script

- This is where the UI Element Inspector comes in.
- You use this to find out about not only the application you want to script
- This application goes out and reads the accessability hierarchy -- not the object hierarchy -- and relates information about the element directly under your mouse
- Lessons for programmers: build your GUIs carefully & logically -- now your end users might be looking at these.

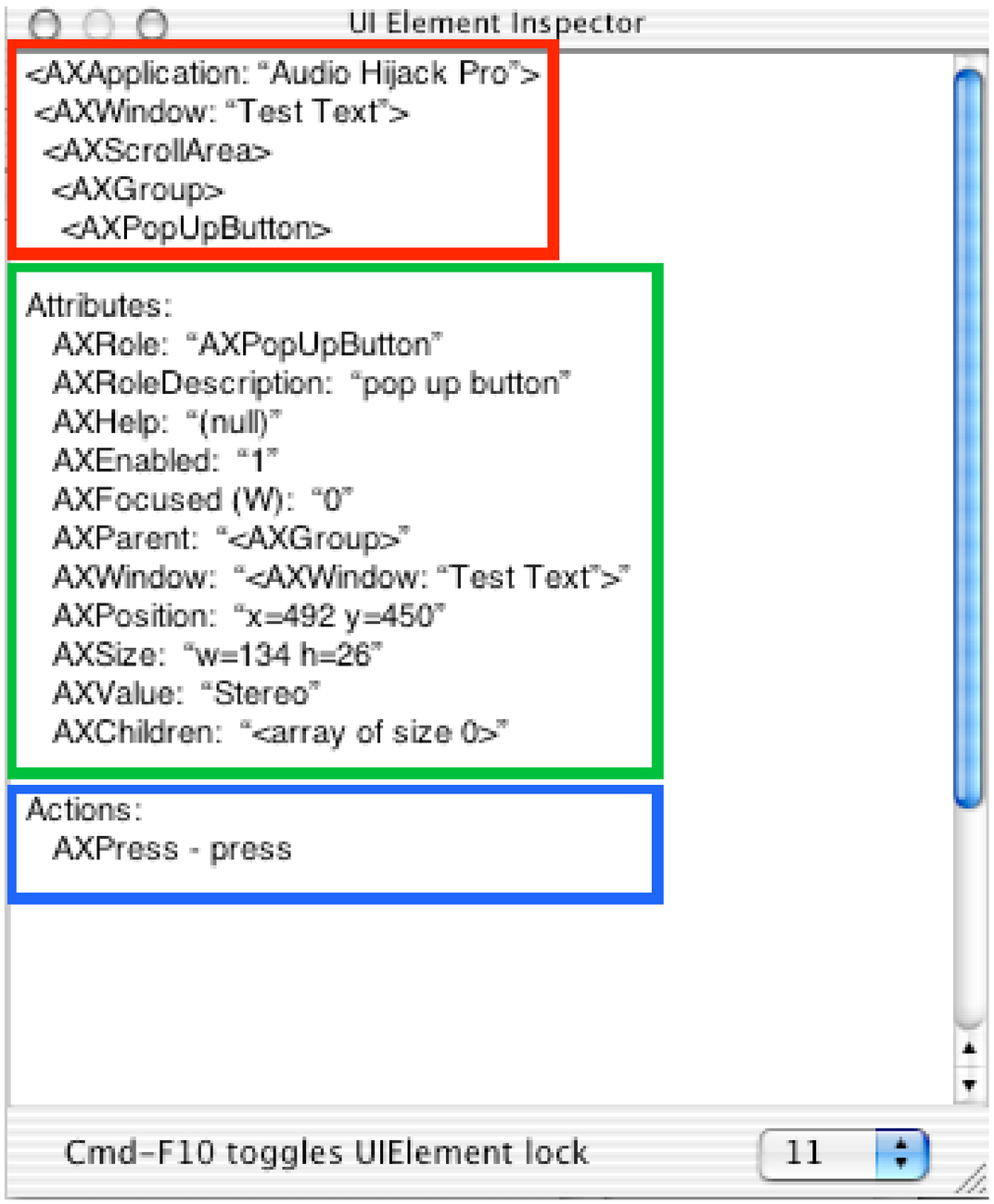
What's the UI Element Inspector?

- Program that allows you to roll over a GUI and "inspect" the elements of any GUI interface.
- Same program bundled with the Accessability API
- It's the most confusing thing about Applescript GUI Scripting.
- So, let's take a look at it!

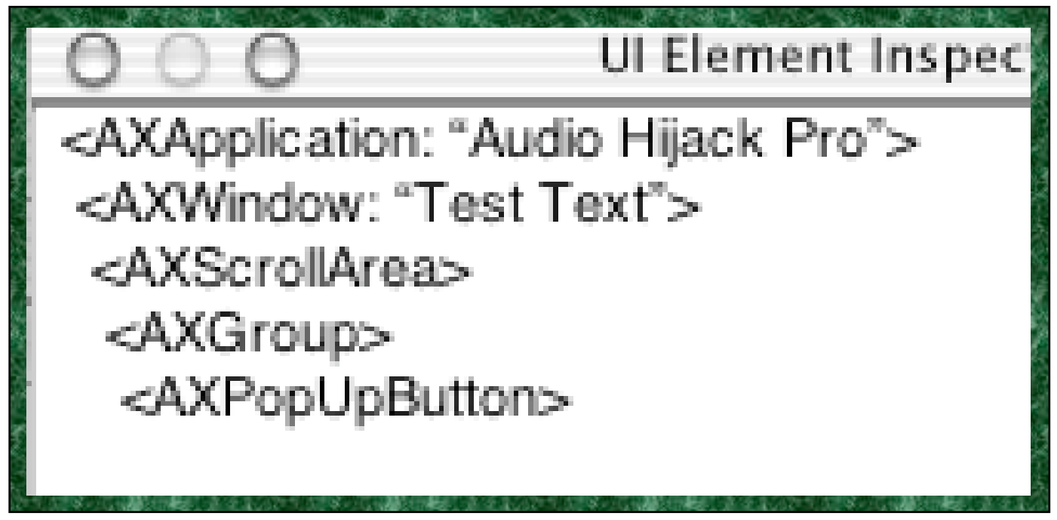


Looking at the Inspector

- If you look at this while you roll over things, you get three distinct areas
- The tree heirarchy of an element
- The attributes of an element
- The actions available to an element



Tree Hierarchy



- This shows the tree you're going to have to walk down to access a GUI element
- So, in this case, you would need to reference the pop up button of the group of the scroll area of the window "Test Text" of application "Audio Hijack Pro"
- References can be made by either name or element number -- although UI Element Inspector doesn't tell you what the number is

What the heck's "AXPopUpButton"?

- This is the proper name of the element as seen by Apple's Accessability API.
- Apple introduced an Accessibility API to allow non-standard input devices to access applications
- Each element of the standard Cocoa and Carbon GUI API implements the accessibility API
- In Applescript, though, you will be calling things like "pop up button", not "AXPopUpButton". This particular tree list is to be used as more of a guide than an absolute.

Attributes and Actions

- Lots of attributes for every element, but not all available for getting
- Some attributes can be "set" -- ones with (W) in front of them
- Some attributes are a pain to access, like children. Will not return a list when called; you have to build your own list.
- Actions show what you can do to that element

```
Attributes:
   AXRole: "AXPopUpButton"
   AXRoleDescription: "pop up button"
   AXHelp: "(null)"
   AXEnabled: "1"
   AXFocused (W): "0"
   AXParent: "<AXGroup>"
   AXWindow: "<AXWindow: "Test Text">"
   AXPosition: "x=492 y=450"
   AXSize: "w=134 h=26"
   AXValue: "Stereo"
   AXChildren: "<array of size 0>"

Actions:
   AXPress - press
```

Why can’t I script some elements?

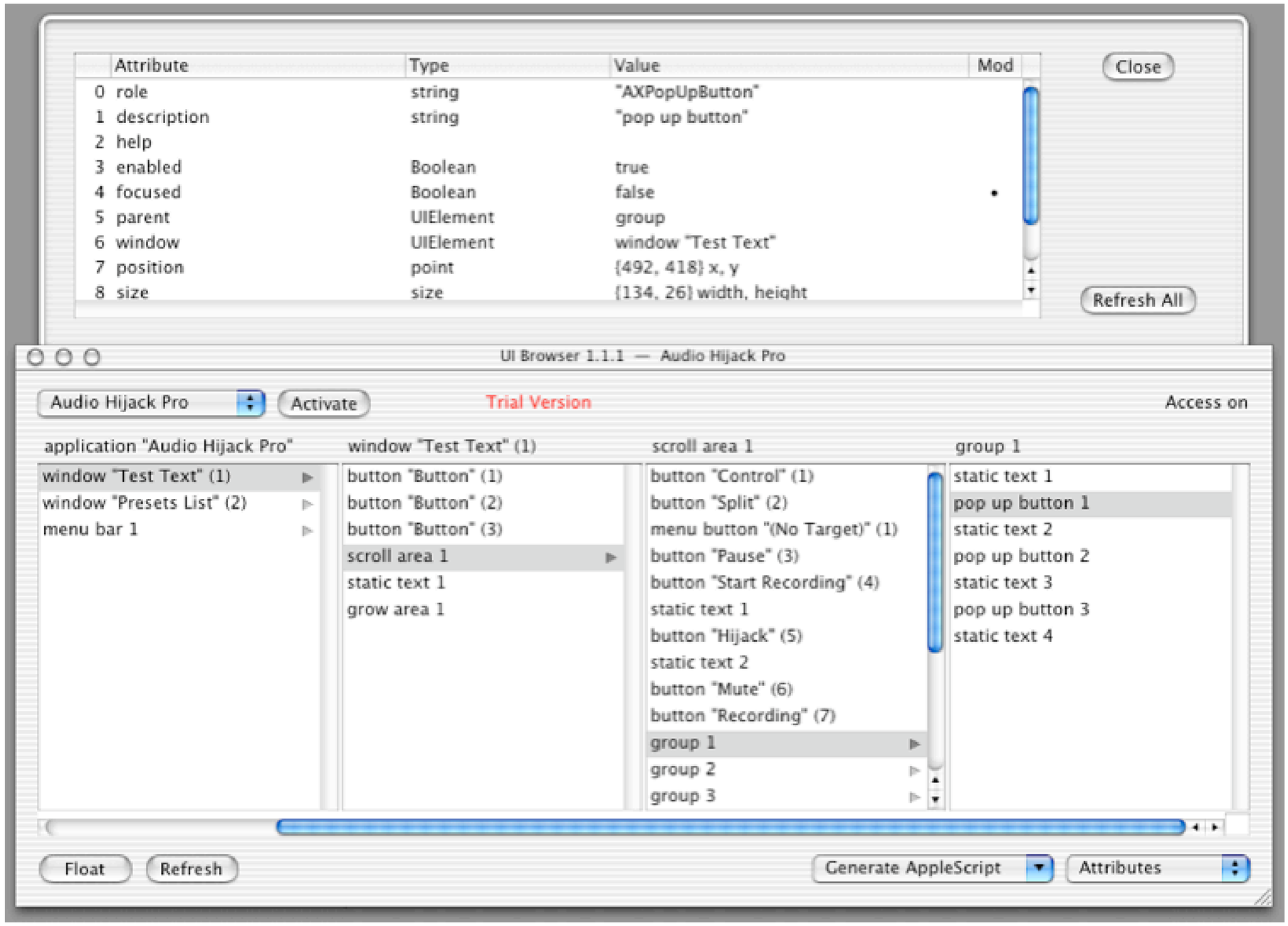
- Accessibility is built into the standard set of UI elements -- but custom UI elements need to implement the Accessibility API to be GUI scriptable.
- For Applescript, you don't need to know this; but if you're building an application, this is why a custom UI element would not have access.
- For Carbon/Cocoa builders, having your custom GUI elements implement the Accessability API’s GUI classes allows them to be accessed both by Applescript and by non-standard input devices
- Java developers seem to need to use the Cocoa/ Java bridge instead of Swing for GUIs.

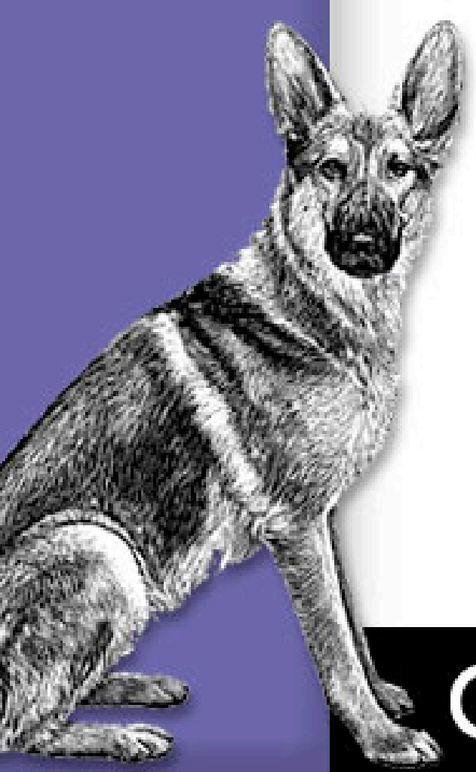
How do I use the UI Element Inspector?

- Use the tree information to find where the UI element is
- Use the attributes to find out information about the UI element
- Use the actions to find out what actions you can perform.
- This gives you the information you need to script any specific UI element.
- But.. the UI Element inspector is a pain -- it was built for working with the Accessability API.

Prefab UI Browser

- Prefab UI Browser: http://www.prefab.com/uibrowser/, $40





Prefab UI Browser

- Easier to use to walk the tree: elements have clear Applescript names, and UI Browser gives number references, where UI Element Inspector does not
- Allows you to test setting attributes, clicking elements, doing keyboard commands, etc. without having to built test Applescripts
- Best of all, UI Browser builds Applescript statements for you, including clicking elements, getting and setting attributes, and building basic try blocks.
- If you're doing GUI scripting, it's near essential. Apple's product isn't built for doing Applescripting.

Building The Script

- Because you're calling the System Events app, you tell the System Events application to tell an application process.
- This gets you access to it’s GUI elements. If you tell the application directly, it would be a reference to that program’s Applescript elements.

```
tell "System Events"
    tell application process "Audio Hijack Pro"
        <!-- insert items here -->
    end tell
end tell
```

Referencing elements

- Since each of the items is in a vast tree, you just have to descend the tree.
- Either tell a series of items (set value of text field 1 of row presetNum of table 1 of scrollArea 1 of "Preset List" to timerName) or..
- Use a series of tell statements --

```
tell "System Events"
    tell application process "Audio Hijack Pro"
        tell window “Preset List”
            tell scroll area 1
                click button 1
            end tell
        end tell
    end tell
end tell
```

Getting and setting attributes

- Getting attributes is as simple as “get <attribute> of <UI element>”; setting is “set, etc.”
- Note: Some items in the beta claimed to be modifiable, and are not: you will need to test. Not sure about Panther yet.

```
tell "System Events"
    tell application process "Audio Hijack Pro"
        tell window “Preset List”
            tell scroll area 1
                set theVar to get value of button 1
                set value of button 1 to theVar
            end tell
        end tell
    end tell
end tell
```

Buttons

- Again, simple as can be: find the element and use "click"
- A button (or any other element) has to be enabled (enabled attribute is true) in order to click it, or perform any other action on it.

```
tell "System Events"
    tell application process "Audio Hijack Pro"
        tell window “Preset List”
            tell scroll area 1
                click button 1
            end tell
        end tell
    end tell
end tell
```

Drop-down Menus

- Drop-down menus include both "pop up buttons" and "menu buttons"
- You must click the button first to be able to reach the menu items.
- Once the button is clicked, the drop down menus then have access to their menu child element, which has an array of child menu items.

```
tell "System Events"
    tell application process "Audio Hijack Pro"
        tell window “Preset List”
            click pop up button 1
            pick menu item "MP3 VBR" of menu "
OtherViews" of pop up button 1
        end tell
    end tell
end tell
```

Menus & Menu Items

- Menus are basically containers for menu items; you select a menu, then select a menu item.
- Menu items have 3 actions: cancel, to skip selecting a menu item; click, which selects the item; and pick, which also selects the item

```
tell "System Events"
    tell application process "Audio Hijack Pro"
        tell window “Preset List”
            pick menu item "New Preset" of menu "File"
        end tell
    end tell
end tell
```

Menu Bars & Menu Bar Items

- Menu Bars are containers for menu bar items, which are containers for menus.
- To select something in a menu bar, you need to select the menu bar, select the menu bar item, select the menu, then select the menu item.
- This is new in Panther; the beta had menu bar, menu, menu item.

```
tell "System Events"
    tell application process "Audio Hijack Pro"
        tell window “Preset List”
            pick menu item "New Preset" of menu 1 of menu
bar item “File” of menu bar 1
        end tell
    end tell
end tell
```

Text Fields

- Text fields actually are pretty simple; set the attribute "value" for the text field.
- Static text fields, natch, can't be changed; but they occur often inside of a lot of programs as labels.

```
tell "System Events"
    tell application process "Audio Hijack Pro"
        tell window “Preset List”
            set value of text field 1 to “Say hi, Bob!”
        end tell
    end tell
end tell
```

Enabled and Disabled Elements

- All elements have an "enabled" value; if it's not enabled, you can't actually change or act on it.
- Disabled elements still claim to be clickable, but are not -- you need to enable them first.

```
tell "System Events"
    tell application process "Audio Hijack Pro"
        tell window “Preset List”
            if enabled of button 2 is 0 then
                click button 1
            end if
            click button 2
        end tell
    end tell
end tell
```

Open Dialogs

- Because we're using System Events to access this program, we can't just use an "open" statement. We have to use the UI to open files.
- This combines a couple of things: clicking on an element, switching windows, entering a text field, and then click select.

Open Dialogs

- Open dialog boxes contain a text field which you can set to a file or folder location like "/Users/default/Desktop". Just don't forget to click on the "select" button afterwards.

```
tell "System Events"
   tell application process "Audio Hijack Pro"
   click menu item "Select Target..." of menu "
Control" of menu bar 1
   keystroke "/" using control down
   tell sheet 1 of window "Open"
     set value of text field 1 to "/users/stevko/
Desktop/"
     click button 1
   end tell
   click button "Open" of window "Open"
   end tell
end tell
```

Incrementors

- Incrementors are a particular type of UI element which moves values up and down
- Consist both of sliders and up/down buttons
- These can use the actions "increment" and "decriment", which do exactly like they say.
- They also can contain child buttons, which can be used in a click statement

```
tell "System Events"
    tell application process "Audio Hijack Pro"
        tell window “Preset List”
            increment incrementor 1
        end tell
    end tell
end tell
```

Clicking Elements By Location

- You can also "click" at a location on the screen.
- Call click using a list consisting of an {x,y} coordinate point
- Does not work for elements which aren't extensions of the Accessability API
- Finding the point where to click is a pain, because it's based on the whole screen, not the window.

```
tell "System Events"
    tell application process "Audio Hijack Pro"
        tell window “Preset List”
            click at {233, 533}
        end tell
    end tell
end tell
```

Keystrokes

- Finally, you can send keystrokes as well, along with any combination of the command keys (command, control, option, shift, or caps lock)
- Keystrokes can specify any key except function keys, command keys and the escape key.

```
tell "System Events"
    tell application process "Audio Hijack Pro"
        tell window “Preset List”
            keystroke "a" with {command down,
option down}
        end tell
    end tell
end tell
```

Unscriptable Elements

- Despite all of this, some elements aren't scriptable -- anything that doesn't extend the Accessability API won't be accessable.
- Custom UI elements need to have this capability built in, by using the Accessability API. Often it's not done, and the Accessability API has only been available for around a year to a year and a half.

Now, To The Actual Script....

Additional Things To Do

- Wrap this script in a perl script, save it off and run it that way, but that's for another time.
- Set up any DSPs I choose beforehand, and duplicate a preset instead of creating a new preset. Added benefit: I wouldn't have to select the channel, it could be pre-set-up as well.
- Create this as a CGI, set up a web server, and go to town.

Downsides Of GUI Scripting

- Can't register a click at a particular location!
- Scripts will be very specific to a particular program, and require a lot of searching through a program.
- Not all of the elements & attributes are accessable, like being able to access the array of children, or sometimes simple things like an element's value

Positive Benefits of GUI Scripting

- It gives applications that have no other APIs or scripting languages a way to be scripted.
- Encourages application developers to use the Accessability API, a good thing all around
- Encourages good UI development in a logical structure
- Encourages use of standard UI elements whenever possible

Future Developments?

- Obviously, more UI elements and simpler access for common elements
- Support for double-click and click and drag
- A more readable UI Element Inspector
- Extending the API so that a click anywhere on the application has the same effect as a mouse click. Hard, but worthwhile.

Questions, Comments, Insults?

The Final Applescript

```
-- An example script using the GUI Scripting beta software


-- First, set some standard variables like name and time and channel
-- This also helps, so that you don't have to set these multiple times in the script
-- when you're editing it at the command line.

property timerName : "Test Timer"
-- start time is the hour over the 24 hour day.
property startTime : 12
-- recordingTime is in minutes
property recordingTime : "45"
property channel : "kwmu.asf"
property bitrate : "64 Kbps"
property days : {1, 2, 3, 5, 6, 7}


tell application "Audio Hijack Pro" to activate
-- may not need to activate applicaton

tell application "System Events"
    tell application process "Audio Hijack Pro"
        -- first, we need to open the list of recordings and the recording times,
        -- which is called the "Presets List".

        -- To do this, we'll use keystroke. The keystroke for this is command-1
        -- it doesn't close the window if you hit it while it's open

        keystroke 1 using command down

        -- if you want to, you can click a menu item like the following line:
        -- click menu item "Show Presets" of menu "Window" of menu bar 1

        -- next, we need to create a new preset
        click button "New" of tool bar 1 of window "Presets List"

        -- I'm renaming the new item from "untitled preset" to the timer name
        tell window "Presets List"
            tell table 1 of scroll area 1
                set presetNum to get count of rows
                if presetNum is greater than 0 then
                    set value of text field 1 of row presetNum to timerName
                    set selected of row presetNum to true
                else
                    error "Something went wrong when I added an preset. Stopping script"
                end if
            end tell
            click button "Open" of tool bar 1
        end tell
```

O'REILLY®

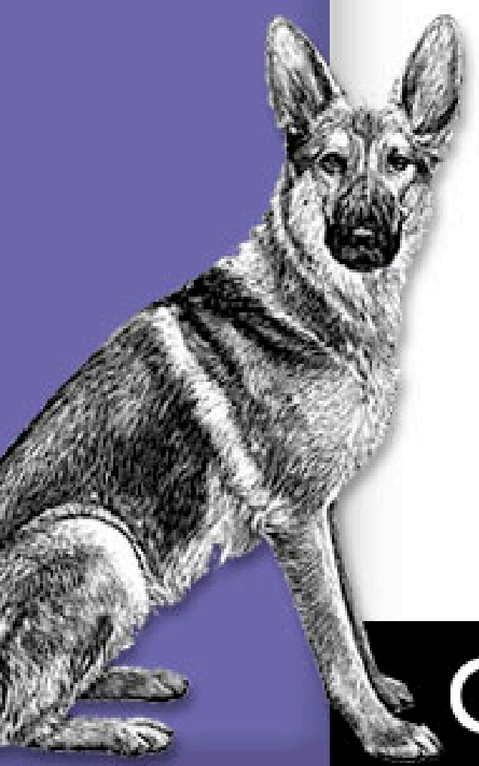
The Final Applescript cont'd

```
tell window timerName
    -- open all of the needed areas first
    if not (exists (button "Mute" of scroll area 1)) then
        click button "Control" of scroll area 1
    end if
    if not (exists (group 4 of scroll area 1)) then
        click button "Recording" of scroll area 1
    end if
    if not (exists (checkbox "Timer Enabled" of scroll area 1)) then
        click button "Timer" of scroll area 1
    end if
end tell

-- make the selections for control
click menu item "Select Target..." of menu "Control" of menu bar 1
keystroke "/" using control down
set filePath to "/users/stevko/Desktop/" & channel

tell sheet 1 of window "Open"
    set value of text field 1 to filePath
    click button 1
end tell
click button "Open" of window "Open"

tell window timerName
    -- make selections for recording
    -- change type
    tell pop up button 1 of group 1 of scroll area 1
        click
        click menu item "MP3" of menu 1
    end tell
    delay 1
    -- change stereo/mono
    tell pop up button 2 of group 1 of scroll area 1
        click
        click menu item "Stereo" of menu 1
    end tell
    delay 1
    -- change bitrate
    tell pop up button 3 of group 1 of scroll area 1
        click
        pick menu item bitrate of menu 1
    end tell
    delay 1
```



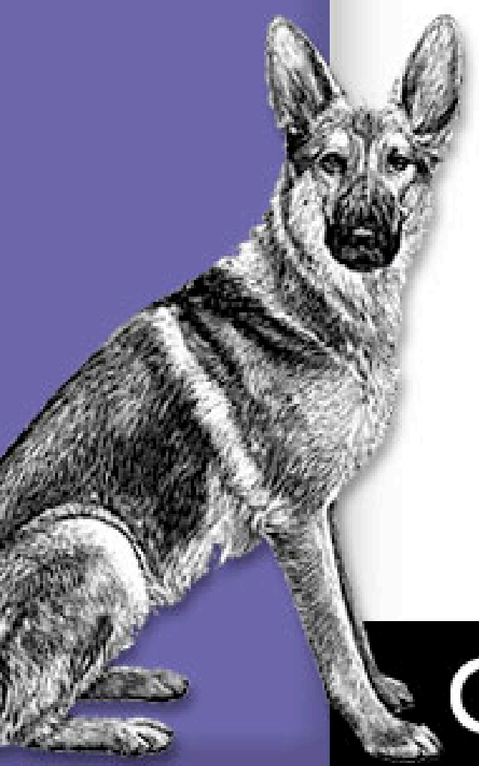
The Final Applescript cont'd

```
tell window timerName
    -- open all of the needed areas first
    if not (exists (button "Mute" of scroll area 1)) then
        click button "Control" of scroll area 1
    end if
    if not (exists (group 4 of scroll area 1)) then
        click button "Recording" of scroll area 1
    end if
    if not (exists (checkbox "Timer Enabled" of scroll area 1)) then
        click button "Timer" of scroll area 1
    end if
end tell

-- make the selections for control
click menu item "Select Target..." of menu "Control" of menu bar 1
keystroke "/" using control down
set filePath to "/users/stevko/Desktop/" & channel

tell sheet 1 of window "Open"
    set value of text field 1 to filePath
    click button 1
end tell
click button "Open" of window "Open"

tell window timerName
    -- make selections for recording
    -- change type
    tell pop up button 1 of group 1 of scroll area 1
        click
        click menu item "MP3" of menu 1
    end tell
    delay 1
    -- change stereo/mono
    tell pop up button 2 of group 1 of scroll area 1
        click
        click menu item "Stereo" of menu 1
    end tell
    delay 1
    -- change bitrate
    tell pop up button 3 of group 1 of scroll area 1
        click
        pick menu item bitrate of menu 1
    end tell
    delay 1
```



The Final Applescript cont'd

```
-- set recording value to minutes
tell pop up button 1 of group 4 of scroll area 1
    click
    pick menu item "Minutes" of menu 1
end tell
delay 1
-- set recording time
tell text field 1 of group 4 of scroll area 1
    set value to recordingTime
end tell

-- make selections for timer

-- first, select timer enabled.
if (value of checkbox "Timer Enabled" of scroll area 1 is equal to 0) then
    click checkbox "Timer Enabled" of scroll area 1
end if

if (value of checkbox "Record" of group 6 of scroll area 1 is equal to 0) then
    click checkbox "Record" of group 6 of scroll area 1
end if

if (value of checkbox "Quit Target" of group 6 of scroll area 1 is equal to 0) then
    click checkbox "Quit Target" of group 6 of scroll area 1
end if

-- setting array of days
set daysLength to length of days
repeat with counter from 1 to daysLength
    set thisDay to item counter of days as number
    click button thisDay of list 1 of group 5 of scroll area 1
end repeat

-- getting end time
set endTime to startTime + (recordingTime div 60)
if (recordingTime mod 60 > 0) then
    set endTime to endTime + 1
end if

-- setting start time
repeat startTime times
    increment incrementor 2 of group 5 of scroll area 1
end repeat

-- setting end time
repeat endTime times
    increment incrementor 1 of group 5 of scroll area 1
end repeat

        end tell

    end tell
end tell
```

O'REILLY®

Mac OS X Conference

OCT. 27 - OCT. 30, 2003 • SANTA CLARA, CA