EUDORA CLIENT ADWARE DESIGN

TABLE OF CONTENTS

Basic Concepts....3
    Definitions....3
    Privacy Concerns....5
User State Flow Diagrams....7
    Same-Version Flow Diagrams....7
    New Users, Old Users, EP4 Users....7
    Ad Users, Registered Ad Users....7
    Freeware Users and Registered Freeware Users....8
    All Users - Update Information....9
    Box Users....10
    But What If The Version Does Change?....11
Nagging....12
    Nagging Details....12
    How We Nag....12
    When Do We Nag?....13
Ads....15
    How to Display Ads....15
    When to Display Ads....15
    Ad Scheduling....16
    How to Get Ads....23
    Ad Playlists....25
    Ad Fetch Process....25
    Ad Bounties....26
    Ad Security....26
    What If We Don't Get Ads?....27
    Instrumentation & Auditing....28
    Ad Formats....28
    Clicking On Ads....29
Web Pages....30
    The Query Part....30
    URL's....31
    Payment Web Page....31
    Freeware Registration Web Page....31
    Adware Registration Web Page....31
    Box Registration Web Page....32
    Lost Code Web Page....32
    Update Page....32
    Archived Versions Web Page....32
    Profile Web Page....32
    Support Web Pages....32
Registration....32
    Registration Code....32
    Entering the Code....33
    Email Acceptance Policy....34
    Code Validation....35
    Refunds....35
    Report Bug & More Help! Items....36

Dialogs and Windows 37
- Intro Window 37
- Payment & Registration Window 38
- Pre-Registration Dialog 39
- Pre-Payment Dialog 40
- Registration Nag 40
- Send Audit Window 41
- Update Nag 41
- Freeware Downgrade Dialog 42
- Full-Feature Nag 43
- Code Entry Dialog 44
- Offline Ad/Link Dialog 45
- Link History Window 45
- Ad Failure Nag 46
- Obscured Ad Nag 47
- Deadbeat Alert 47

Appendix A - Open Issues 49
Appendix B - Preliminary Tasks and Times 50
Appendix C - Alternate Ad Placement 51
Appendix D - Freeware Sponsor Placement 52
Appendix F - Tech Support Text 54

Basic Concepts

One Application. We will produce one binary (per-platform, of course) of Eudora for use by all users. This binary will operate in one of three major modes: **Payware**, **Freeware**, and **Adware**. The payware mode is so named because they must pay us to use it. Freeware is free for all to use, but has fewer features than either Payware or Adware. Adware has all the features of Payware, but does not require payment from the user. What it does require is that the user display ads, which he will download from our site.

Ad Guidelines. We are going to attempt to download ads unobtrusively and without drawing significant bandwidth. The ads will be displayed in a way that doesn't detract from the use of Eudora.

Registration Codes. Payware users will prove their payment by a registration code that we will give them at time of payment. This code will be self-validating, and contain enough data to identify what versions the user is entitled to.

Year of Updates Policy. Users who pay for Eudora will be entitled to all versions of Eudora we produce in the calendar year following their payment.

Upgrade Notifications. Eudora will poll our site periodically to determine if an update is available, and present the user with a small web page of options.

Initial State is Adware. Few users go out of their way to change things. Rather than try to get the user to make some sort of decision to accept ads or pay, we'll just start them out accepting the ads. This should get us many more adware users than if they had to take some action to receive ads.

Definitions

In order to talk compactly about how the program will operate, we must first define some terms for user classes, program dialogs, and web pages.

Applications. We have several versions of Eudora to worry about. Here are the terms we use:

EP4	Eudora Pro 4.x, either Windows or Macintosh.
Eudora	The new three-modal version of Eudora, running in any of its modes.
Payware	Eudora running in full-feature mode, after the user has paid.
Freeware	Eudora running in reduced-feature mode.
Adware	Eudora running in full-feature mode with ads.
Paid App	Any version of Payware to which the user's registration entitles him.
Unpaid App	Any version of Payware newer than that to which the user is registered.
Old Eudora	Eudora versions prior to Eudora Pro 4.x.

User States. A user state is the most basic concept to understanding how the various modes of the application are interrelated. The user state determines how the program treats the user. The states are defined thus:

EP4 User	A user of EP4 who has not registered via the old

	registration process.
Registered EP4 User	A registered user of EP4.
New User	A user using Eudora for the first time, but who has not bought a boxed copy.
Paid User	A user who has paid for Eudora, entered his registration code, and is using a version of Eudora to which he is entitled.
Box User	This is a user who has been given their reg code by an installer, either from the box product or from an EP4 updater, and whose registration information is therefore unknown to us.
Free User	A user who has chosen to use Freeware but who has not entered a Freeware registration code.
Ad User	A user who is using the version that displays ads.
Registered Free User	A Free user who has entered a Freeware registration code.
Registered Ad User	An Ad user who has entered an Ad registration code. <u>Our business folk may need to come up with some benefit users get from registering, to make it seem worthwhile to the user.</u>
Deadbeat User	A former adware user who has been shut off due to Eudora receiving no ads.

Windows and Dialogs. Several windows and dialogs will be used in the process. A fuller description of these will be given later, but we define the major players briefly here:

Intro Dialog	A dialog presented to new users explaining to them what is going on.
Registration Nag	A window presented to the user every so often to suggest that the user register with us.
Full-Feature Nag	A window presented to Freeware users begging them to try Pro again.
Free Downgrade	A dialog that tells the user what wonderful things they'll miss if they switch to Freeware, but allows them to do so if they really wish.
Code Entry Dialog	A dialog allowing the user to enter their registration code.
Ad Window	A window or portion of screen displaying an ad.
Link History Window	A window that will display links the user has clicked on and ads the user has seen.

Web Pages. We'll need some new web pages. In general, we intend to do as much interaction via the web as practical. This will allow us the most flexibility in how we deal with our users. We list the major pages here, though these may actually turn out to be groups of pages, or pages customized to a given user.

Freeware Reg Page	A page that allows the user to register Freeware.

Payware Reg Page	A page that takes money for Pro and returns a registration code.
Adware Reg Page	A page that allows users of adware to give us their info.
Lost Code Page	A page that helps users who have lost their registration codes. May require the intervention of a human.
Update Page	A page generated for a user that lists possible upgrades and the latest version for which he is registered.
Archived Versions Page	A page from which users can download all versions of Eudora.
Profile Page	A web page where users can enter their profile information.

Nag Schedules. A "nag schedule" is a bracketed set of numbers. The numbers signify # of days since the start of a trial period. Users will be nagged on the days indicated. The last number signifies what happens when the other numbers run out; the user will either not be nagged (0), or be nagged every so many days. For example, a schedule of [0,5,2] means the user will be nagged on the first day, the sixth day, and every other day thereafter.

Privacy Concerns

One thing of which we must be very careful is the protection of the user's privacy. There is an extremely vocal and paranoid subset of the user community, who object to practically all forms of information gathering, even the most benign. Even relatively innocent devices like serial numbers are considered Completely Evil. While this may be silly, it is the world we inhabit, and we would do well to pay attention to it. I suggest that we maintain the following principles in regard to privacy:

Obtain Permission. Before we gather or transmit any data that might identify the user, we must get the user's near-explicit permission. I use the term *near-explicit* to mean that I do not believe we need, for example, to put a special privacy warning in the web page where the user registers Eudora. Here, the user is clearly taking an action to give us data, and explicit permission shouldn't be needed. On the other hand, we should go out of our way to identify areas where an unreasonable user might be able to claim that he didn't know he was giving us information, and ask for explicit permission there, even if it seems relatively obvious to us.

Data Separation. Insofar as possible, we should keep our payment information separate from our registration information separate from our demographic information, etc. I know it is very tempting to correlate these databases, but we will be crucified if we do, and we can still deliver very targeted advertising if we do not. So let us avoid such correlation.

User Verifiability. Insofar as possible, our protections should be verifiable by end users with packet sniffers. We may even encourage the practice of watching Eudora's actions. It is one thing to say "We do not give your personal data to advertisers," it is quite another for the user to be able to verify that this is the case.

Strong Public and Private Commitment. We need to be clear and public with our policies, and we need to respect them internally. If we view privacy as something we must do to avoid being in hot water with the press, we will do it poorly and wind up in trouble. However loony all this seems, we must learn to leave our senses ourselves.

User State Flow Diagrams

Same-Version Flow Diagrams

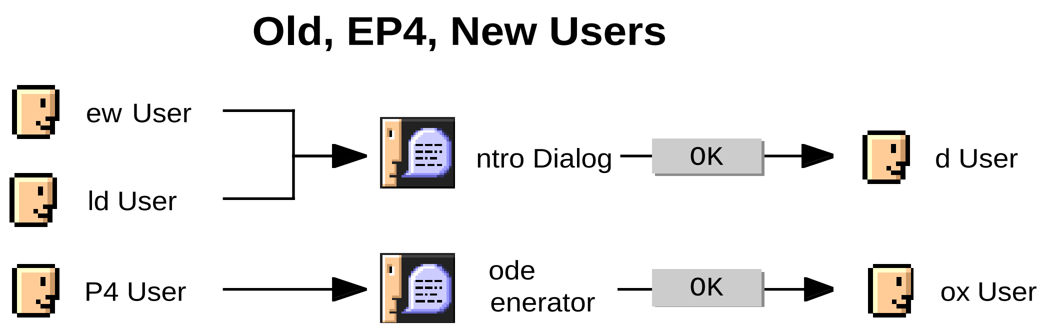
On the principle that we must start somewhere, there now follow some flow diagrams that show the major user states and the transitions among them, assuming the user does not install a new version of Eudora. Flow resulting from the user changing versions of Eudora will be considered separately.

Diagram Conventions.

- Raised grey squares are conceptual names for buttons in dialogs.
- A few paths are labelled with menu items. These items can be used to bring up the window in question directly, without waiting for nags.
- In principle, *any dialog or nag can be cancelled*, leaving the user back in the initial state.
- Web pages cannot change user state or generate more dialogs; hence, all web pages lead back to the user's initial state.

New Users, Old Users, EP4 Users

The following diagram shows what happens for first-time users of this new version.

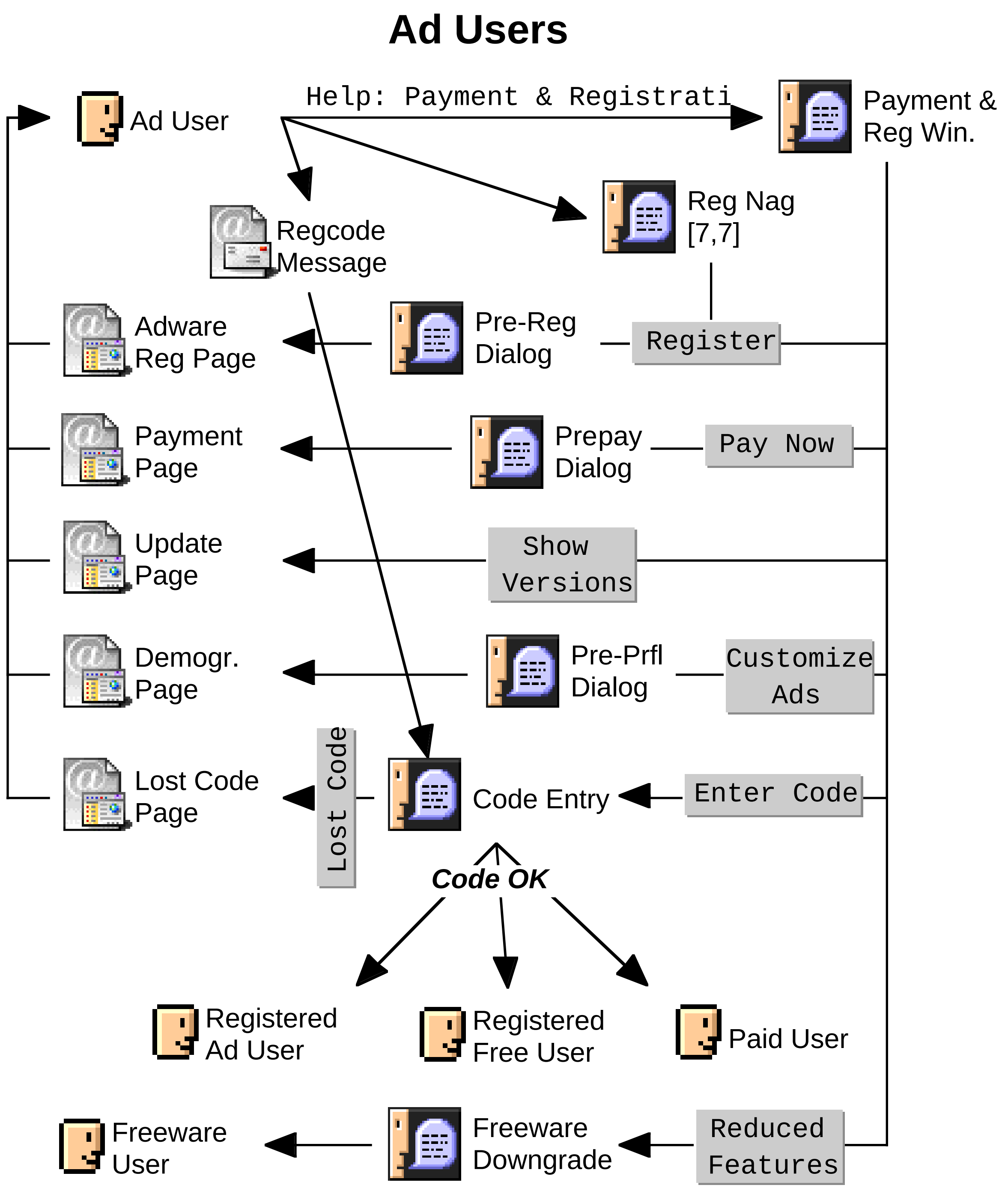


Note that we don't give the user options to pay or reduce features in the intro dialog. While we will explain those options and that the user can get to them from the Help menu, we won't give them buttons here, because of the possibility of a knee-jerk reaction. Let them first see that the ads aren't going to end the world; if they then go out of their way to turn off the ads, so be it.

The path taken by EP4 users & Box Purchasers bears elaboration. The Code Generator referred to happens *in the installer*, not in Eudora. If the user is using the 4.x->4.3 updater, it searches for a copy of EP4, and on finding one it allows the user to generate a reg code file. If the user is running the installer from the box, it allows reg code generation without looking for a copy of EP4 first. The reg code file so generated is special in that it contains a line saying "Eudora-Needs-Registration: Yes". Eudora will notice this, and put the user in the unregistered state, and nag them to register. Once they do register, the same registration code will be retransmitted to them, Eudora will silently accept it (since it will be the same as their current code), and turn off the need to register flag.

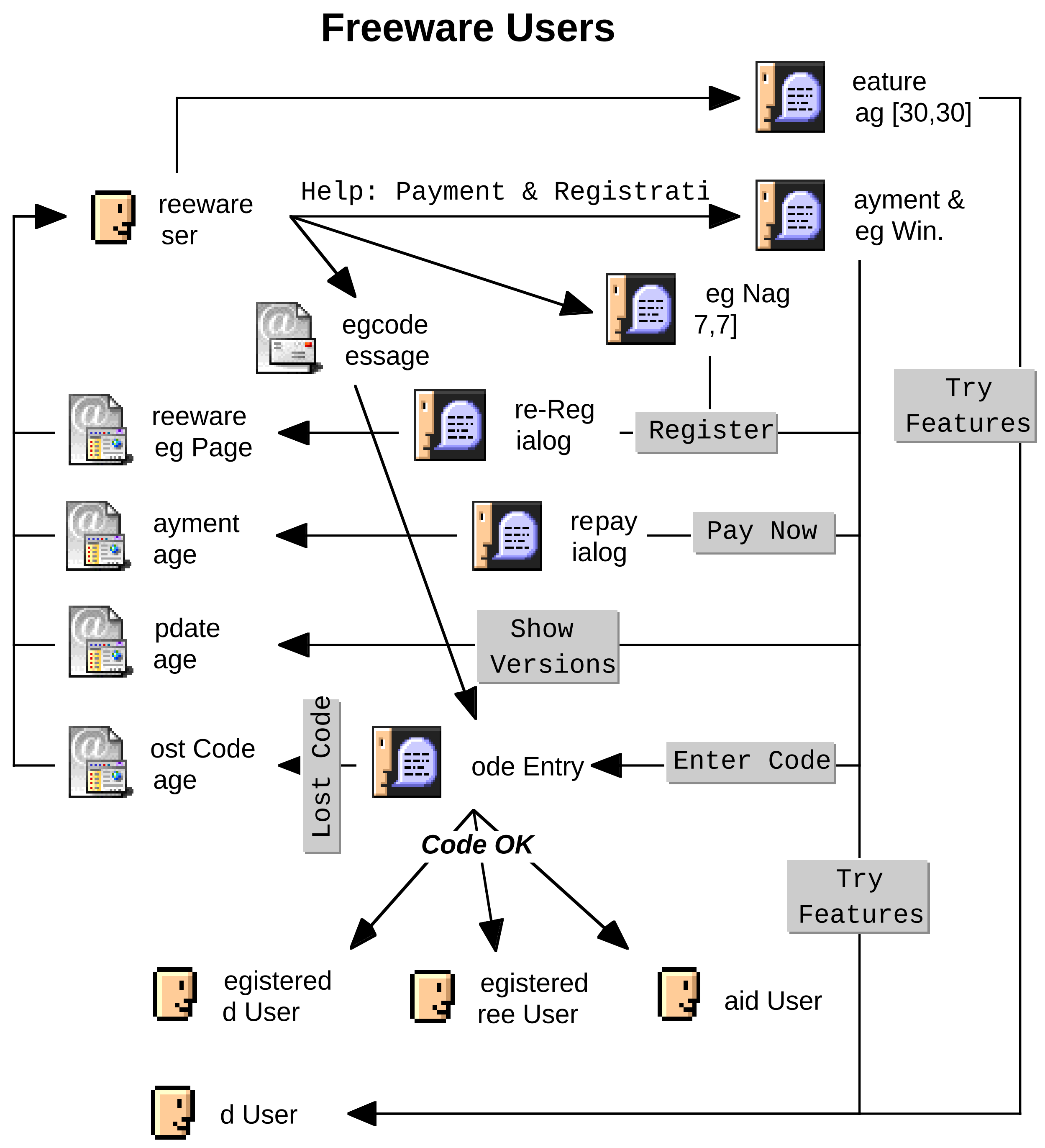
Ad Users, Registered Ad Users

Once we have an Ad User, we're not in a hurry for that user to change modes. So the only nag an ad user is subject to is one that asks him to register with us. Registered Ad Users have the same diagram, except that they are not subject to the registration nag.



Freeware Users and Registered Freeware Users

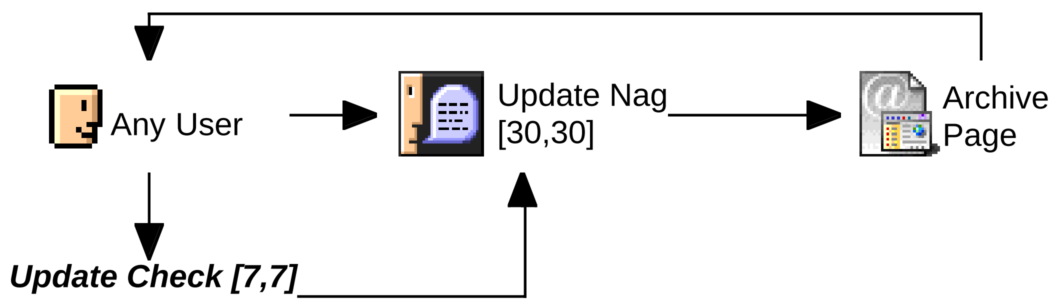
The following diagram covers both Freeware Users and Registered Freeware Users. The only difference is that the registration nag is not applicable to Registered Freeware Users. This is very similar to the Adware Users, with the exception that Freeware Users are given the option to try the full features rather than enter their demographics.



All Users - Update Information

Simply every user is allowed to get this particular nag. Eudora checks the Update Page once per week during a mail check. If the page has changed, the user is nagged. Even if the page hasn't changed, the user is nagged on a 30-day schedule.

Update Nags

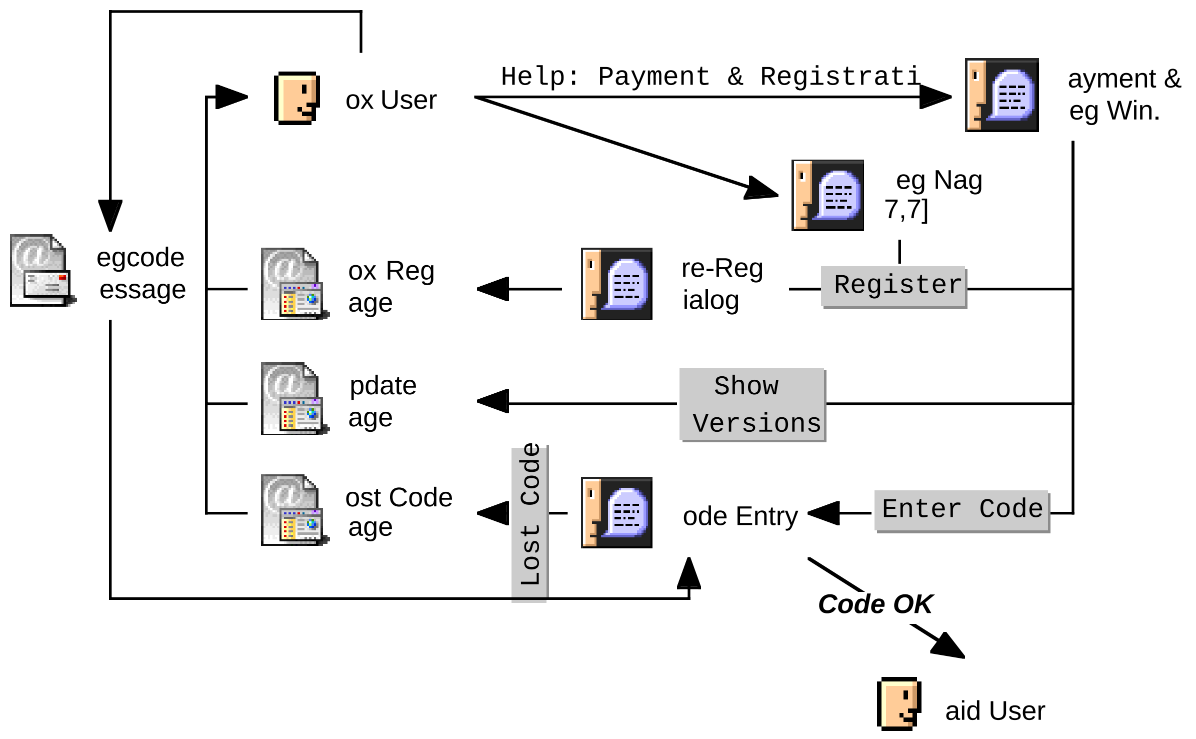


The update nag presents the user with versions to which he is entitled to upgrade (if any). The nag itself is an HTML document with links to versions for him to download.

Box Users

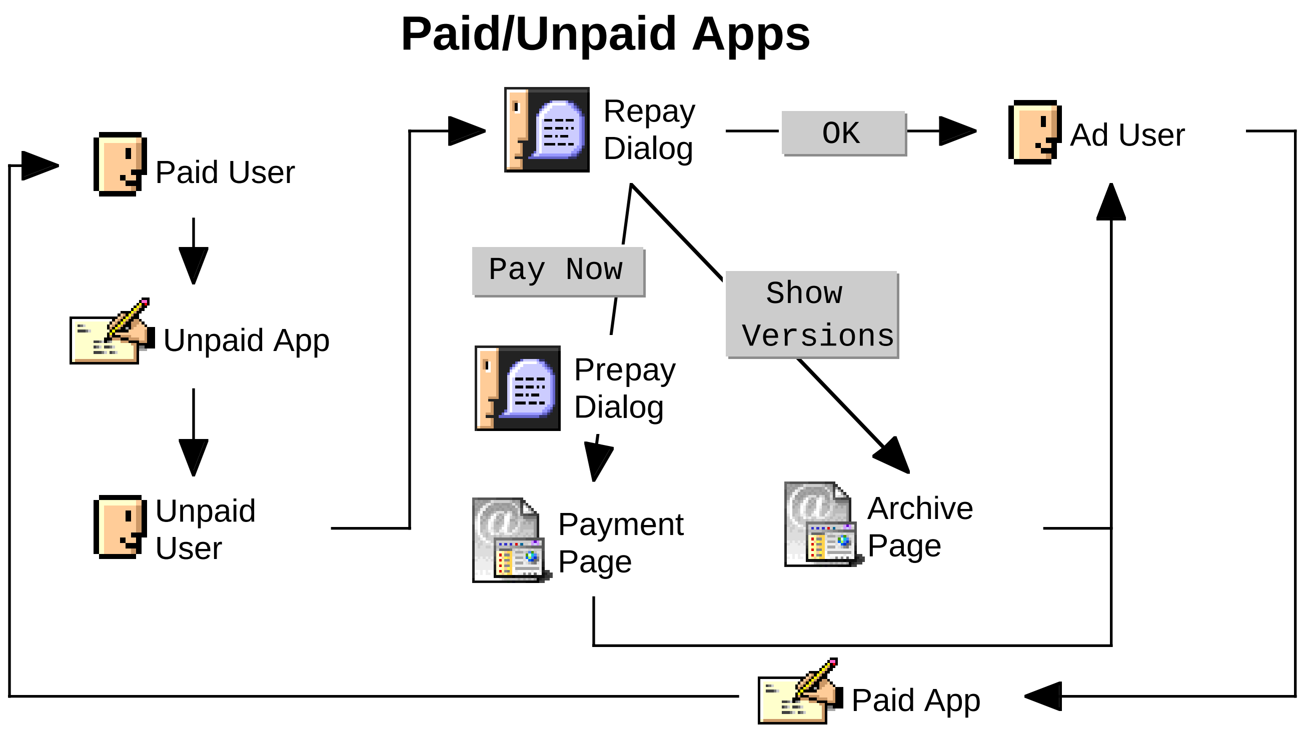
The only nag we give specifically to box users is the registration nag. Once they register, they're converted to normal paid users. However, their payment date is set to a specific value, so that we can control what versions we give them for free in the future.

Box Users



But What If The Version Does Change?

The only version upgrade that matters to us is that of a Paid User to an Unpaid App (that is, an application past the date at which he is entitled to free upgrades), and back again.



Nagging

Nagging Details

There are two major things we need to know about nagging the user. One is how we nag the user, and the other is when we nag the user.

How We Nag

Window Properties. Ideally, nag windows are modeless windows. The user can close them using close boxes, or dismiss them by taking one of their action items, or simply leave them open and let them drift wherever they will in the window list. Due to implementation constraints[1], Windows nag windows will be slightly different in behavior. They will be floating windows, which we expect the user will probably dismiss in fairly short order. They will not, however, stop background tasks from executing.

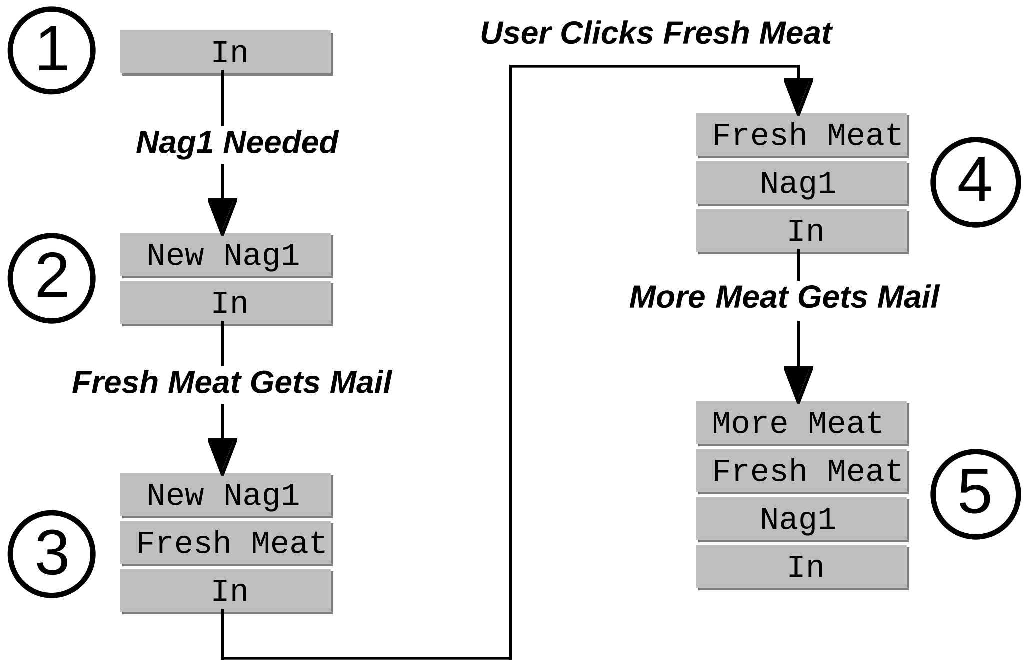
Renagging. There is at most one nag window of each variety open at a time; old windows of the same variety should be recycled. That is, if a given nag window is still open the next time the user is due to be nagged, that window will be reused and brought back to the top of the window stack.

Nag Me, Please. All nags applicable to the user should be available from the Help menu, so that the user who dismisses one inadvertently can deliberately annoy himself if he wishes, though such manual nag invocations do not reset the nag's timer.

Window Placement. Nag windows will be opened on top of all other windows, and automatically opened windows (except other nags) will never be placed above them until the user has manually brought another window above them. Due to the implementation constraints in Windows Eudora, the only windows that can obscure nags would be other floating windows.

[1]Chiefly the requirement that MDI child windows be maximizable, which makes no sense for dialog-like nag windows.

Mac Nag Window Placement



In our initial conditions (1), we have just the In mailbox. Eudora determines it needs to nag, and places the nag atop the mailbox (2). Some mail arrives in the Fresh Meat mailbox. Ordinarily, this would open on top, but since there is a “new” nag, one the user has not manually sent behind anything, Fresh Meat instead opens below the nag (3). The user manually brings Fresh Meat to the front (4). After that, when mail arrives in More Meat, the nag is no longer new, and More Meat can be opened on top as normal (5).

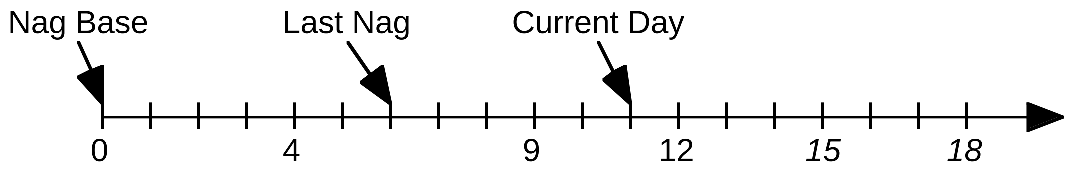
Windows Nag Window placement is in general considerably simpler. Nag windows float outside the MDI box[2], above other floating windows, until the user closes them. The exception to this rule is the Update Nag, which acts as a Macintosh Nag window, if you assume that the entire Macintosh diagram takes place inside an MDI box. Note particularly that this means the Update Nag may be maximized.

When Do We Nag?

Nag Schedules. These were introduced earlier in the document. It is now time to discuss them in more detail. Each schedule is a set of numbers representing (save for the last) the number of days since a given date (the *nag base*). We further must keep track of the last time the user was nagged (the *last lag*). Note that both the nag base and last nag should be tracked separately per type of nag; we do not mix values for registration nags and update nags, for example. The last number of the nag schedule is a repeat interval. Once the other nags are all exhausted, the user is nagged each time the final number of days passes.

The best way to understand a nag schedule is to view it as a timeline:

[2] I’ll be more than happy to explain to you why the MDI “frame” is a 3-dimensional object.



This timeline is for a nag schedule of [0,4,9,12,3]. Note that the nags at 15 and 18 are there because of the final number, the repeat interval (of 3 days).

A user is due to be nagged if there is a nag day greater than the last nag and less than or equal to the current day. If more than one nag day has passed, the user is still nagged only once.

Once the nag window has been opened, the last nag is reset to the current day.

A final nag interval of 0 indicates that the nag is not done after the period expires.

Nag Checking. We'll check to see what nags are due at application startup and at the completion of each mail check.

Update Nag Special Processing. Once per week during a mail check, we will check the modification date on the Update Page. If it has been modified, we will download it during the mail check, and nag the user.

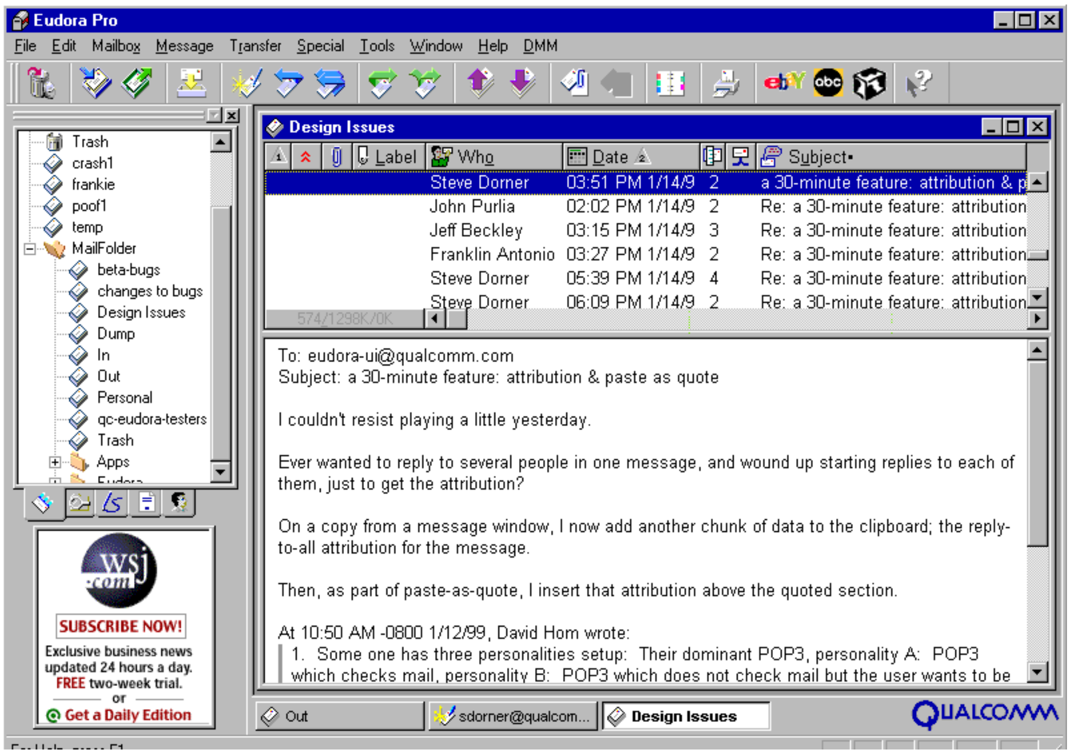
Auto-Dismissal. When a user's state changes so that an open nag is no longer relevant, that nag should be closed.

Ads

The major client issues involving ads are how we display them, when we display them, how we get them, how we obtain and transmit demographic information, and how we verify that things have not gone terribly, terribly wrong.

How to Display Ads

Here we show a squarish[3] ad and three ad buttons. The mailboxes area is reduced 38%, but the content area is left untouched.



Label	Who	Date		Subject
	Steve Dorner	03:51 PM 1/14/9	2	a 30-minute feature: attribution & p
	John Purlia	02:02 PM 1/14/9	2	Re: a 30-minute feature: attribution
	Jeff Beckley	03:15 PM 1/14/9	3	Re: a 30-minute feature: attribution
	Franklin Antonio	03:27 PM 1/14/9	2	Re: a 30-minute feature: attribution
	Steve Dorner	05:39 PM 1/14/9	4	Re: a 30-minute feature: attribution
	Steve Dorner	06:09 PM 1/14/9	2	Re: a 30-minute feature: attribution

574/1298K/0K

To: eudora-ui@qualcomm.com
Subject: a 30-minute feature: attribution & paste as quote

I couldn't resist playing a little yesterday.

Ever wanted to reply to several people in one message, and wound up starting replies to each of them, just to get the attribution?

On a copy from a message window, I now add another chunk of data to the clipboard; the reply-to-all attribution for the message.

Then, as part of paste-as-quote, I insert that attribution above the quoted section.

At 10:50 AM -0800 1/12/99, David Hom wrote:
> 1. Some one has three personalities setup: Their dominant POP3, personality A: POP3 which checks mail, personality B: POP3 which does not check mail but the user wants to be

One troublesome issue regarding our proposed ad placement is the ease with which a user might be able to hide the ads. Our basic response to this will be to check that the ad is onscreen and uncovered. If it is not, we will ask/help the user to display it, and if the user persists in covering the ad we will devolve to freeware mode.

Please see Appendix C for some other ad size/placement possibilities that were considered (but discarded).

When to Display Ads

The sophistication with which me might eventually decide when to display ads is virtually unlimited. I am at this point wary of promising too much and not being able to deliver. So, I suggest that we lay the groundwork as best we can for a sophisticated system, but initially implement only a simple one.

Face Time. One thing we clearly want to do is do our best to ensure that Eudora users are actually looking at the ads. Time with the ad on the screen and the user in the bathroom is

[3]This particular ad is144 pixels high by 128 pixels wide; we will allow up to 144 pixels by 144 pixels.

not very worthwhile. So we will attempt to measure the actual time the user in front of the computer while the ad is present.

Lacking some sort of positive ocular fastening device, the best thing we can do to measure user attention is watch for user input. The primary user input devices are the mouse and the keyboard[4]. Thus, we will look for the user actually controlling mouse or keyboard when Eudora is frontmost.

To wit, the user will be considered "present and accounted for" if the mouse moves significantly,[5] if the mouse button states change, or if keys are pressed or released. We will consider a period before and after such an event as "face time" for the ad. Let us call the total length of this period *kFaceInterval.* There is no need to be greatly precise over this value, so we will assume for the moment that kFaceInterval will be 60 seconds, and will begin with the user event.[6]

Ad Parameters. We wish to offer a variety of options for advertisers. The parameters, which include how long to run the ad, where to go when the ad is clicked, what dates to run the ad, etc, are specified in detail in the separate "PlayLists and You" document. Many of these extra parameters could be used to generate extra charges. It's perfectly reasonable for us to charge extra to chain ads, or to keep an ad consecutively onscreen.

I urge restraint in our initial implementation; the more factors we use, the more difficult the system will be to implement, test, and verify once it's in operation.

Changing Ads. The only issue I can see is the fact that the transition will draw attention to the ad. This can be viewed as either a benefit or a drawback. It's a benefit because there is some extra chance that the user will pay attention to the new ad. The drawback is that the user might be as distracted by this as by an animated ad, and be less likely to continue with ads enabled. I have no clear idea on which view is more correct. It may even make sense to make this a per-ad parameter, and charge extra to guarantee that the user is looking when the ad is changed.

AD SCHEDULING

Given a set of available ads, we need to choose which ad to display next. This is a matter of much excitement in the web ad industry, and many choices are allegedly made to maximize the profit of the advertiser. Ads that generate better user response are preferred because they generate extra revenue.

It's unlikely that we'll be able to get direct benefit from the ad scheduling algorithms currently run on ad services. This is in part due to the fact that our ads are divorced from the content being displayed (ie, we don't know that the user is looking at a message about dogs, unlike the owners of the Purina Dog Chow web site), and in part due to the fact that we will be requesting ads in a batch for later display, rather than requesting them in "real time".

The playlists provide certain global inputs to the ad scheduling algorithm:

faceTimeQuota	The amount of time per day we are supposed to show regular ads.

[4]Though in future we may need to worry about microphones or other devices.

[5]That is, more than the pixel or two that might result from, say, bumping the desk with the back of one's chair.

[6]Clearly the actual user attention begins before the event, but this turns out to be something many people simply can't "get", and as it doesn't really matter I'm not going to push the point.

rerunInterval	The age beyond which ads should not be "rerun" after the runout time is passed.

The playlists' per-ad inputs associated with ad scheduling are:

showFor	This is the number of seconds the ad should be shown for at any given time. This number might be small, like a TV ad (eg, 30), or large, more like a billboard (eg, 3600 for an hour all at once).
showForMax	Maximum total amount of time to show this ad. The ad is exhausted after this time, and should be discarded once new ads arrive.
dayMax	Maximum number of times per day to show this particular ad.
blackBefore	The amount of time the ad window should be blank before the ad is displayed.
blackAfter	The amount of time the ad window should be blank after the ad is displayed. BlackAfter runs concurrently with the blackBefore of the next ad, so that the actual time between ads is max(blackAfter,blackBefore), not blackAfter+blackBefore.
startDT	Date/time (timezone optional) before which the ad should not run.
endDT	Date/time (timezone optional) after which the ad should not run.

There are some values we compute that are also input to the scheduling algorithm. The global ones are:

adFaceTimeToday	The amount of time today that we have shown regular ads.
totalFaceTimeToday	The amount of time today total.
currentBlock	If we're showing ads from a playlist marked "block", this will point to the current block.
blockGoal	The number of showings we're currently looking for in a playlist marked "block".

We also keep track of these values for every ad:

numberShownToday	The number of times an ad has been shown on this calendar day.
thisShowTime	The amount of facetime the current ad has been given.
lastShownDate	The last time we showed and charged for this ad

We have three major states of the ad scheduler:

regularState	We are showing our regular ads and accounting for them. This is what actually generates charges for the bulk of our ads.
runoutState	We have shown enough regular ads to fill our faceTimeQuota, and we have one or more runout ads,

	which we are showing.
rerunState	We have exhausted our regular quota, and we have exhausted our runout ads, and we are now reshowing our regular ads, but not charging for them.

Here follows an algorithm, god help us all. It is not intended to dictate implementation, merely to describe the desired behavior:

```
////////////////////////////////////
// Main ad scheduler
ScheduleMain
{
   // Has a new day dawned?
   Do CheckForNewDay

   // Are we are within the current ad's showFor?
   if ( ad.thisShowTime < ad.showFor )
   {
      // there is nothing to be done
      return
   }

   // At this point, we know that we need a new ad

   // Perform housekeeping tasks on the old one
   Do AdEndBookkeeping

   // Pop out of a block if all ads on par
   if ( block isn't all playlists )
   {
      find ad with minimum ad.numberShown
      if ( ad.numberShown >= blockGoal )
         set block to all playlists
   }

   // If we are over our quota of regular ads for the day,
   // look for a runout
   if ( adFaceTimeToday > faceTimeQuota )
   {
      Do ShowARunout
   }
   else
   {
      Do ShowARegularAd
   }
}
// end ad schedule main


////////////////////////////////////
// We must perform certain tasks when the calendar day changes.
CheckForNewDay
{
   if ( the calendar day has changed )
   {
      // Perform housekeeping tasks on the ad currently showing
      Do StopShowingCurrentAd

      // Runout ads are charged for a full showFor if they've been shown
      // at all on a given day.  Charge any runout ads if they've been
      // shown at all.
      for runout ads
      {
         if ( ad.thisShowTime > 0 )
         {
            ad.totalTimeShown += ad.showFor
            ad.thisShowTime = 0
         }
      }

      // Now, reset the counters for all ads to reflect the fact that
      // a new day has dawned.
      for all ads
      {
         ad.numberShownToday = 0
      }

      // Record yesterday's facetime
      // Might not literally be yesterday, be sure to use
      // whatever day the app was last run on
      set old current day's facetime to totalFaceTimeToday

      // and reset our global regular ad facetime counter
      adFaceTimeToday = 0
      totalFaceTimeToday = 0

      // if we were in a block, back out
      set block to all playlists
   }
}
// end CheckForNewDay


////////////////////////////////////
// This function shows a runout ad, and if it
// can't find one, goes to a rerun
ShowARunout
{
   for runout ads
   {
      // has the ad been flushed?
      if ( ad.flushed )
         try next ad

      // are we done showing this runout today?
      if ( ad.numberShownToday > ad.dayMax )
```

```
            try next ad // this one's used up for the day

         // are we done showing this runout for ever and ever?
         if ( ad.shownFor > ad.showForMax )
            try next runout ad // this one's used up forever

         // are we between the ad's start and end dates?
         if ( ad.startDate < the current date < ad.endDate )
            try next runout ad
      // the ad is not supposed to run today

         // do we actually HAVE the ad?
         if ( ad has not been downloaded )
         {
            ask for ad to be downloaded
            try next ad
         }

         // ok, we believe we should show this runout
         // we are now in runout state
         Do ShowAnAd
         return
      }

      // if we haven't found a runout ad, we will go to "rerun" state
      Do ShowARerun
   }
   // end ShowARunout


   ////////////////////////////////////
   // Rerun state.  Look for a regular ad to rerun
   ShowARerun
   {
      for regular ads [ in current block ]
      {
         // has the ad been flushed?
         if ( ad.flushed )
            try next ad

         // is this ad recent enough to rerun?
         if ( ad.lastShownDate is older than returnInterval )
            try next ad
      // this one is too old to rerun

         // if in block, show ads only if it's their "turn"
         if ( ad.numberShownToday >= blockGoal )
            try next ad // need to find a friend in this block

         // are we between the ad's start and end dates?
         if ( ad.startDate < the current date < ad.endDate )
            try next ad
      // the ad is not supposed to run today
```

```
         // do we actually HAVE the ad?
         if ( ad has not been downloaded )
         {
            ask for ad to be downloaded
            try next ad
         }

         // ok, at this point we can show this ad, but because
         // we're in rerun, we don't keep the books
         Do ShowAnAd
         return
      }

      // if we get here, we have no ads to show.  Punt.
      return
   }
   // end ShowARerun

   ////////////////////////////////////
   // Show a regular ad
   ShowARegularAd
   {
      for regular ads [ in current block ]
      {
            // has the ad been flushed?
            if ( ad.flushed )
               try next ad

            // are we done showing this ad today?
            if ( ad.numberShownToday > ad.dayMax )
               try next ad // this one's used up for the day

            // if in block, show ads only if it's their "turn"
            if ( ad.numberShownToday >= blockGoal )
               try next ad // need to find a friend in this block

            // are we done showing this ad for ever and ever?
            if ( ad.shownFor > ad.showForMax )
               try next ad // this one's used up forever

            // are we between the ad's start and end dates?
            if ( ad.startDate < the current date < ad.endDate )
               try next ad
      // the ad is not supposed to run today

            // do we actually HAVE the ad?
            if ( ad has not been downloaded )
            {
               ask for ad to be downloaded
               try next ad
            }
            // ok, we believe we should show this ad
            // we are now in regular state
            Do ShowAnAd
            return
      }

      // If we get here, we have failed to find a regular
      // ad.  Go to runout
      Do ShowARunout
   }
   // end ShowARegularAd

   ////////////////////////////////////
   // Perform necessary housekeeping when we're taking
   // down an ad
   AdEndBookkeeping
   {
      // In rerun state, we don't do any bookkeeping
      if ( in RerunState )
         return

      // Account for at most ad.showFor seconds, provided
      // we've shown the ad for at least ad.showFor seconds
      // Note that this means we don't charge for time beyond
      // ad.showFor seconds, which is important
      if ( ad.thisShowTime >= ad.showFor )
      {
         ad.numberShownToday += ad.showFor
         ad.shownFor++
         // we do NOT reset thisShowTime here, we do it in
         // AdStartBookkeeping.  It actually doesn't matter where
         // we do it, provided we are careful NOT to do it for
         // runout ads.
      }
   }
   // end AdEndBookkeeping

   ////////////////////////////////////
   // Show an ad, including bookkeeping and block handling
   ShowAnAd
   {
      // If the ad is in a block, notice that
      if ( it's in a "block" playlist )
      {
         if ( not currently in a block )
         {
            find ad in block with minimum numberShown
            make that our ad
            set blockGoal to minimum numberShown+1
         }
         set current block to this playlist
      }
      // now do bookkeeping
      Do AdStartBookkeeping

      // and actually show it
      Do DisplayThatAd
   }

   ////////////////////////////////////
   // Perform housekeeping when we put up an ad
   AdStartBookkeeping
   {
      // In rerun state, we don't do any bookkeeping
      if ( in RerunState )
         return

      // For regular ads
      if ( it's a regular ad )
      {
         ad.thisShowTime = 0
         ad.lastShownDate = now
      }
   }
   // end AdStartBookkeeping
```

How to Get Ads

The general outline of the plan is to connect to a QUALCOMM site during a mail check or other time when we know there is a live network connection and download ads into a local cache. The act of downloading the ad will be the trigger for billing the advertiser, in order to avoid the necessity of collecting billing information from individual clients.

Assumptions. In order to make reasonable decisions about how to download ads, we need to have some idea of what impact the ad downloads will have on users. In order to assess that impact, we must make assumptions (or gather information) about what a typical Eudora user's habits are, and what our ads will be like.[7] For now, these are our assumptions:

[7]Part of the adware process will be to add instrumentation in the client so that we can begin to answer these questions intelligently, rather than by guesswork.

Assumptions	
Average Connect Speed, Kbps	28.8
Average Ad Size, Kbytes	9.3
Number of Users	8,000,000
Number of Hours Running Eudora	2
Number Mailchecks Per User Per Hour	2
Playlist Entry Size, Bytes	500

Calculations. Based on these assumptions, we get these numbers:

Implications

# of New Ads Per User Pe Day	# Seconds Downloadi ng Ads	# Seconds Added Per Check	8M Users Ad Bandwidth , Mbps	Ad Mbps / 100,000 users	Avg Sim. Connectio s, 1000'	8M Users Playlist Bandwidth , Mbps	PlayList Mpbs / 100,000 users
10	26	6	67	0.8	2.4	4	0.0
15	39	10	101	1.3	3.6	5	0.1
20	52	13	135	1.7	4.8	7	0.1
25	65	16	168	2.1	6.0	9	0.1
30	78	19	202	2.5	7.2	11	0.1
35	90	23	235	2.9	8.4	12	0.2

Our current goal is for an average turnover of an ad in three days, so that the top line in the chart would be the one we use. Our worst-case scenario would be to turn over 25 ads a day; above that we do not currently choose to go. These values are highlighted in the table.

AD PLAYLISTS

In order to determine what ads are to be shown for a particular user class, as well as in order to transmit particular ad parameters, we will use a *playlist.* The playlist is in its essence a list of URL's from which to fetch the actual ads as well as a set of attribute-value pairs, on a per-ad basis. The exact format of the playlist is specified in a separate document.

Playlists will specify the complete set of ads the client should have, along with parameters for displaying those ads, as discussed above. Note that ads may appear in a playlist but not be scheduled for display for a long time (or even at all). The presence of such ads in the playlist will push the ads to the clients or hold ads on the clients for future display.

The request for a playlist will contain information to help the playlist server determine what ads a copy of Eudora needs to fetch.

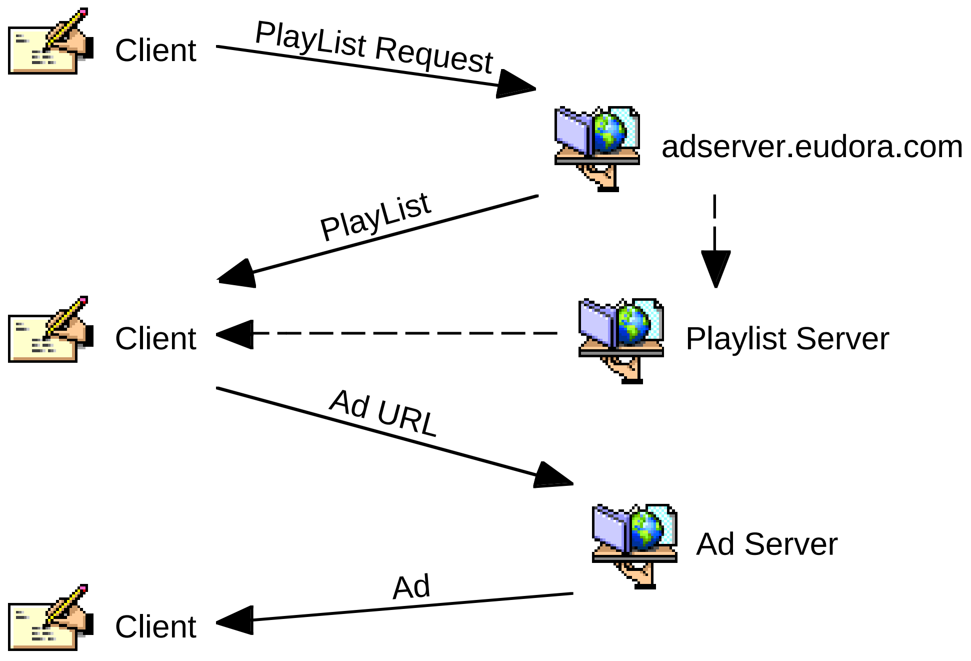
The playlist can also contain parameters for Eudora as a whole, including the ability to modify how often new playlists are checked for.

Finally, playlists are allowed to specify whether or not they should replace all older playlists or merely be merged with them. The merge function will allow a more web-like advertising model, should we choose to use it.

Playlists are more completely specified in a separate document.

Ad Fetch Process

The basic ad fetch process is as follows:



Basic Process. First, the client identifies itself to ads.eudora.com, providing basic client information and the id of the play list it currently has. Ads.eudora.com responds either with an indication that the current playlist is still valid, uses an HTTP redirect to send the client to a different playlist server, or responds directly with the playlist.

Finally, the client compares the new playlist with its current set of ads, and begins fetching ads from ad servers according to URL's in the playlist. It also deletes ads not currently appearing in the playlist.

How Often? We will check for a new playlist every three[8] days. Ads will be fetched as needed to fill the playlist, possibly over many mailchecks.

Deadman Timer. The ad fetch process will be limited to one minute, no matter what. After one minute, we rudely disconnect. This may mean that we have not filled the playlist. That's ok, we'll go with what we have until the remaining ads are downloaded.

Multiple Servers. We should allow for multiple servers on a peer with ads.eudora.com. These servers will provide extra ads for some Eudora user communities. Perhaps all the users at a company, with one ISP, etc. It's unclear how we make multiple servers easy for the community to configure but not easy for individual users to configure, or indeed if we try.

Ad Bounties

We will provide a custom installer to various ISP's, book publishers, etc., that will label the copies of Eudora they have given out. We will then give these distributors a percentage of the ad revenue generated by the clients they distribute.

Ad Security

There are really two security issues to consider. One is whether or not the client is getting valid ads (call this *client security*) , and the second is whether or not a valid client is fetching ads (call this *server security*).

Client Security. Client security is of relatively small importance. If a given person manages to trick Eudora into displaying some ads other than our own, it doesn't matter a great deal. It could of course be problematic if large sites began doing it; however, a carefully worded license agreement should make large sites sufficiently afraid of our lawyers to avoid this particular problem. To avoid trivial attacks, playlists and ads will be checksummed with SHA-1 and a secret seed, and the checksums recorded in the playlist. Then the client can checksum the playlist and ads using the same secret seed, and compare its checksums to those in the playlist. If it fails to get the proper ads, this will be treated as failure to get ads at all.

Server Security. This is the really big question. We intend to charge our advertisers for ads, on the understanding that our users will actually see the ads we're charging for. To do this with confidence, we need to make sure that it is actually Eudora that is downloading the ads, and not some rogue process written to fetch many ads.

Why would someone bother to fetch ads? While we can't discount the "because they can" motivation of the amateur hacker, the real issue is the ad bounty. Because every ad fetch can generate revenue for a third party, there is a very significant financial incentive for that third party to cause a lot of ad fetches.

It thus becomes imperative that we prevent (and/or detect in a legally important way) ad fetches not made by copies of Eudora. Given that such fetches would be in violation of the agreement we signed with the distributor, these fetches would be a form of fraud, and we will use that term in our discussion.

Possible Approaches to Fraud Detection. Whatever method we eventually use to prevent fraud, it will be important also to detect fraud should it occur. There are two broad classes of fraud detection; authentication and statistical analysis. Authentication is easily understood; if the program fetching the ads fails to prove that it is a valid copy of Eudora, we will be alerted

[8]Whether 3 days is the proper interval between playlist checks is unclear.

to possible fraud. However, as we will discuss later, authentication provides challenges of its own, and may be impossible or impractical or simply unnecessary.

Statistical analysis has some great benefits, but also significant drawbacks. The benefits include minimal work in the client (and hence no vulnerability to disassembly, etc.), no run-time burdens on client or server (everything can be done "after the fact" during accounting runs), easily changeable from our end, ability to be applied retroactively, etc.. The drawbacks to statistical analysis include that it will never be entirely certain, and that we may not collect the proper statistics, etc..

So precisely what statistics might we gather or compute? Those that come to mind are listed below:

clientID	It's hard to see a way to avoid generating some sort of client id for use with fetching ads. We might hope that such identifiers will be self-validating, but it seems clear that we will need to be told what particular installation of Eudora is actually fetching ads. This can then be used in statistics and computations. By "installation" we probably mean a single folder with mail in it.
ipAddress	We will likely want to log requests by IP address.
distributorID	Of course a cornerstone of the whole bounty thang is the fact that we will record the distributor ID for the client fetching ads. It may not also be obvious that we should collect this when users pay or even register our software, but we probably must.
numPaidUsers	This statistic is the number of paid users with a given distributor ID.
numClientIDs	This statistic is the number of client id's with a given distributor ID.
numAdsFetched	The number of ads fetched by a particular client ID.

So much for the raw data. Here are some interesting applications of these statistics:

numAdsFetched	A client id with a very high number of ads fetched is suspicious.
numClientIDs/numPaidUsers	Paid users is a very hard number, because we have collected a credit card and charged it. Thus, it can serve as a useful measuring stick for how many clients we expect. A particular distributor with a very high ratio or a ration that goes suddenly higher bears investigation.

What If We Don't Get Ads?

Users or ISP's may simply shut off the flow of ads to Eudora by using firewalls or other means. If this happens, then users might get the full-featured version of Eudora without either seeing ads or paying. Bad.

On the other hand, users may have hardware or software problems or other issues that keep them from fetching ads, or our ad servers might even be down for some reason. Users should not be punished for this.

We will distinguish between these two situations by asking a simple question--is the user sending or receiving mail? If the answer is yes, we will assume that the blocking of ads is something we need to address.

The way we address it is with an escalating series of Ad Failure Nags. These will continue for two weeks or until we receive ads. For every two days we do receive ads, we will back up the nag timer by one day. If the timer runs off the end, we will apologize to the user, revert to the Freeware version, and mark the user ad a Deadbeat User. Deadbeat Users will only be allowed to return to Adware if the ad server can be connected to at the time the user attempts to return to Adware.

Note that if we should ever decide to retire Eudora and wish to let people use it without ads, we can simply publish a permanent registration code.

Instrumentation & Auditing

One of the things we will need to know is that the ads we think are being displayed actually are being displayed, for as long and as often as we think they are being displayed. This will be crucially important in maintaining our credibility with advertisers. The following audit scheme is suggested:

•**Keep a rotating log of ad displays.** This log will be rolled over once per week. The log will record ad-related events--when an ad was displayed, when it was removed, and when it was clicked on--in addition to other events, like cumulative face time in Eudora, cumulative run time, etc..

•**At random, ask the user for permission to transmit the log.** At a frequency of once per hundred users per month, ask[9] for the user's permission to return the log to us. If the permission is given, the log will be formatted in ascii, placed in an outgoing message, and queued. The user will be given the opportunity to inspect and, if he desires, cancel the log collection.

• **For selected users, deliver a pastry.** In addition to the random send of the log, we will also have to at random ask particular users for their permission to audit transactions in detail with the server. This will allow us to correlate client and server behavior.

More detail on instrumentations is given in a separate document, "Instrumenting Eudora".

Ad Formats

The actual information we accept from advertisers will be relatively simple. For standard ads, will ask for a GIF file of not more than 15K, not more than144 pixels tall by 144 pixels wide, preferably using the web safety palette, plus a single URL to which people who click on the ad will be directed, plus whatever scheduling information the advertiser desires. In order to actually transmit the ad to Eudora, we may wrap the ad in HTML. This will allow us to include ad parameters as META tags in the HTML, specify the link address, etc.

Toolbar icons will be requested in GIF format as well, but will actually be delivered to the client in a composite format and blitted into standard icons. Placards for sponsors of the freeware version will be no more than 31 pixels tall, and on the order of 88 pixels wide, though the precise width can be tweaked at runtime.

Clicking On Ads

When the user clicks on an ad, we will normally take the user to our clickthrough counter and then to the link listed with the ad. The complication occurs if the user is offline at the time of

[9]Using a nag window, so as both to get the user's attention and yet not to stop other processes.

the click[10]. If the user is offline, several possibilities present themselves. Two of these will be present in the initial release:

- **Go online.** We would do this for a mailcheck, why not an ad click? Another weapon in our arsenal.
- **Mark it in the link history facility.** This is a window/menu we intend to keep, similar to the history lists of browsers. We will add a special mark to the link in the history window, so that the user knows they want to visit that site later.

These two might be considered for later:

- **Let the browser deal with it.** Some browsers may have sophisticated features of their own for dealing with offline-ness, and we shouldn't discount the idea that the user might wish to rely on them.
- **Schedule it for later transmission to the browser.** We can allow the user to tell us to send the link to his browser next time he's online.

The clickthrough counter will be our server. We will compose a URL out of our server name, some tracking information, and the ultimate destination URL, and then the server will redirect the user's browser to the destination URL.

[10]Just how we determine the user is offline is an interesting issue in and of itself.

WEB PAGES

This design calls for several web pages, briefly mentioned before. However, the general purpose of the pages will be defined, as well as the URL's we will use to access them.

THE QUERY PART

For many of our URL's, it will be helpful for the client to give the server information to help it direct the user to the proper place or prefill items on forms. The elements that might go in query parts are listed below.

The following items are considered personal, and great care should be taken to transmit them only when appropriate:

realname	The Real Name field from the user's Dominant personality.
regfirst	The first name under which the user registered last time (if any).
reglast	The last name under which the user registered last time (if any).
regcode	The user's current Eudora registration code (if any).
oldReg	The user's old-form reg code.
email	The email address from the user's Dominant personality.
profile	The profile information the user has entered.
destination	This is the URL which the user wishes to visit.
adid	This is the id of an ad on which the user clicked.

These items are not considered to be privacy-sensitive[11]:

platform	MacOS, Windows, Palm, Nintendo64, whatever.
product	Our code name for the product being registered. Eudora, PDQMail, etc..
version	The version number of the product being used to register. This should be of the form Major.Minor.Bugfix.Build.
distributorID	This will be a code for which sites may apply that will allow the site to receive a bounty for having provided users with this copy of Eudora.
action	What it is the user has requested to do; register, pay, lostcode, etc.
mode	Either "paid", "ad", or "free".
topic	Used for support items, this tells the server what particular kind of support is needed.

URL's

[11]By me, anyway. We may find users who feel differently.

All of our non-ad URL’s begin with:

```
http://jump.eudora.com/jump.cgi?action=whatever
```

The “action” value determines what function the user wishes to perform. We then append various other query parts, suitably %-escaped, according to the following chart:

	action	rm	uct	on	ID	ode	me	ail	rst	ast	ode	age	vel	ile	wl	did	topic	
Payment	pay	X	X	X	X	X	X	X	X	X	X	X						
Freeware Registration	register-free	X	X	X	X	X	X	X	X	X	X							
Adware Registration	register-ad	X	X	X	X	X	X	X	X	X	X							
Box Registrations	register-box	X	X	X	X	X	X	X	X	X	X							
Lost Code	lostcode	X	X	X	X	X	X	X	X	X	X	X						
Update	update	X	X	X	X	X							X					
Pro Update	proupdate	X	X	X	X	X							X					
Archived	archived	X	X	X	X	X												
Profile	profile	X	X	X	X	X	X	X						X				
Introduction	intro																	<
Support	n/a	X	X	X	X	X	X	X	X	X	X	X						
QuickTime Missing	support	X	X	X	X												no-qt	
Ad Failure	support	X	X	X	X												ad-fail	
Tutorial	support	X	X	X	X												tutor	<
FAQ	support	X	X	X	X												faq	<
Light Users	support	X	X	X	X												light	<
Search Support	support	X	X	X	X												search	<
Newsgroups	support	X	X	X	X												usenet	<

PAYMENT WEB PAGE

This web page should take the user’s credit card info, name, email address, and whatever other information we want to remember about our users. It will also ask them for a question and answer for use if they ever lose their payment code. It should spit back (that is, display and also email) their officially blessed registration name and their registration code.

FREEWARE REGISTRATION WEB PAGE

This web page should take the same info as the Payment web page, minus the credit card information. It should spit back (that is, display and also email) their officially blessed registration name and their registration code.

ADWARE REGISTRATION WEB PAGE

This web page should take the same info as the Payment web page, minus the credit card information. It should spit back (that is, display and also email) their officially blessed registration name and their registration code.

BOX REGISTRATION WEB PAGE

This web page exists to accept regisrations generated by our box or updater installers. It should simply accept the user's code, validate it, mail it back, and display a "thank you for registering page".

Lost Code Web Page

This web page needs to help users find their registration codes. When they register/pay, they'll be asked to provide their name, email address, and a question and answer. When they come to the lost code page, they'll be asked for name and address, and if that matches, they'll be asked their question. If all that goes well, their reg code will be mailed to them. If they can't receive mail, they'll have to call.[12]

Update Page

This web page should list the updates that are available to the user. Ideally, it would list only those updates the user does not already have, and clearly indicate which updates are free and which updates the user needs to pay for. This web page will be downloaded to the user's system from time to time and displayed "off-line" in Eudora, and so it should be kept small.

Archived Versions Web Page

This web page should list all versions of Eudora, so that users can download whatever they happen to need.

Profile Web Page

The purpose of this web page is to collect demographic information so that ads delivered to the user can cost advertisers more. What sort of pork chop we hang around the page's neck to entice users to actually cough up this data is not my department.[13]

At this page, the user will be asked a series of questions about his buying habits, sleeping habits, boxers or briefs, etc.. No information identifying the user is to be collected on this page! The information will be reduced to a cookie, mailed to Eudora and stored in his settings. The procedure for accepting a profile is the same as the procedure for accepting a registration code, detailed below.

Support Web Pages

We will need several web pages for resolving user problems. For these pages, we'll use the "topic" part of the query to direct users to situation-specific help as needed.

Registration

Registration Code

We will use a registration scheme with a self-validating registration code, so that databases do not need to be used to validate registrations. The algorithm for verification is intended to satisfy several conflicting constraints; it needs to be secure, yet easy to implement and not unduly burdensome for the user to type.

Inputs. The inputs to the registration code are as follows:

[12]Oh my god! They called Kenny!

[13]It is suggested that this enticement be delivered to the user in the form of an ad.

RegName	The name the users wishes to register under. We will imply but not require that this be the user's real name. The only thing this name will be used for is registration. Supplied by the user. When we actually collect this name from the user, we will ask for it in terms of first and last names, called RegFirstName and RegLastName, respectively. RegName is built by concatenating RegFirstName, a single space, and RegLastName. Each of the first and last names is limited to 20 significant characters; beyond that, characters will be ignored.
RegMonth	The date of the registration, expressed as the number of months since Jan 1, 1998. Supplied by us, 8 bits (20 years). All 1's reserved for "never expires".
Product	A numeric code indicating what product the registration is for. The user will choose the product, we will translate that choice into an 8-bit code.
RegLevel	This is 255*Product + RegMonth. Used in the update nag to avoid privacy concerns about other reg values.
Random Number	16 bits of random number, supplied by us.

Algorithm. The full reg code algorithm is beyond the scope of this document. In brief, we take the inputs above, checksum them, then mix the inputs (minus the RegName) and the checksum together in an obscure way and encode the result at a string of 16 numbers. The encoding and bit-mixing can be reversed and, together with the RegName, the checksum can be used to verify the validity of the code. Please see <http://echo.qualcomm.com/perforce/perfbrowse.perl?//depot/regcode/...> for details.

NOTE: We will store codes separately for Light, Sponsored and Paid modes. Acceptance of a code for one mode does not imply that code for the other modes should be destroyed.

Entering the Code

We will provide three ways for the registration code to be entered.

Manually. Users can type or paste values into the Enter Code dialog.

Windows Registry. At Eudora startup, we will look for the regcode in the Windows registry ((Software\Qualcomm\Eudora\Check, FName, LName, RCode). The values should be copied into the preferences if found and valid.

RegCode File. At Eudora startup, we will look for a file next to the application named "RegCode" (RegCode.dat if you insist on the 8.3 stuff). The values should be copied into the preferences if found and valid.

MIME Part. We will allow a special-case MIME part to be mailed to Eudora. The user receiving the part will automatically be asked to verify and enter the information. He can also execute the attachment again later. However, he cannot forward the attachment to anyone else using Eudora, because a special Content-Type attribute ("regcode") is required to activate the part, and Eudora can't send those.

Format

The format of the MIME part (and the RegCode file) is that of a text file containing RFC822-header-style fields. It has a registered MIME type of application/vnd.eudora.data. The fields are:

Eudora-File-Type	This is always the first field, and describes what sort of information the rest of the file contains. Its value will be either “RegFile” or “Profile”.

Other fields, which may or may not be present:

Eudora-First-Name	The first (given) name of the registrant.
Eudora-Last-Name	The last (family) name of the registrant. US-ASCII.
Eudora-Reg-Code	The registration code as produced by the registration system.
Profile	Profile information. This takes the form of a shortish (say, not more than 127 bytes) ascii string.
Eudora-Needs-Registration:	If this field contains “yes”, then the user should be nagged to register their copy of Eudora. This is used by installers that generate regcodes that we otherwise would not have in our database.
Mailed-To	This is the address the information was mailed to. If this field is present and does not match any of the user’s personalities or “me” nickname, the information should not be acted on.

Email Acceptance Policy

Regcodes mailed to the user should be validated before use. In order to be used, a code should meet the following tests:

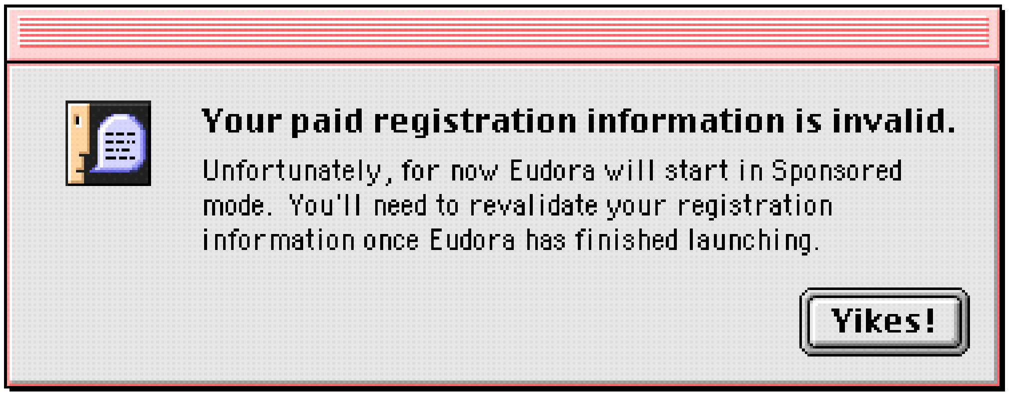
Validity	Invalid regcode should be ignored.
Directness	The mailed-to field of the regcode should contain an address for one of the user’s personalities or be in the user’s “me” nickname.
Applicability	A new reg code should not automatically override an existing valid reg code. The only exceptions to this policy are that a paid mode reg code should override a light or adware regcode, and a paid mode reg code that is the same as the user’s existing paid mode reg code can be used to turn off the “Eudora-Needs-Registration” situation.

Once the regcode has been determined to meet the above tests, the user should be asked to accept the code. The acceptance dialog is as follows:

Code Validation

The registration code is self-validating, since one part is a function of the other. However, there is another sense of “validation” to be considered, that is whether or not the registration code is “valid” for use with a particular version of Eudora. This is accomplished by comparing the ExpMonth in the registration code with a BuildMonth field we will put into the application (in a place that cannot be overriden by plug-ins, settings, etc.).

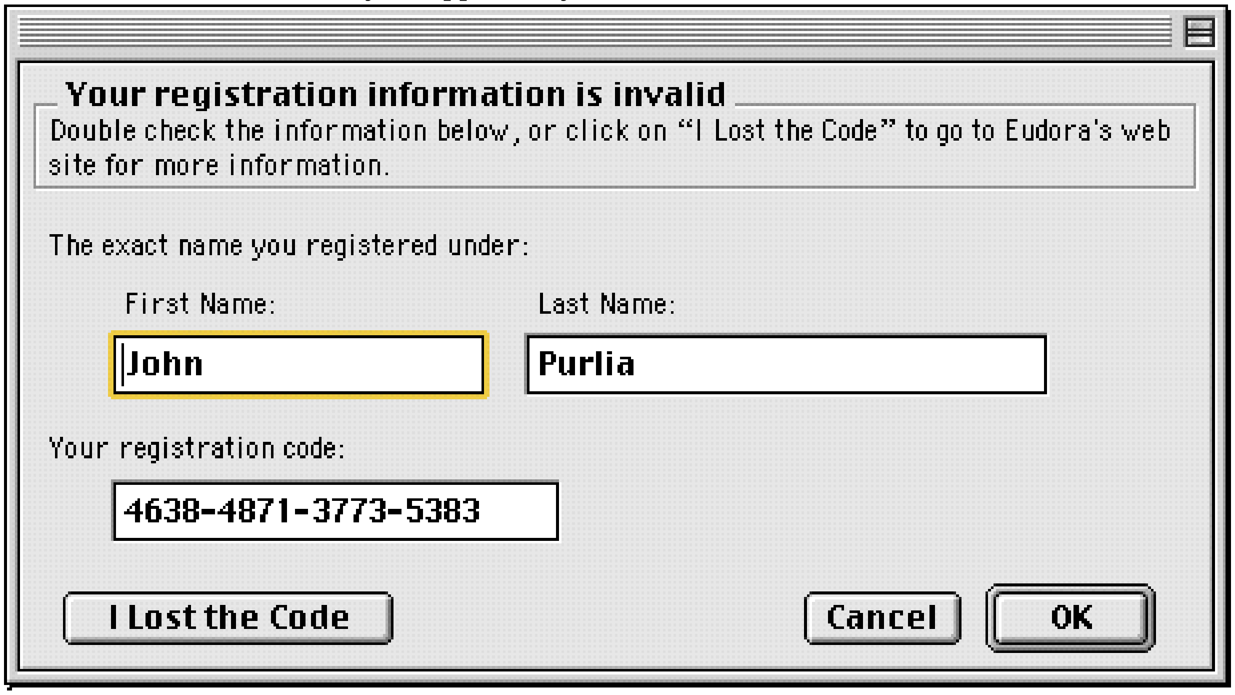
Eudora should check its registration code at startup for validity. If the registration code is invalid, the user should be considered unregistered. If the user is a paid mode user, this will involve a switch to Sponsored mode, about which the user should be warned:



Your paid registration information is invalid.

Unfortunately, for now Eudora will start in Sponsored mode. You'll need to revalidate your registration information once Eudora has finished launching.

This alert will be followed by an opportunity to reenter the code:



Your registration information is invalid

Double check the information below, or click on "I Lost the Code" to go to Eudora's web site for more information.

The exact name you registered under:

First Name: John

Last Name: Purlia

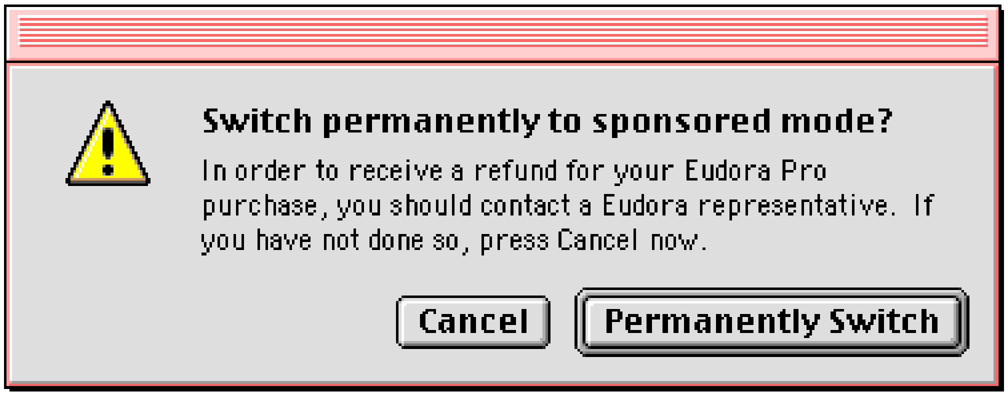
Your registration code: 4638-4871-3773-5383

Refunds

We hesitate to call this engineering, but we will put some smoke and mirrors in the program to make people believe that getting a refund is secure. In order to receive a refund, users will have to call the refund people, who will instruct them to:

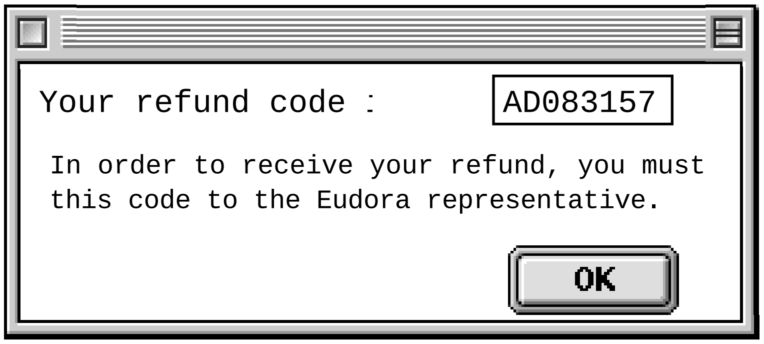
- Open the Payment & Registration window from the Help menu.
- Hold down the Shift key and click "Sponsored Mode (free, with ads)"
- Click "Permanently Switch". At this point, we will erase the paid registration code from Eudora's settings.

- Read the "refund code" back to the refund person. The code is a random number in hex.



Switch permanently to sponsored mode?

In order to receive a refund for your Eudora Pro purchase, you should contact a Eudora representative. If you have not done so, press Cancel now.



Your refund code : AD083157

In order to receive your refund, you must this code to the Eudora representative.

Report Bug & More Help! Items

We want to make it easy for users to report bugs and get tech support, provided they're entitled. The reg code is also the key to this entitlement. We will add two help items "More Help!" and "Report A Bug". "More Help!" will display some text that depends on the user's mode and version, and will give them links to various help resources. "Report A Bug" will send mail that includes the user's registration information in X-Eudora-Regcode and X-Eudora-Regname headers. We will add a special filter action to our own copies of Eudora to automatically validate the reg code so provided.

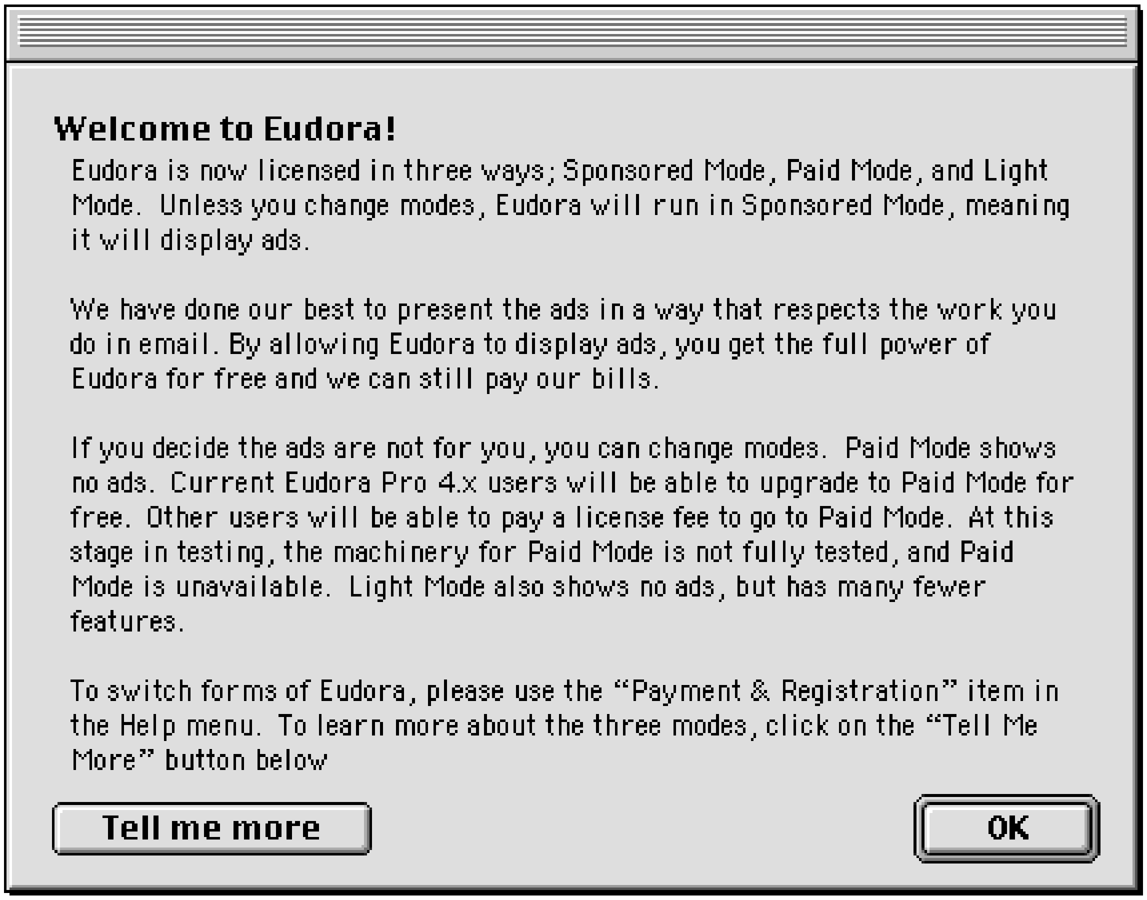
See Appendix F for the text for More Help!

Dialogs and Windows

What follows are samples of the various windows we will need. They have been introduced briefly before, we specify them in more detail here.

Intro Window

Upon first entering Eudora, users will see this:



Welcome to Eudora!

Eudora is now licensed in three ways; Sponsored Mode, Paid Mode, and Light Mode. Unless you change modes, Eudora will run in Sponsored Mode, meaning it will display ads.

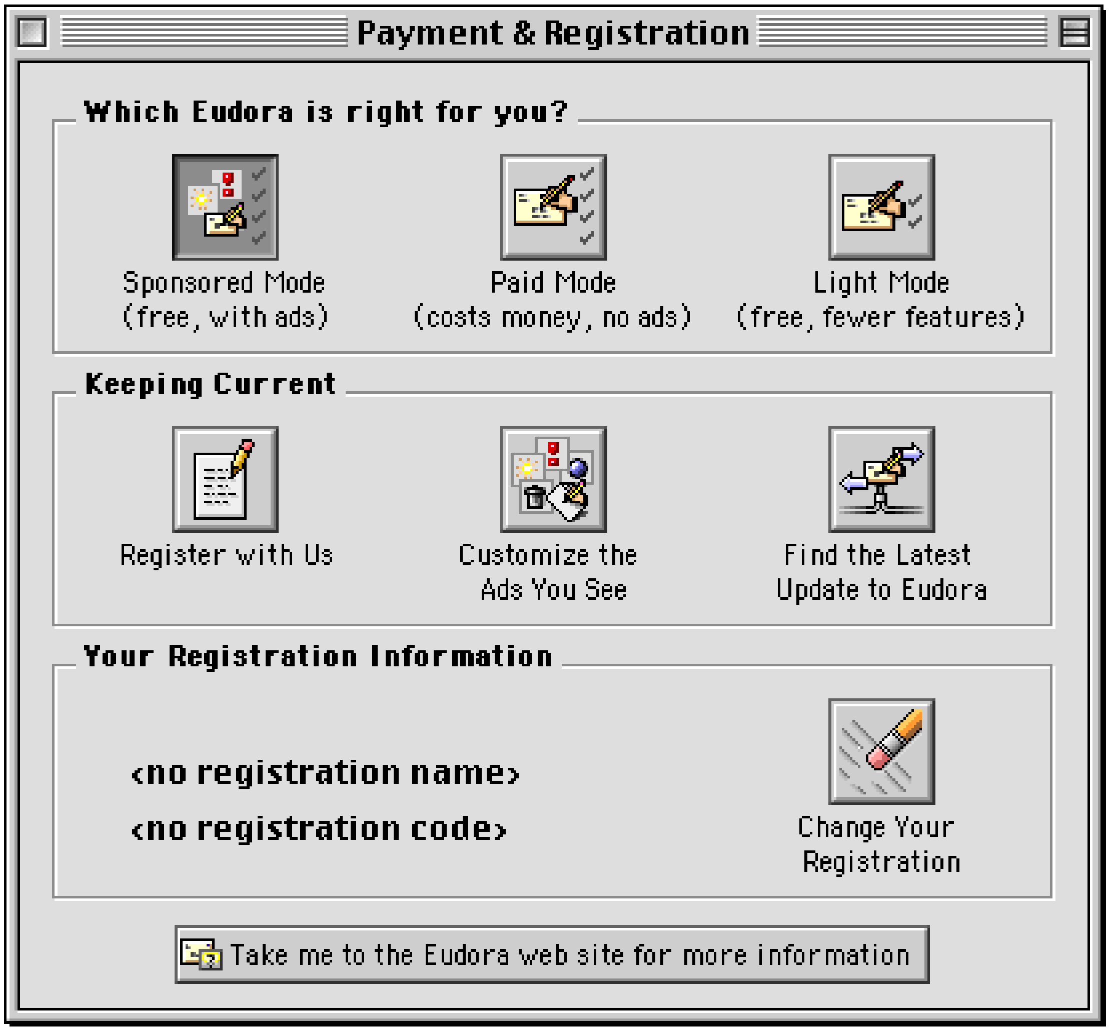
We have done our best to present the ads in a way that respects the work you do in email. By allowing Eudora to display ads, you get the full power of Eudora for free and we can still pay our bills.

If you decide the ads are not for you, you can change modes. Paid Mode shows no ads. Current Eudora Pro 4.x users will be able to upgrade to Paid Mode for free. Other users will be able to pay a license fee to go to Paid Mode. At this stage in testing, the machinery for Paid Mode is not fully tested, and Paid Mode is unavailable. Light Mode also shows no ads, but has many fewer features.

To switch forms of Eudora, please use the “Payment & Registration” item in the Help menu. To learn more about the three modes, click on the “Tell Me More” button below

Payment & Registration Window

This will be the primary place where the user interacts with the various Eudora modes. They're allowed to register, pay, switch modes, look for updates, etc. from this window.



Payment & Registration

Which Eudora is right for you?

Sponsored Mode (free, with ads)
Paid Mode (costs money, no ads)
Light Mode (free, fewer features)

Keeping Current

Register with Us
Customize the Ads You See
Find the Latest Update to Eudora

Your Registration Information

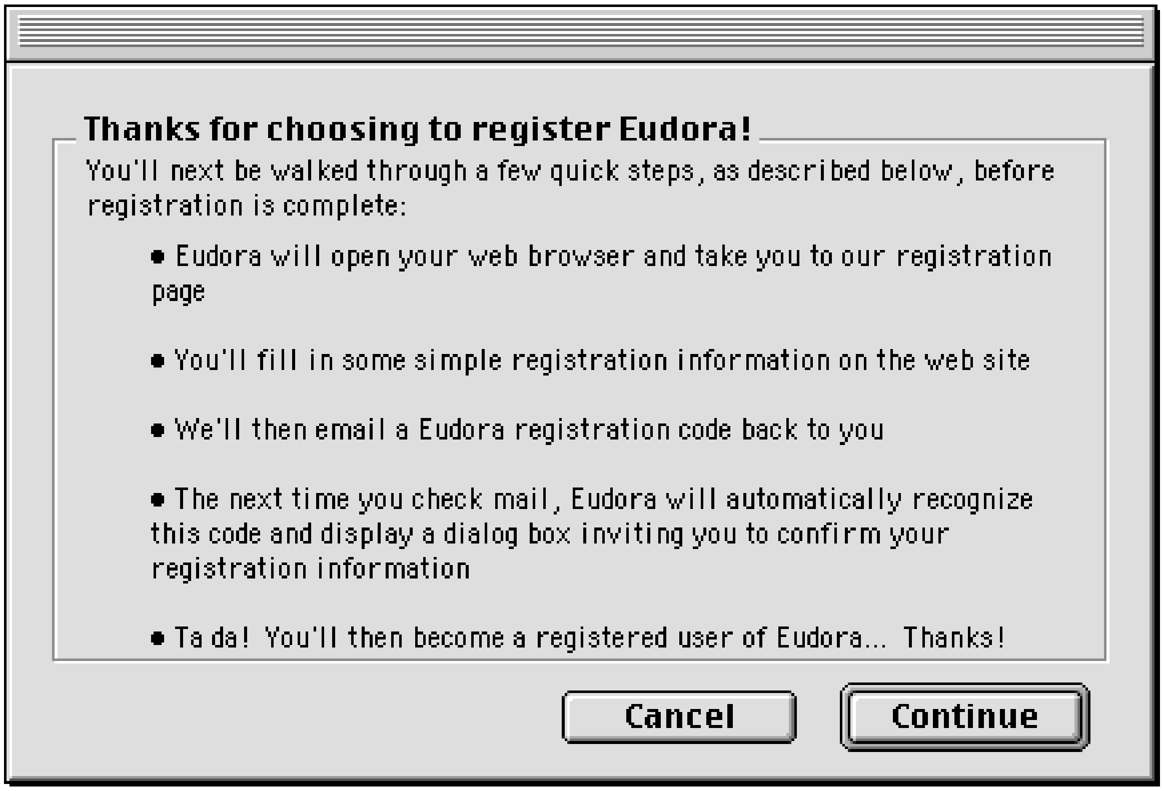
<no registration name>
<no registration code>

Change Your Registration

Take me to the Eudora web site for more information

PRE-REGISTRATION DIALOG

Because our registration process is a little cumbersome, we will explain it to the user in excruciating detail before they embark on the process.



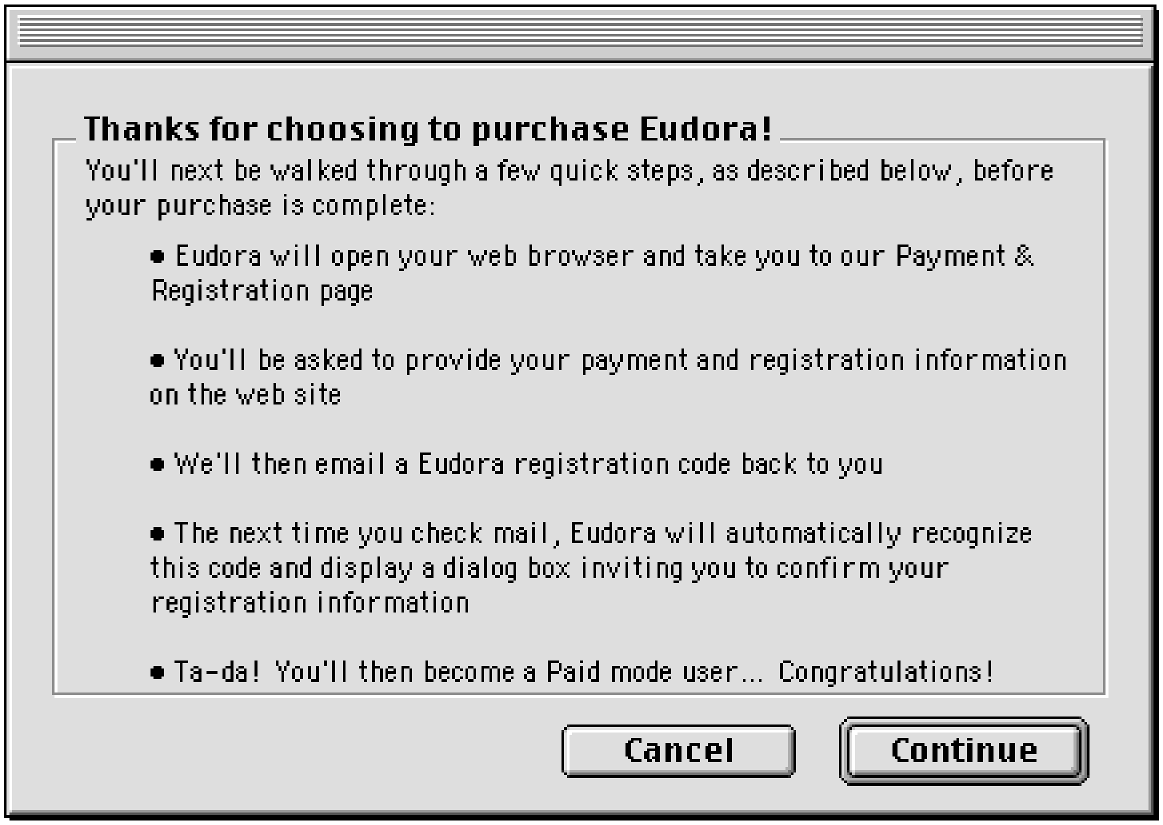
Thanks for choosing to register Eudora!

You'll next be walked through a few quick steps, as described below, before registration is complete:

- Eudora will open your web browser and take you to our registration page
- You'll fill in some simple registration information on the web site
- We'll then email a Eudora registration code back to you
- The next time you check mail, Eudora will automatically recognize this code and display a dialog box inviting you to confirm your registration information
- Ta da! You'll then become a registered user of Eudora... Thanks!

Pre-Payment Dialog

Again, we explain the payment process to the user:

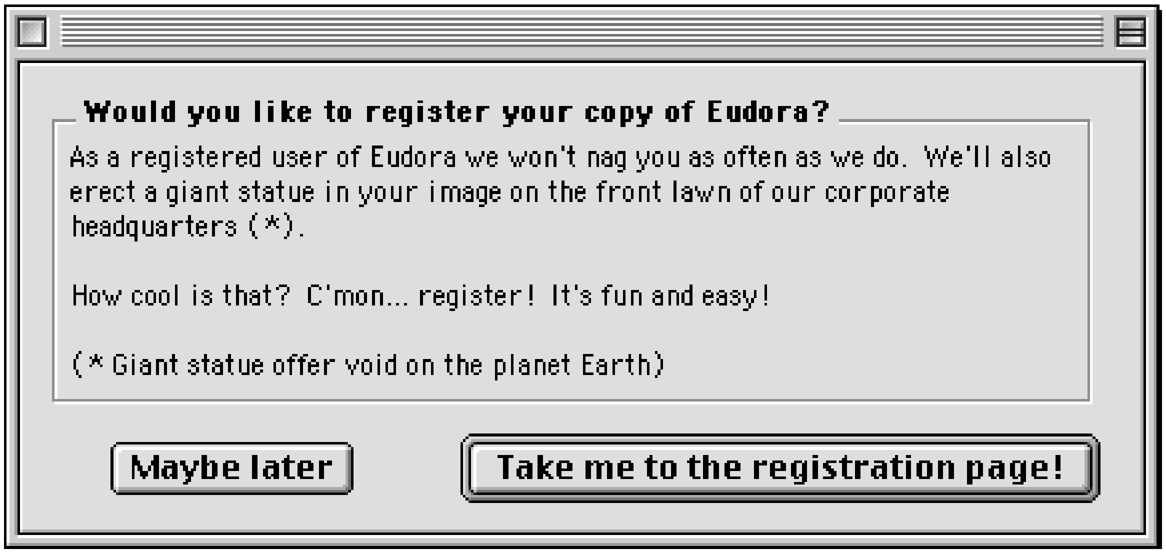


Thanks for choosing to purchase Eudora!

You'll next be walked through a few quick steps, as described below, before your purchase is complete:

- Eudora will open your web browser and take you to our Payment & Registration page
- You'll be asked to provide your payment and registration information on the web site
- We'll then email a Eudora registration code back to you
- The next time you check mail, Eudora will automatically recognize this code and display a dialog box inviting you to confirm your registration information
- Ta-da! You'll then become a Paid mode user... Congratulations!

Registration Nag



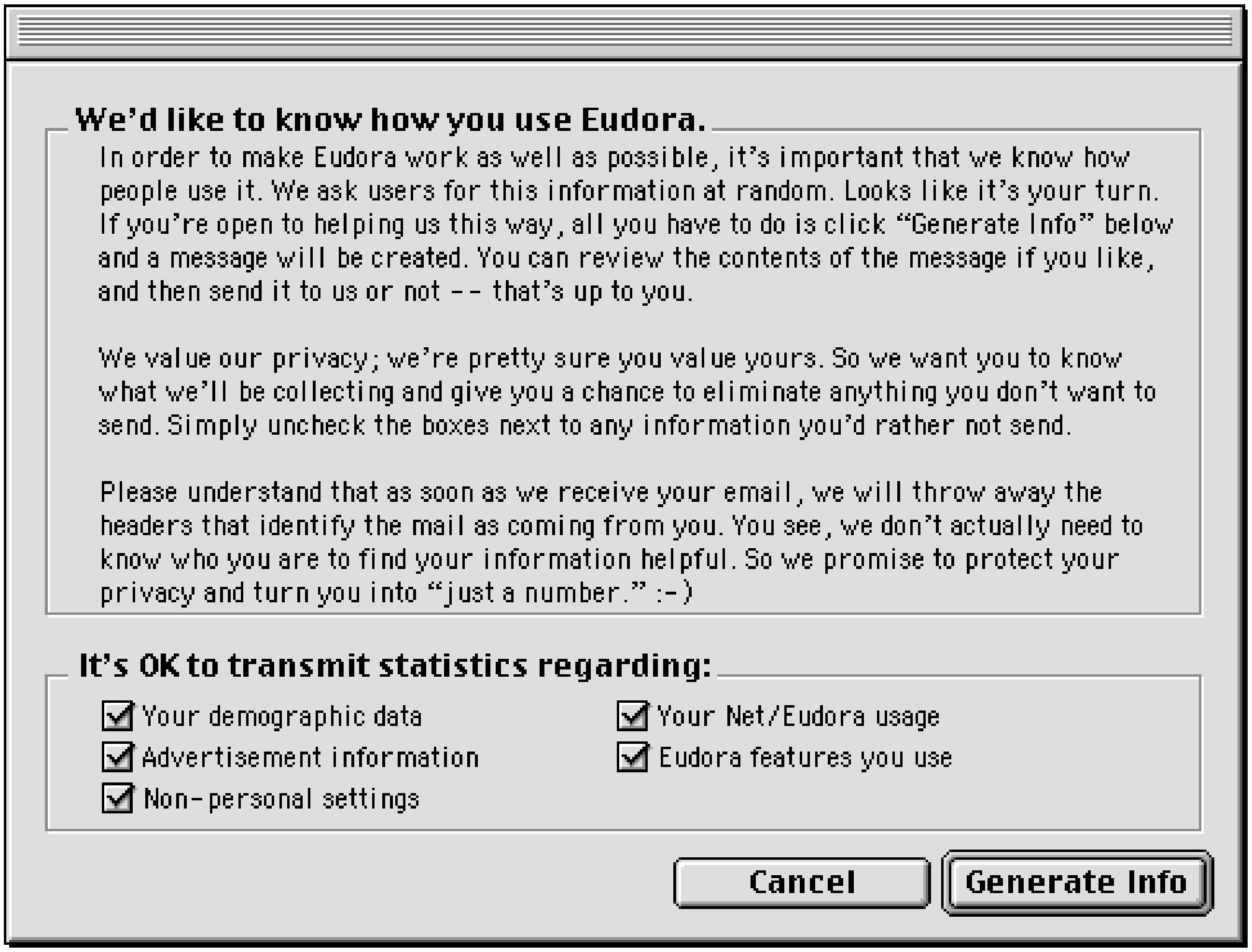
Would you like to register your copy of Eudora?

As a registered user of Eudora we won't nag you as often as we do. We'll also erect a giant statue in your image on the front lawn of our corporate headquarters (*).

How cool is that? C'mon... register! It's fun and easy!

(* Giant statue offer void on the planet Earth)

Send Audit Window



We'd like to know how you use Eudora.

In order to make Eudora work as well as possible, it's important that we know how people use it. We ask users for this information at random. Looks like it's your turn. If you're open to helping us this way, all you have to do is click "Generate Info" below and a message will be created. You can review the contents of the message if you like, and then send it to us or not -- that's up to you.

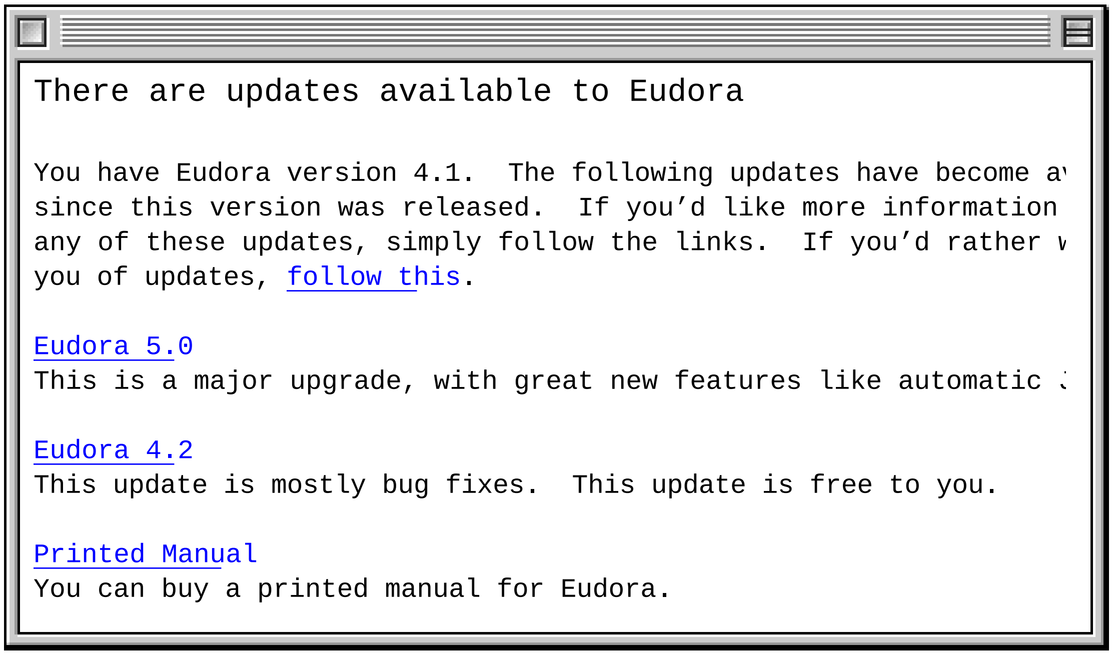
We value our privacy; we're pretty sure you value yours. So we want you to know what we'll be collecting and give you a chance to eliminate anything you don't want to send. Simply uncheck the boxes next to any information you'd rather not send.

Please understand that as soon as we receive your email, we will throw away the headers that identify the mail as coming from you. You see, we don't actually need to know who you are to find your information helpful. So we promise to protect your privacy and turn you into "just a number." :-)

It's OK to transmit statistics regarding:

- ✓ Your demographic data
- ✓ Advertisement information
- ✓ Non-personal settings
- ✓ Your Net/Eudora usage
- ✓ Eudora features you use

Update Nag



There are updates available to Eudora

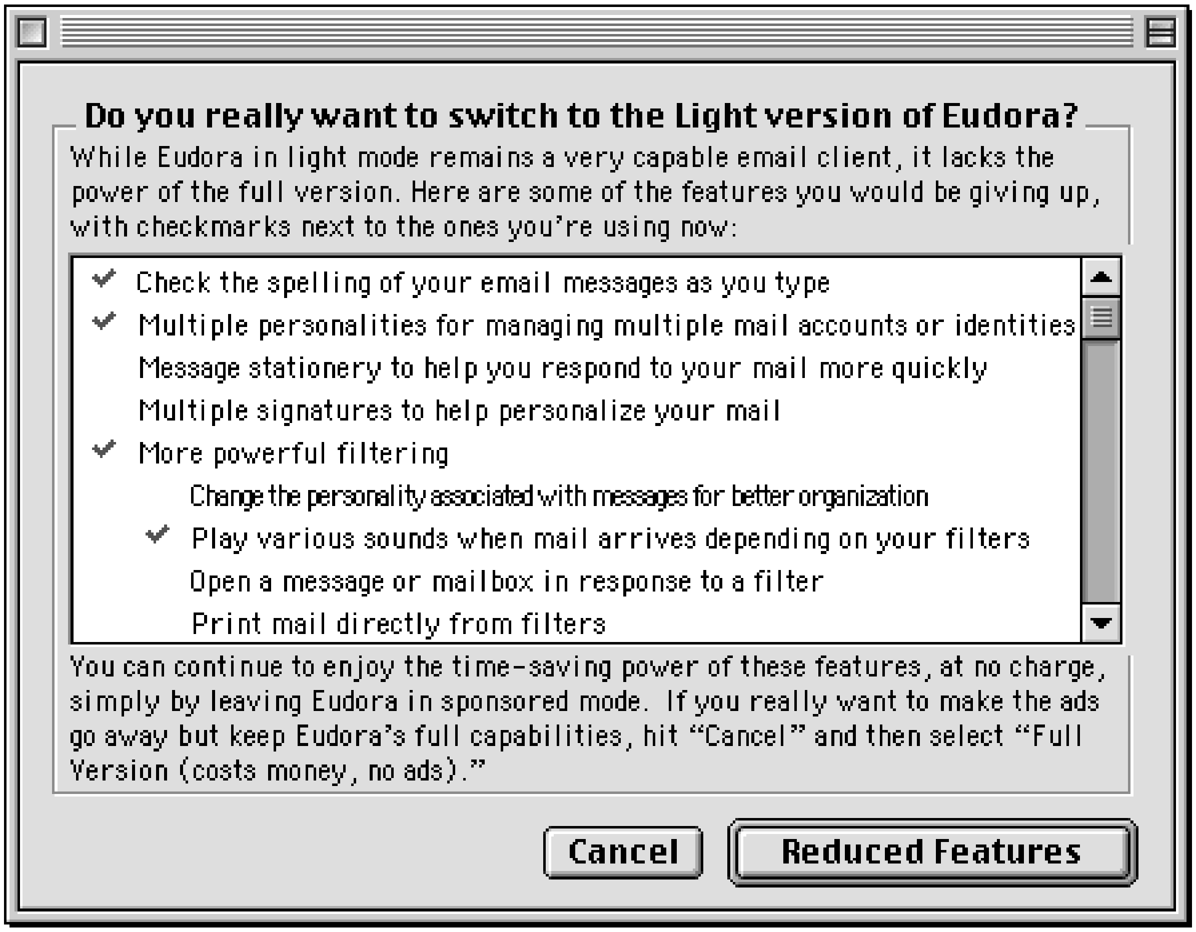
You have Eudora version 4.1. The following updates have become a
since this version was released. If you'd like more information
any of these updates, simply follow the links. If you'd rather w
you of updates, follow this.

Eudora 5.0
This is a major upgrade, with great new features like automatic

Eudora 4.2
This update is mostly bug fixes. This update is free to you.

Printed Manual
You can buy a printed manual for Eudora.

Freeware Downgrade Dialog



Do you really want to switch to the Light version of Eudora?

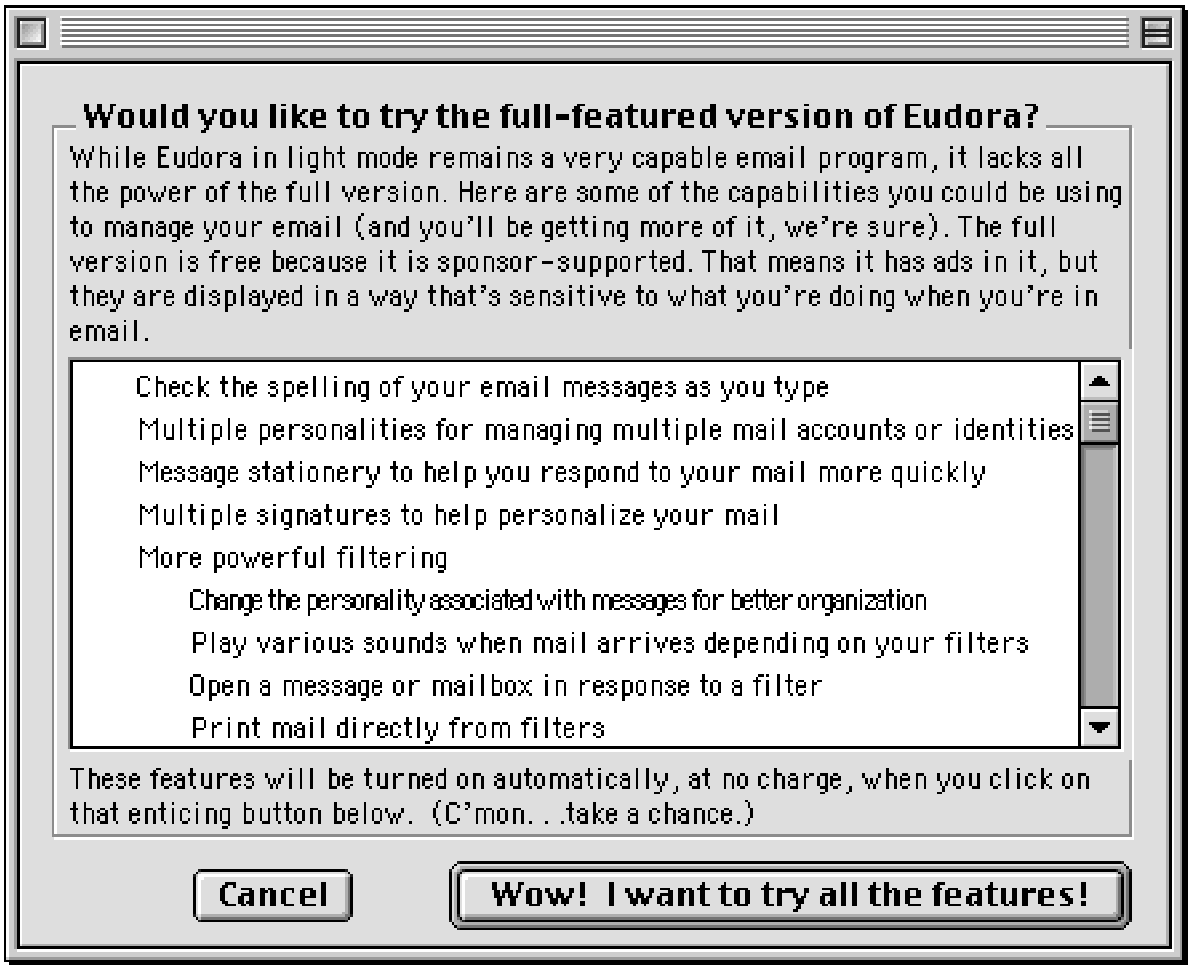
While Eudora in light mode remains a very capable email client, it lacks the power of the full version. Here are some of the features you would be giving up, with checkmarks next to the ones you're using now:

- ✓ Check the spelling of your email messages as you type
- ✓ Multiple personalities for managing multiple mail accounts or identities
- Message stationery to help you respond to your mail more quickly
- Multiple signatures to help personalize your mail
- ✓ More powerful filtering
  - Change the personality associated with messages for better organization
  - ✓ Play various sounds when mail arrives depending on your filters
  - Open a message or mailbox in response to a filter
  - Print mail directly from filters

You can continue to enjoy the time-saving power of these features, at no charge, simply by leaving Eudora in sponsored mode. If you really want to make the ads go away but keep Eudora's full capabilities, hit "Cancel" and then select "Full Version (costs money, no ads)."

FULL-FEATURE NAG

This one shows up for freeware users on a [30,30] schedule:



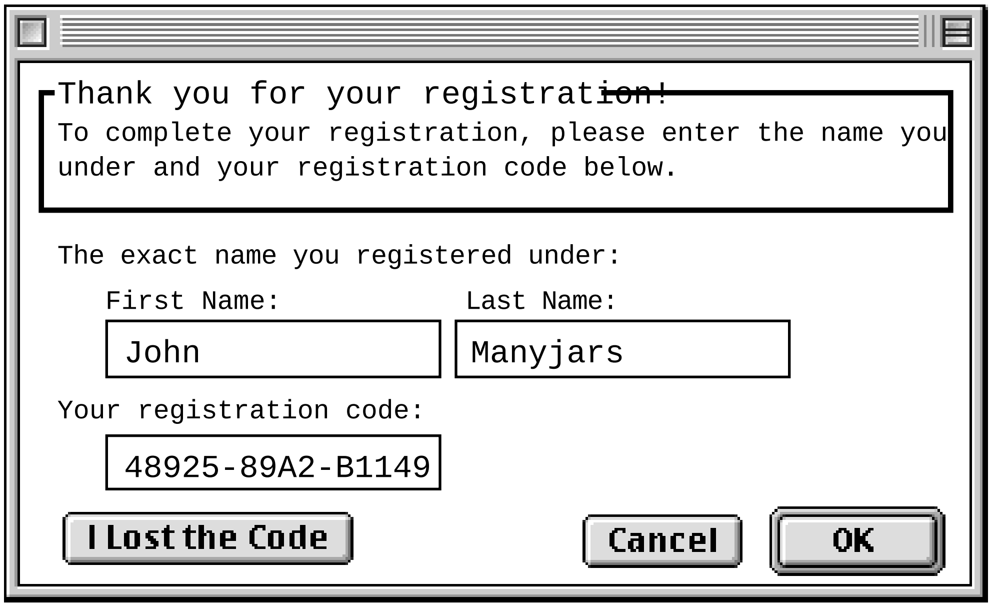
Would you like to try the full-featured version of Eudora?

While Eudora in light mode remains a very capable email program, it lacks all the power of the full version. Here are some of the capabilities you could be using to manage your email (and you'll be getting more of it, we're sure). The full version is free because it is sponsor-supported. That means it has ads in it, but they are displayed in a way that's sensitive to what you're doing when you're in email.

- Check the spelling of your email messages as you type
- Multiple personalities for managing multiple mail accounts or identities
- Message stationery to help you respond to your mail more quickly
- Multiple signatures to help personalize your mail
- More powerful filtering
  - Change the personality associated with messages for better organization
  - Play various sounds when mail arrives depending on your filters
  - Open a message or mailbox in response to a filter
  - Print mail directly from filters

These features will be turned on automatically, at no charge, when you click on that enticing button below. (C'mon. . .take a chance.)

CODE ENTRY DIALOG


Thank you for your registration!
To complete your registration, please enter the name you
under and your registration code below.

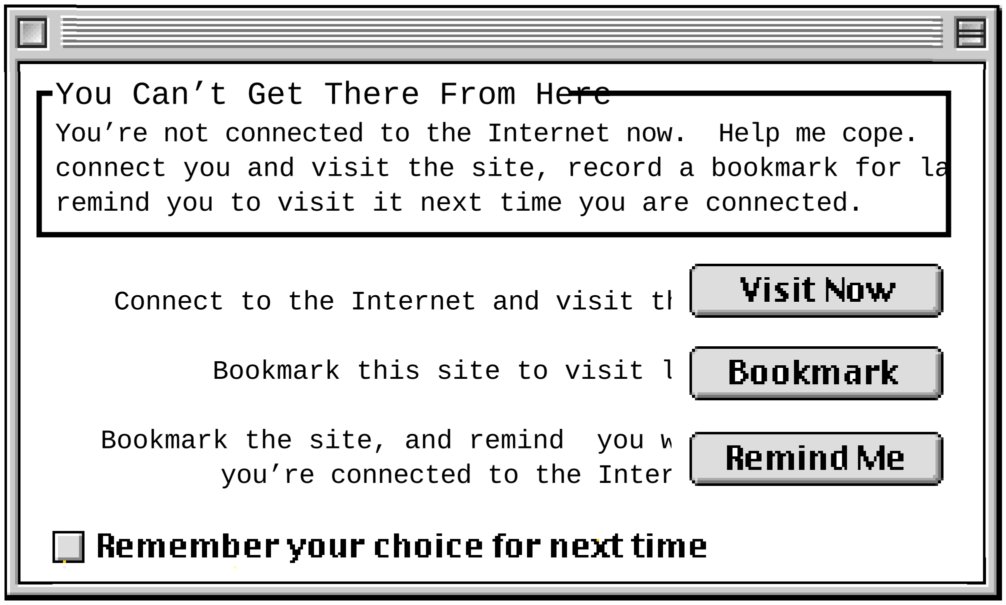
The exact name you registered under:

First Name: John
Last Name: Manyjars

Your registration code: 48925-89A2-B1149

Offline Ad/Link Dialog

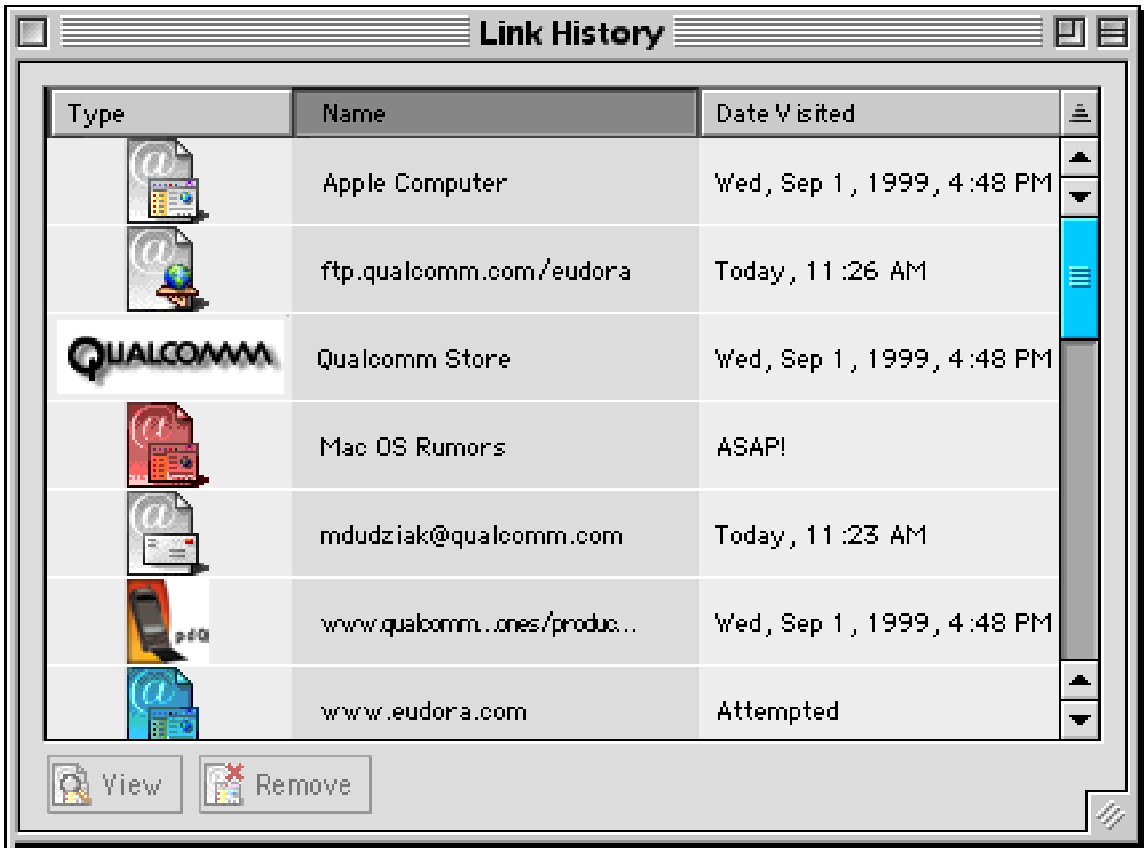
We will bring up this dialog when the user is known to be offline and clicks a link or ad.


You Can’t Get There From Here
You’re not connected to the Internet now. Help me cope.
connect you and visit the site, record a bookmark for la
remind you to visit it next time you are connected.

Connect to the Internet and visit tl
Bookmark this site to visit l
Bookmark the site, and remind you w
you’re connected to the Inter

Link History Window

This window will list links the user has visited and ads the user has been displayed, along with some status information on each. Selecting an ad will display the ad, double-clicking it will visit the site.

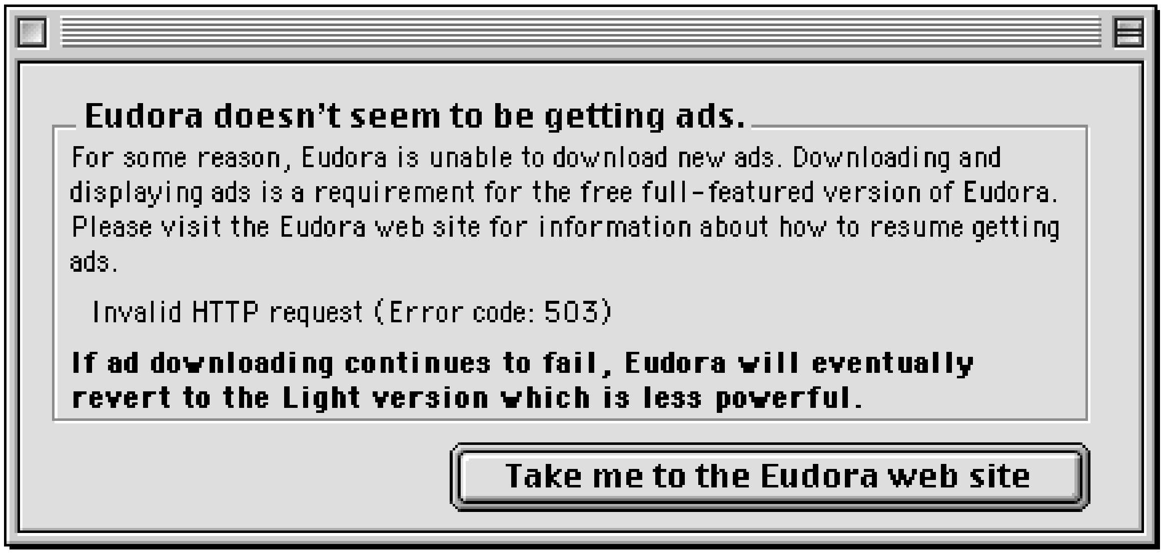


Link History

Type	Name	Date Visited
	Apple Computer	Wed, Sep 1, 1999, 4:48 PM
	ftp.qualcomm.com/eudora	Today, 11:26 AM
	Qualcomm Store	Wed, Sep 1, 1999, 4:48 PM
	Mac OS Rumors	ASAP!
	mdudziak@qualcomm.com	Today, 11:23 AM
	www.qualcomm...ones/produc...	Wed, Sep 1, 1999, 4:48 PM
	www.eudora.com	Attempted

Ad Failure Nag

If the user doesn't get ads for a while, they are nagged thus:



Eudora doesn't seem to be getting ads.

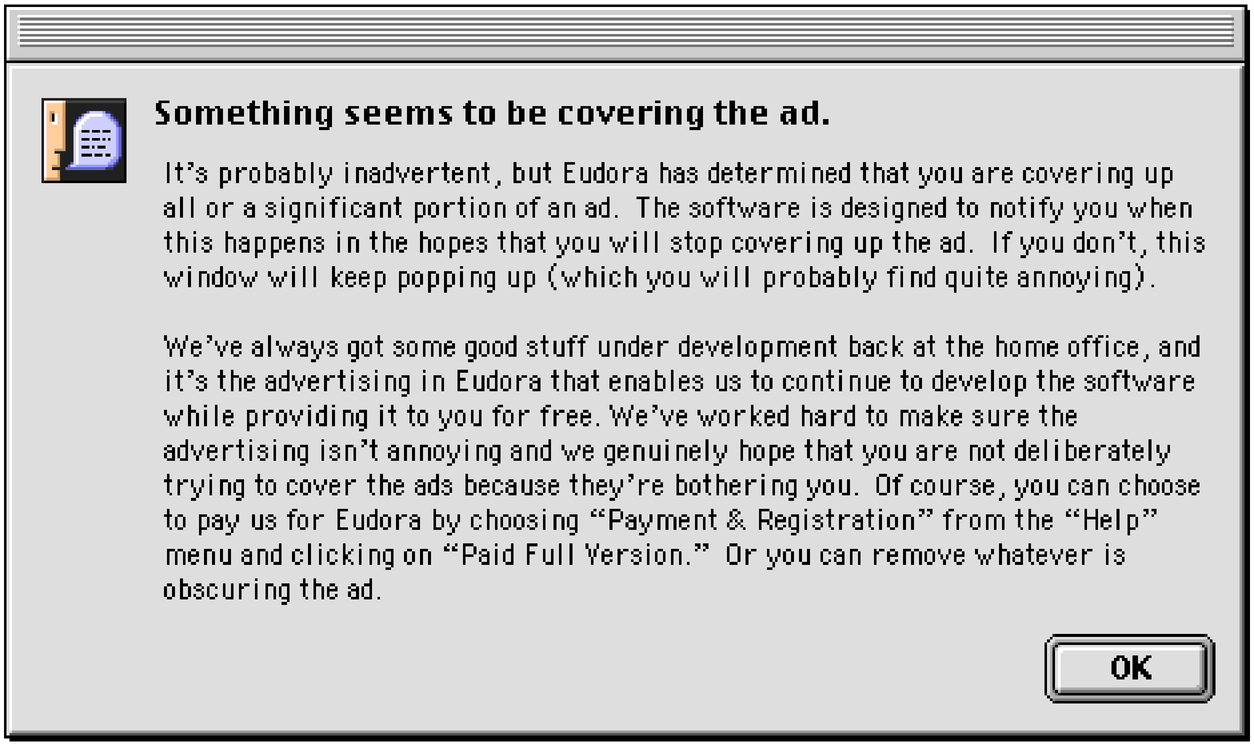
For some reason, Eudora is unable to download new ads. Downloading and displaying ads is a requirement for the free full-featured version of Eudora. Please visit the Eudora web site for information about how to resume getting ads.

Invalid HTTP request (Error code: 503)

If ad downloading continues to fail, Eudora will eventually revert to the Light version which is less powerful.

OBSCURED AD NAG

If the user covers the ad, we say:



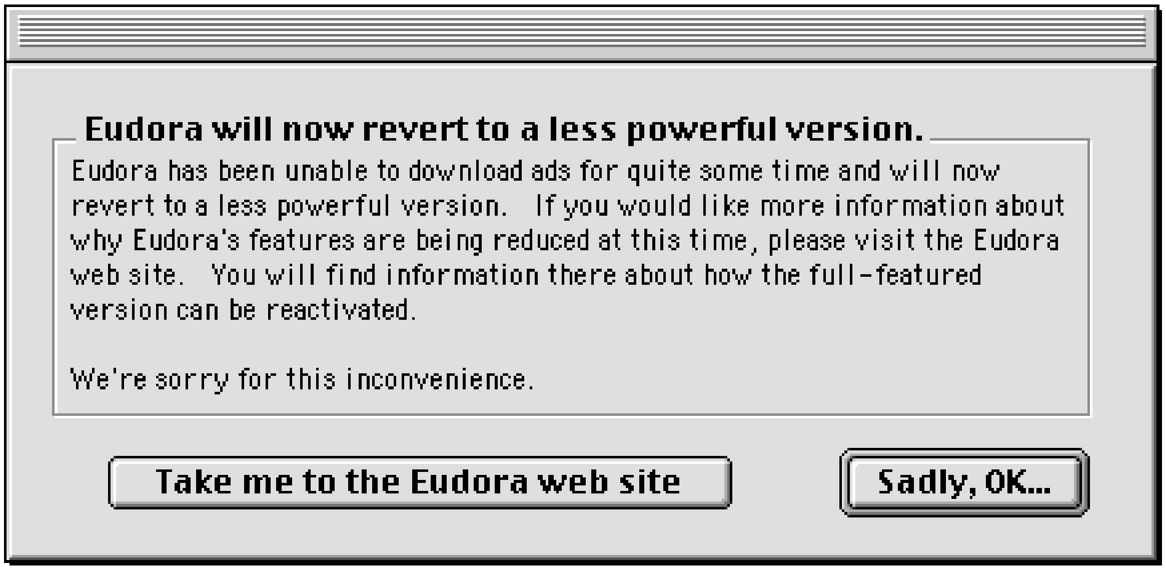
Something seems to be covering the ad.

It's probably inadvertent, but Eudora has determined that you are covering up all or a significant portion of an ad. The software is designed to notify you when this happens in the hopes that you will stop covering up the ad. If you don't, this window will keep popping up (which you will probably find quite annoying).

We've always got some good stuff under development back at the home office, and it's the advertising in Eudora that enables us to continue to develop the software while providing it to you for free. We've worked hard to make sure the advertising isn't annoying and we genuinely hope that you are not deliberately trying to cover the ads because they're bothering you. Of course, you can choose to pay us for Eudora by choosing "Payment & Registration" from the "Help" menu and clicking on "Paid Full Version." Or you can remove whatever is obscuring the ad.

DEADBEAT ALERT

Users who don't see the error of their ways from ad failure nags eventually wind up here.



Eudora will now revert to a less powerful version.

Eudora has been unable to download ads for quite some time and will now revert to a less powerful version. If you would like more information about why Eudora's features are being reduced at this time, please visit the Eudora web site. You will find information there about how the full-featured version can be reactivated.

We're sorry for this inconvenience.

Appendix A - Open Issues

We certainly have open issues, but they are not listed here.

Appendix B - Preliminary Tasks and Times

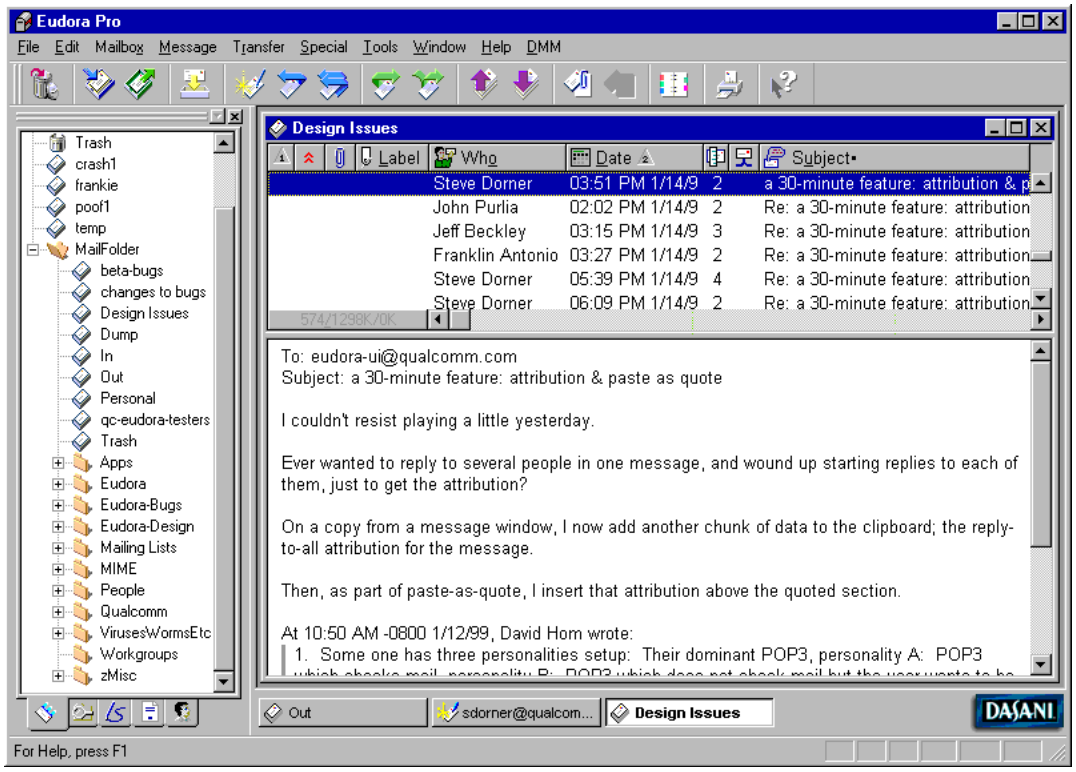
Tasks and times are on vacation.

APPENDIX C - ALTERNATE AD PLACEMENT

Please see separate document.

Appendix D - Freeware Sponsor Placement

In freeware mode, we plan to have a single sponsor at any given time, and to keep the sponsorship notice very low-key.



Label	Who	Date		Subject
	Steve Dorner	03:51 PM 1/14/9	2	a 30-minute feature: attribution & p
	John Purlia	02:02 PM 1/14/9	2	Re: a 30-minute feature: attribution
	Jeff Beckley	03:15 PM 1/14/9	3	Re: a 30-minute feature: attribution
	Franklin Antonio	03:27 PM 1/14/9	2	Re: a 30-minute feature: attribution
	Steve Dorner	05:39 PM 1/14/9	4	Re: a 30-minute feature: attribution
	Steve Dorner	06:09 PM 1/14/9	2	Re: a 30-minute feature: attribution

To: eudora-ui@qualcomm.com
Subject: a 30-minute feature: attribution & paste as quote

I couldn't resist playing a little yesterday.

Ever wanted to reply to several people in one message, and wound up starting replies to each of them, just to get the attribution?

On a copy from a message window, I now add another chunk of data to the clipboard; the reply-to-all attribution for the message.

Then, as part of paste-as-quote, I insert that attribution above the quoted section.

At 10:50 AM -0800 1/12/99, David Horn wrote:
> 1. Some one has three personalities setup: Their dominant POP3, personality A: POP3

Appendix E - Ad Submission Guidelines

Standard Ads

We will require the following of standard advertisements submitted by our advertisers:

- **No larger than 144x144 pixels.** Ads smaller than this will be centered in a 144x144 window and surrounded by the standard frame color.
- **GIF or JPEG.** We will convert GIF to PNG, but this is transparent. We will *not* accept PNG ads directly, because of the gamma bugs in PhotoShop.
- **No larger than 12K.** This will reduce our bandwidth cost as well as the goodwill cost of user bandwidth.
- **No animation.** This is a cornerstone of our "unobtrusive" message to users.
- **A single URL of not more than 900 characters.** There are suspected limits of 1K on URL size. Limiting the customer's URL to 900 characters will allow us to annotate the URL and still stay within 1K.
- **A user-friendly title string of not more than 31 characters.** This string will be put in the link history window, and should be something users will relate to.

In addition, we make the following recommendation:

- **Use Safety/Web Palette.** This is what will display best for users with 256-color systems. We do not take responsibility for the appearance of ads on such systems unless the Safety/Web Palette is used.

Toolbar Buttons

Toolbar buttons have the same requirement as standard ads, except:

- **Both 16x16 and 32x32 sizes required.** These are the sizes the client supports, we need them both.
- **GIF only.** We will not render JPEG's in the toolbar.

Co-Brand Spot

The co-brand spot has the same requirement as standard ads, except:

- **No larger than 95 wide by 31 high, pixels.**
- **GIF only.**

APPENDIX F - TECH SUPPORT TEXT

Under “Help:Questions or Problems?” we will have:

> Many problems can be solved by updating to the latest version of Eudora. *(to Update Nag)*
>
> Confused about some of the powerful features in Eudora? Click here to get step-by-step instructions on how to get the most out of your Eudora. *(support?tutor)*
>
> What are other people asking us? Click here to see our Frequently Asked Questions. *(support?faq)*
>
> Still have a question? Click here search our knowledge base for answers to specific questions you may have. *(support?search)*
>
> There's even a Eudora Usenet News group: comp.mail.eudora.platform. *(news:comp.mail.eudora.platform)*

In Paid or Ad mode, we will go on to say:

> If your question isn't answered in the above locations, you may send mail to eudora-support@qualcomm.com.
>
> Please include your registration number. If you use the link above, your registration number will be included for you.
>
> You may also call and talk to a technician at 1-858-658-1292. Our hours are from 8:00 - 5:00 Pacific time Monday through Friday. Please have your registration number ready.

In Light Mode, the user will get:

> If your question isn’t answered, you might consider switching to Paid or Sponsored mode, where person-to-person technical support is available. For more information, click here. *(support?light)*