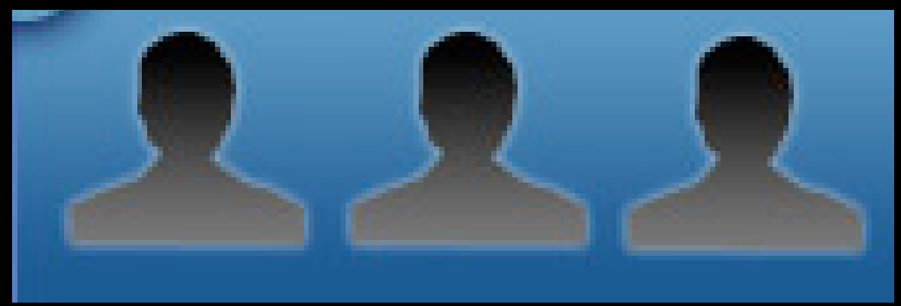
Client Management Fundamentals

MacWorld SF 2007
Session IT851

Who?

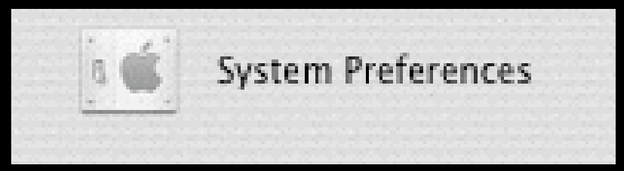
- Administrative staff?
- Faculty?
- V.I.P.’s?



What?

- Applications?

- System settings?



- Look and feel?

Why?

- Efficiency?
- Lack of staff?
- Maximizing uptime?

Three types of managed environment

- Fully Managed (Kiosk)
  - Users have little or no control of their environment
  - Machine is always in a known state on reboot
- Mostly managed (Desktop/Laptop)
  - Users do not run as administrators
  - Users can control their environment
  - Machine is in a fairly known state
- Mostly unmanaged (Student laptop,
  - Users have full control of the box
  - Box is always in an unknown state

Options for great flexibility

(Without Workgroup Manager)

/etc/authorization

- Allows equivalent of “power-user”
- Rights can be granted/denied based on group
- Fine grained control of some administrative rights

Example right

```
<key>com.apple.activitymonitor.kill</key>
<dict>
        <key>class</key>
        <string>user</string>
        <key>comment</key>
        <string>Used by Activity Monitor to authorize
killing processes not owned by the user</string>
        <key>group</key>
        <string>admin</string>
        <key>shared</key>
        <false/>
        <key>timeout</key>
        <integer>0</integer>
</dict>
```

Example right

```
<key>com.apple.activitymonitor.kill</key>
<dict>
        <key>class</key>
        <string>user</string>
        <key>comment</key>
        <string>Used by Activity Monitor to authorize
killing processes not owned by the user</string>
        <key>group</key>
        <string>admin</string>
        <key>shared</key>
        <false/>
        <key>timeout</key>
        <integer>0</integer>
</dict>
```

Example right

```
<key>com.apple.activitymonitor.kill</key>
<dict>
        <key>class</key>
        <string>user</string>
        <key>comment</key>
        <string>Used by Activity Monitor to authorize
killing processes not owned by the user</string>
        <key>group</key>
        <string>admin</string>
        <key>shared</key>
        <false/>
        <key>timeout</key>
        <integer>0</integer>
</dict>
```

Example right

```
<key>com.apple.activitymonitor.kill</key>
<dict>
        <key>class</key>
        <string>user</string>
        <key>comment</key>
        <string>Used by Activity Monitor to authorize
killing processes not owned by the user</string>
        <key>group</key>
        <string>admin</string>
        <key>shared</key>
        <false/>
        <key>timeout</key>
        <integer>0</integer>
</dict>
```

Another example

```
<key>system.preferences</key>
 <dict>
        <key>allow-root</key>
        <true/>
        <key>class</key>
        <string>user</string>
        <key>comment</key>
        <string>This right is checked by the Admin
framework when making changes to the system preferences.</string>
        <key>group</key>
        <string>admin</string>
        <key>shared</key>
        <false/>
    </dict>
```

Another example

```
<key>system.preferences</key>
<dict>
        <key>allow-root</key>
        <true/>
        <key>class</key>
        <string>user</string>
        <key>comment</key>
        <string>This right is checked by the Admin
framework when making changes to the system preferences.</string>
        <key>group</key>
        <string>admin</string>
        <key>shared</key>
        <false/>
    </dict>
```

File authorization

authopen provides authorization-based file opening services. In its simplest form, authopen verifies that it is allowed to open filename (using an appropriate sys.openfile.* authorization right) and then writes the file to stdout. If -w is specified, authopen will read from stdin and write to the file.

authopen is designed to be used both from the command line and programmatically. The -stdoutpipe flag allows a parent process to receive an open file descriptor pointing to the file in question.

Before opening filename, authopen will make an authorization request for a right of the form:

sys.openfile.[readonly|readwrite|readwritecreate]./fully/qualified/path.
`.readonly' rights only allow for read-only file descriptors.
`.readwrite' rights allow for read/write file descriptors.
`.readwritecreate' rights allow for read/write descriptors and the creation of new files.

Let’s dig in!

Sudo!

Granularity in sudo: aliases

```
User_Alias      FULLTIMERS = millert, mikef, dowdy
User_Alias      PARTTIMERS = bostley, jwfox, crawl
User_Alias      WEBMASTERS = will, wendy, wim
Runas_Alias     OP = root, operator
Runas_Alias     DB = oracle, sybase
Host_Alias      SPARC = bigtime, eclipse, moet, anchor :\
                SGI = grolsch, dandelion, black :\
                ALPHA = widget, thalamus, foobar :\
                HPPA = boa, nag, python
Host_Alias      CUNETS = 128.138.0.0/255.255.0.0
Host_Alias      CSNETS = 128.138.243.0, 128.138.204.0/24, 128.138.242.0
Host_Alias      SERVERS = master, mail, www, ns
Host_Alias      CDROM = orion, perseus, hercules
Cmnd_Alias      DUMPS = /usr/bin/mt, /usr/sbin/dump, /usr/sbin/rdump,\
                        /usr/sbin/restore, /usr/sbin/rrestore
Cmnd_Alias      KILL = /usr/bin/kill
Cmnd_Alias      PRINTING = /usr/sbin/lpc, /usr/bin/lprm
Cmnd_Alias      SHUTDOWN = /usr/sbin/shutdown
Cmnd_Alias      HALT = /usr/sbin/halt
Cmnd_Alias      REBOOT = /usr/sbin/reboot
Cmnd_Alias      SHELLS = /usr/bin/sh, /usr/bin/csh, /usr/bin/ksh, \
                         /usr/local/bin/tcsh, /usr/bin/rsh, \
                         /usr/local/bin/zsh
Cmnd_Alias      SU = /usr/bin/su
```

Granularity in sudo: defaults

```
Defaults                syslog=auth
Defaults>root           !set_logname
Defaults:FULLTIMERS     !lecture
Defaults:millert        !authenticate
Defaults@SERVERS        log_year, logfile=/var/log/sudo.log
```

Examples:

```
operator        ALL = DUMPS, KILL, SHUTDOWN, HALT, REBOOT, PRINTING,\
                 sudoedit /etc/printcap, /usr/oper/bin/

joe             ALL = /usr/bin/su operator



pete            HPPA = /usr/bin/passwd [A-z]*, !/usr/bin/passwd root

aaron  shanty = NOEXEC: /usr/bin/more, /usr/bin/vi
```

Stock sudo on Macintosh OS X

```
root    ALL=(ALL) ALL
%admin  ALL=(ALL) ALL
```

Refining sudo permissions

```
root    ALL=(ALL) ALL
singleuser  ALL=(ALL) ALL
```

Managed Preferences

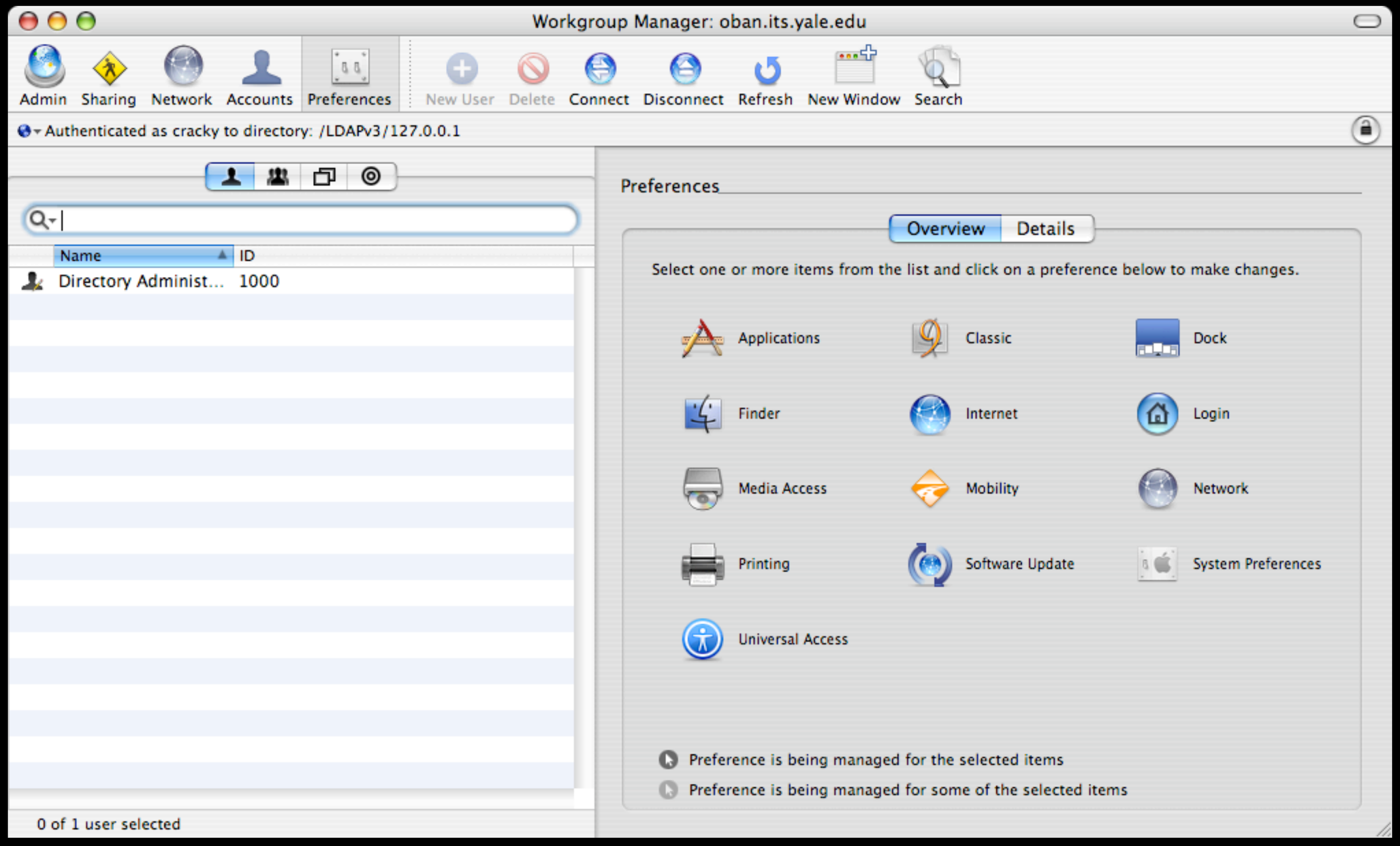
Three types of management

- Users
  - Applications, Dock, Login
- Groups
  - Same as Users
- Computers
  - Ability to manage Energy Saver is added
  - Login scripts are added as well
  - Ability to restrict machine to specific set of groups

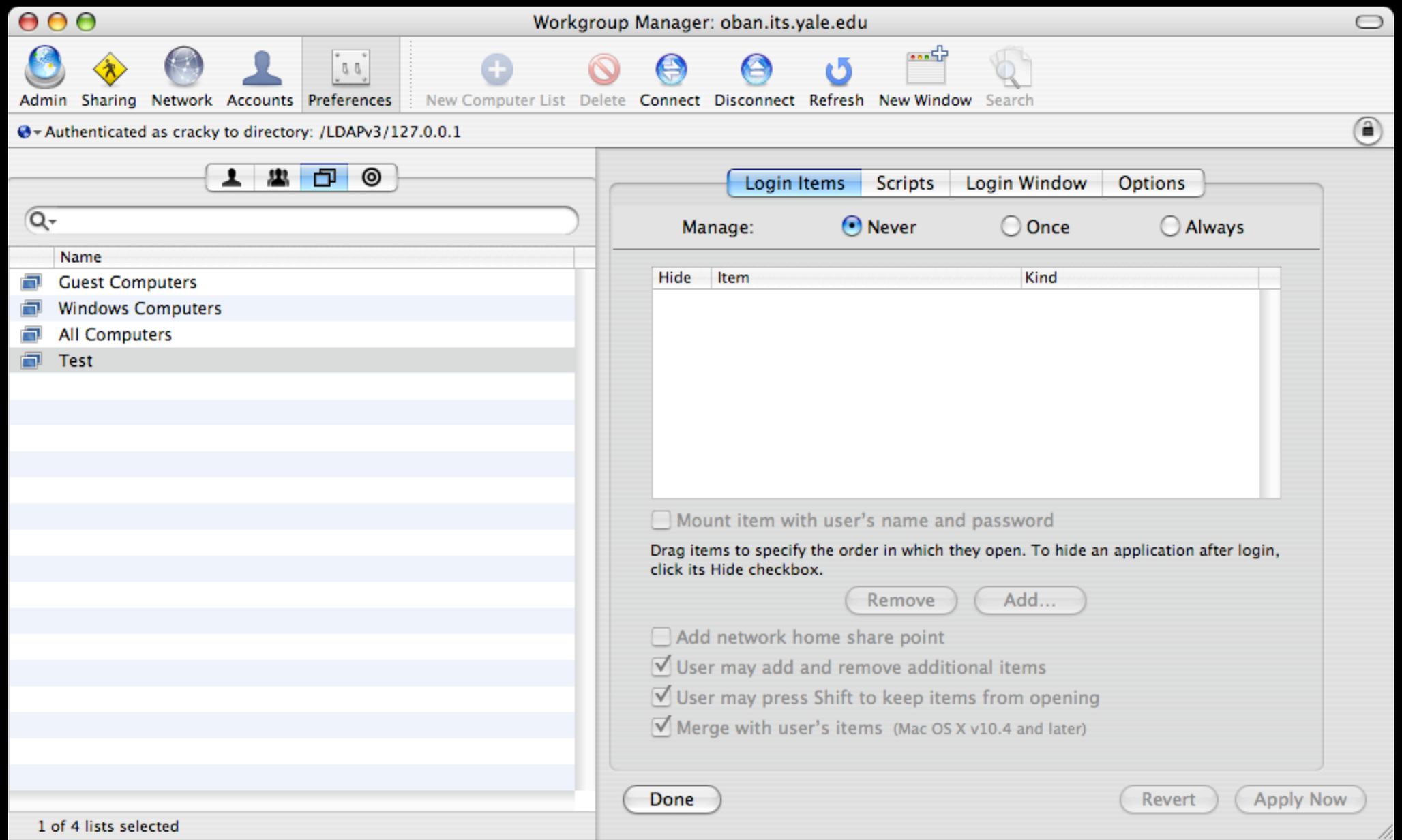
How?

- Open Directory Server for authentication and management
- “Triangle” method
- Local management

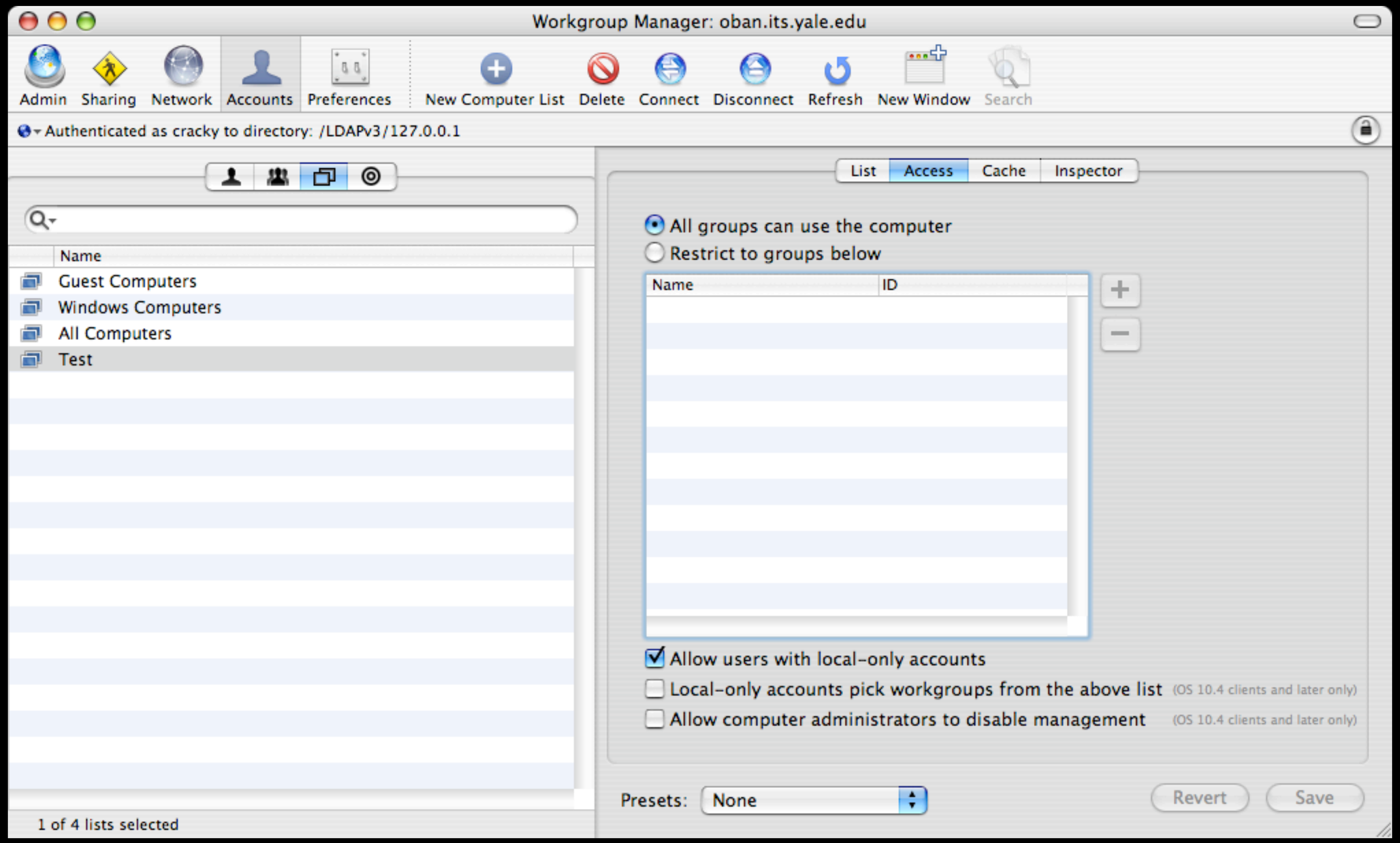
Open Directory Server


Workgroup Manager: oban.its.yale.edu
Admin
Sharing
Network
Accounts
Preferences
New User
Delete
Connect
Disconnect
Refresh
New Window
Search
Authenticated as cracky to directory: /LDAPv3/127.0.0.1
Name
ID
Directory Administ...
1000
0 of 1 user selected
Preferences
Overview
Details
Select one or more items from the list and click on a preference below to make changes.
Applications
Classic
Dock
Finder
Internet
Login
Media Access
Mobility
Network
Printing
Software Update
System Preferences
Universal Access
Preference is being managed for the selected items
Preference is being managed for some of the selected items

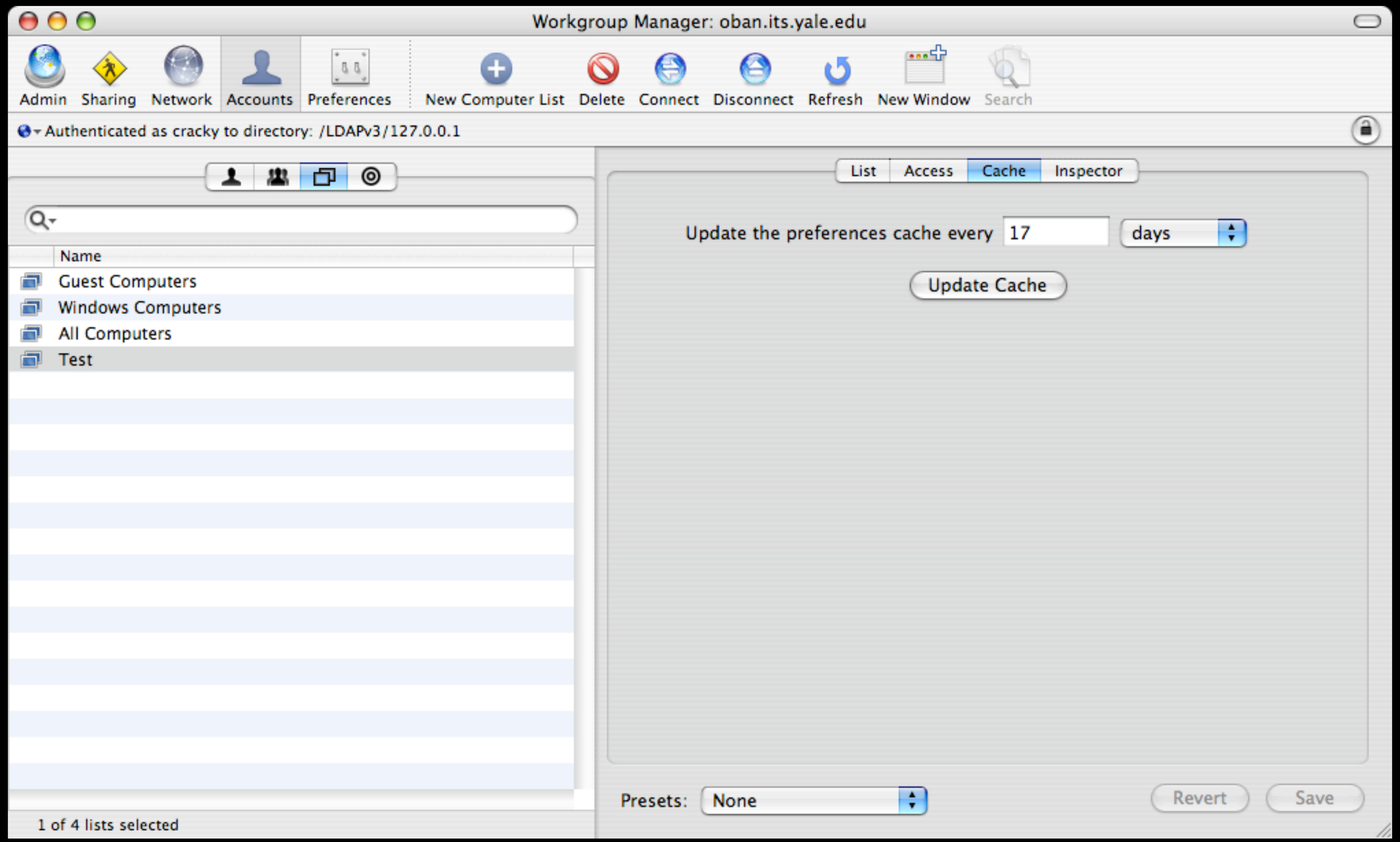
One example


Workgroup Manager: oban.its.yale.edu
Admin
Sharing
Network
Accounts
Preferences
New Computer List
Delete
Connect
Disconnect
Refresh
New Window
Search
Authenticated as cracky to directory: /LDAPv3/127.0.0.1
Name
Guest Computers
Windows Computers
All Computers
Test
1 of 4 lists selected
Login Items
Scripts
Login Window
Options
Manage:
Never
Once
Always
Hide
Item
Kind
Mount item with user's name and password
Drag items to specify the order in which they open. To hide an application after login, click its Hide checkbox.
Remove
Add...
Add network home share point
User may add and remove additional items
User may press Shift to keep items from opening
Merge with user's items (Mac OS X v10.4 and later)
Done
Revert
Apply Now

Access


Workgroup Manager: oban.its.yale.edu
Admin
Sharing
Network
Accounts
Preferences
New Computer List
Delete
Connect
Disconnect
Refresh
New Window
Search
Authenticated as cracky to directory: /LDAPv3/127.0.0.1
List
Access
Cache
Inspector
Name
Guest Computers
Windows Computers
All Computers
Test
1 of 4 lists selected
All groups can use the computer
Restrict to groups below
Name
ID
Allow users with local-only accounts
Local-only accounts pick workgroups from the above list
(OS 10.4 clients and later only)
Allow computer administrators to disable management
(OS 10.4 clients and later only)
Presets:
None
Revert
Save

Cache settings



The “Triangle”

Active Directory

Open Directory

Active Directory

Open Directory

Authentication

Active Directory

Open Directory

Authentication

Management

Caveats

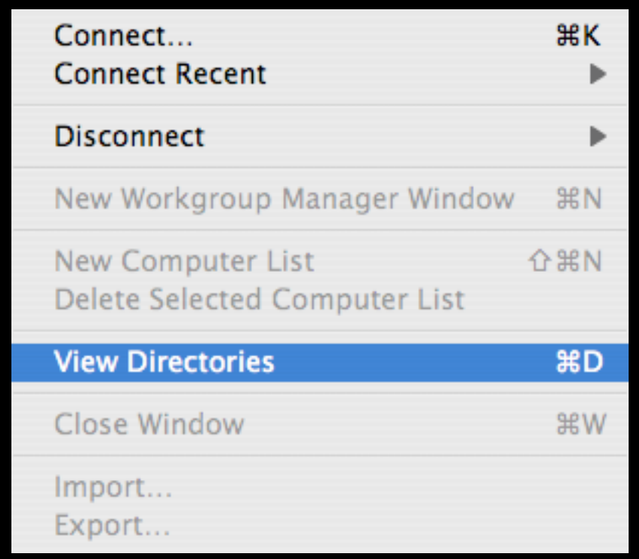
- Standard behavior is to have Active Directory first in Authentication path in Directory Services
- DNS resolution is important and can cause managed preferences oddities
- Need to remove or disable OD kerberos config to prevent conflicting authentication information
  - Alter Kerberos attribute
  - Remove kdc

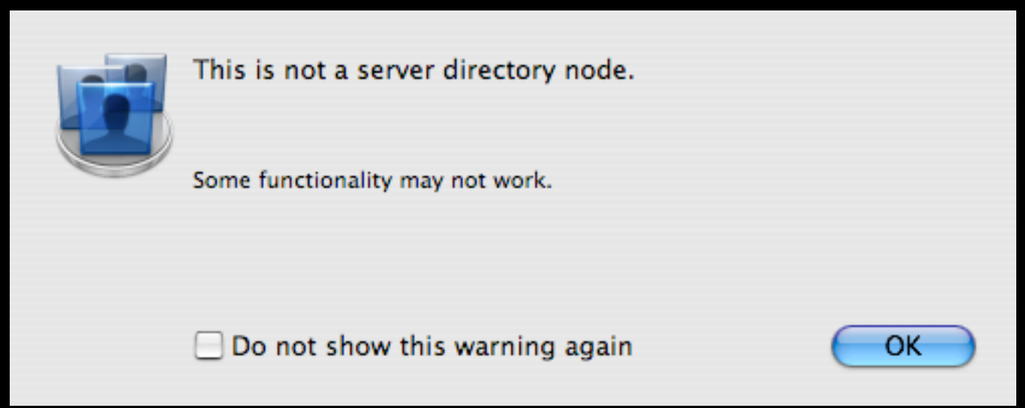
Applying mcx locally

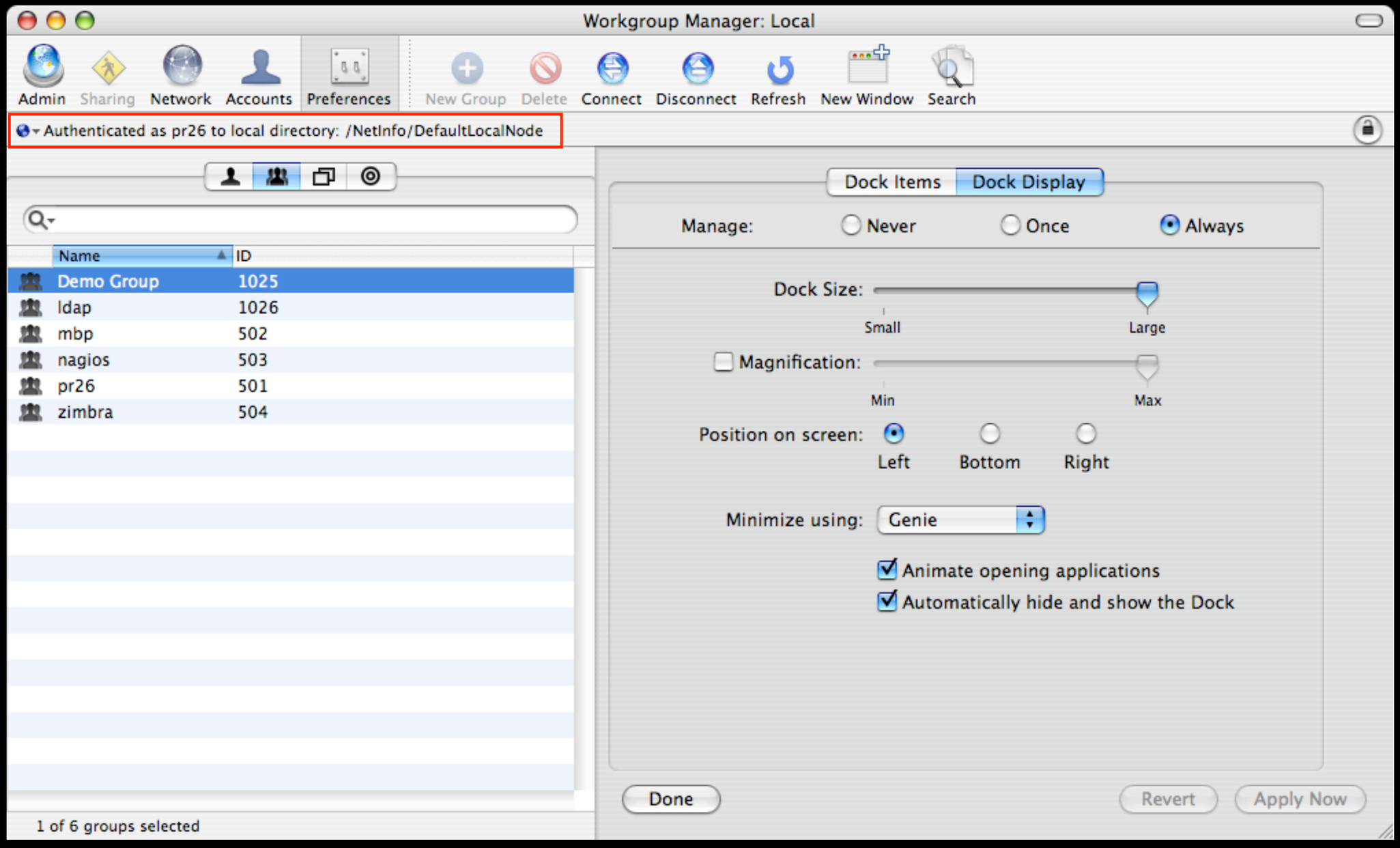
What?

- No need for OS X server
- Management is “point-in-time”
- User and group settings work, computer settings may not

How?


Connect... ⌘K
Connect Recent
Disconnect
New Workgroup Manager Window ⌘N
New Computer List ⇧⌘N
Delete Selected Computer List
View Directories ⌘D
Close Window ⌘W
Import...
Export...

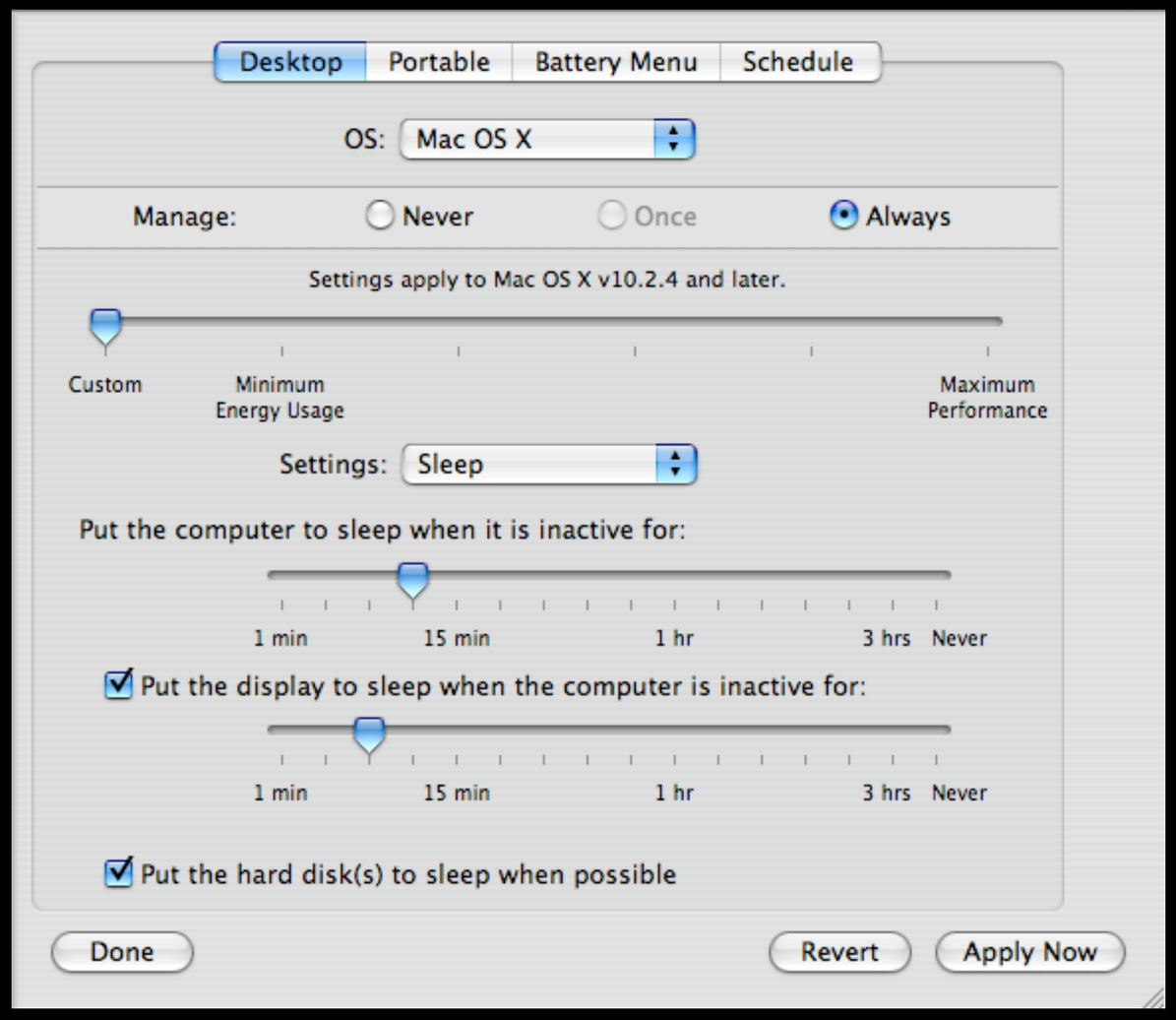

This is not a server directory node.
Some functionality may not work.
Do not show this warning again
OK


Workgroup Manager: Local
Admin
Sharing
Network
Accounts
Preferences
New Group
Delete
Connect
Disconnect
Refresh
New Window
Search
Authenticated as pr26 to local directory: /NetInfo/DefaultLocalNode
Name
ID
Demo Group 1025
ldap 1026
mbp 502
nagios 503
pr26 501
zimbra 504
1 of 6 groups selected
Dock Items
Dock Display
Manage:
Never
Once
Always
Dock Size:
Small
Large
Magnification:
Min
Max
Position on screen:
Left
Bottom
Right
Minimize using:
Genie
Animate opening applications
Automatically hide and show the Dock
Done
Revert
Apply Now

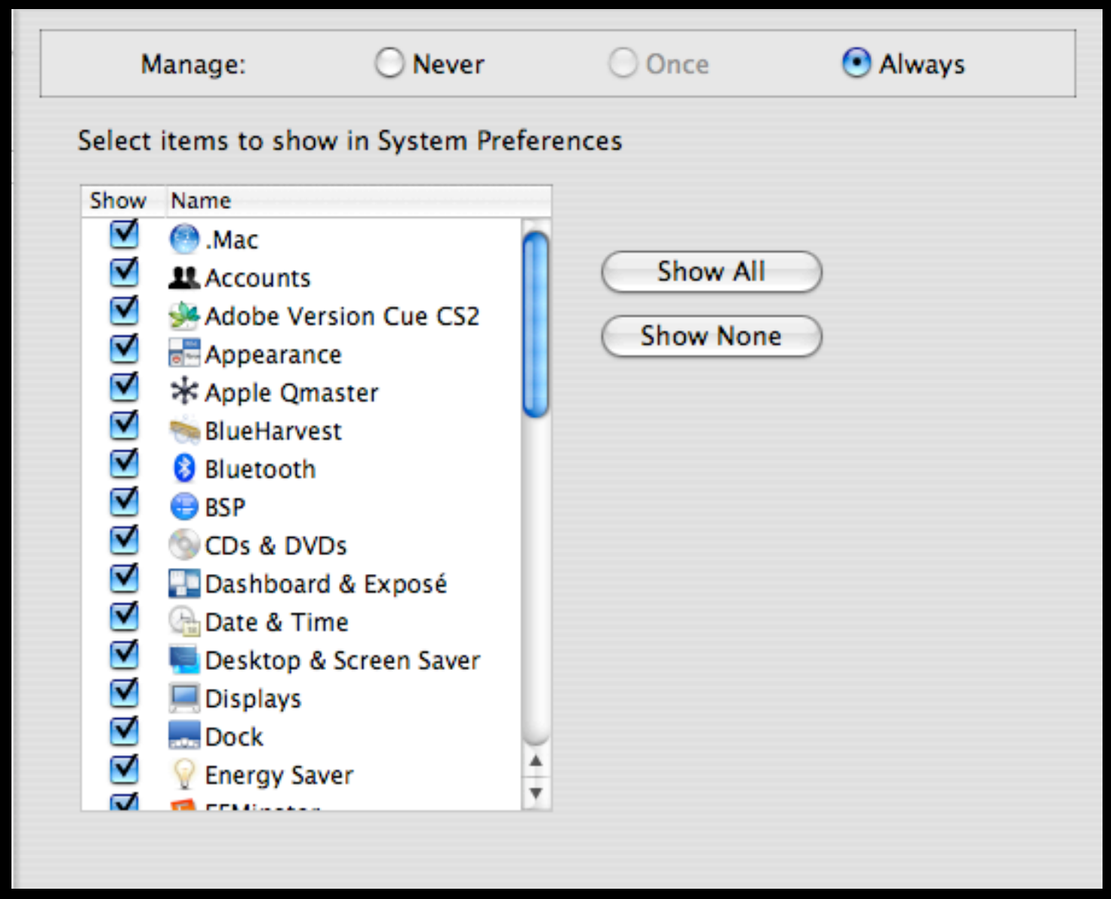
Preference management

Light touch

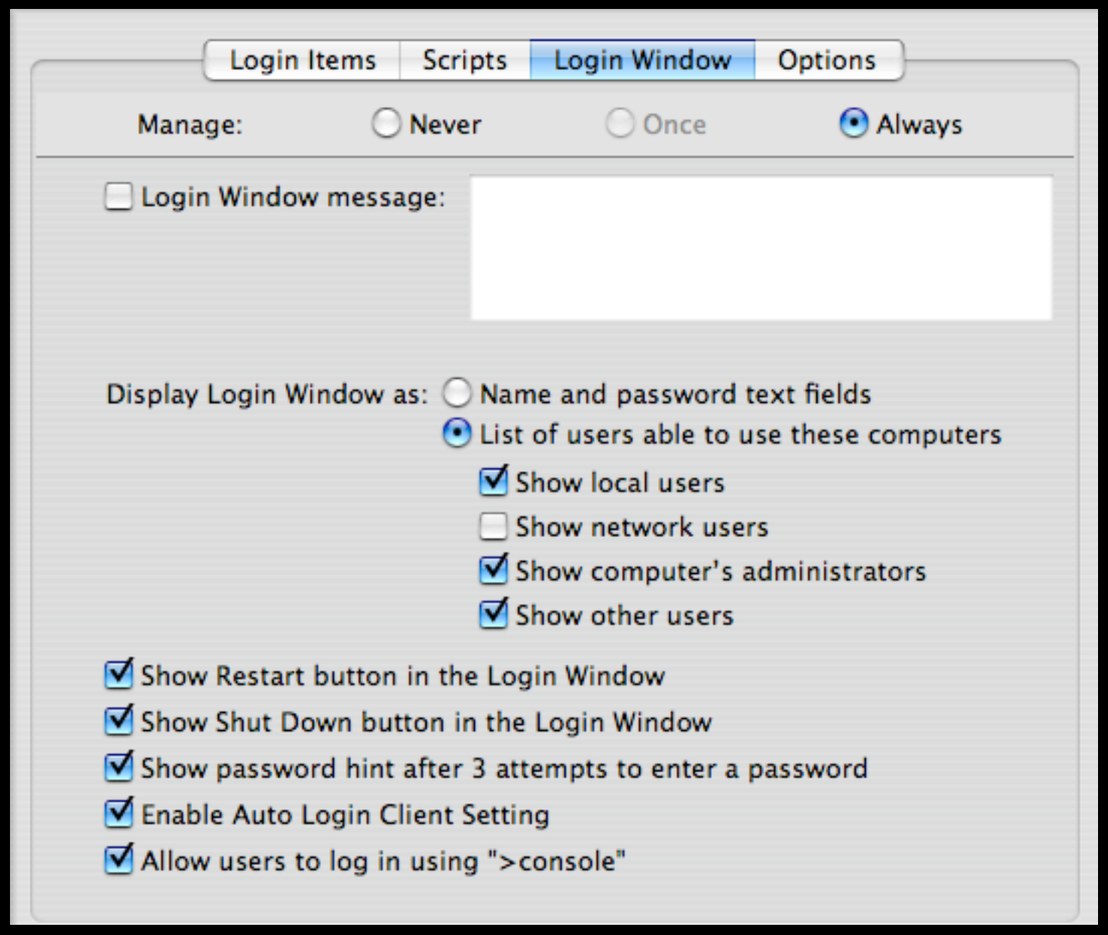
Energy Saver



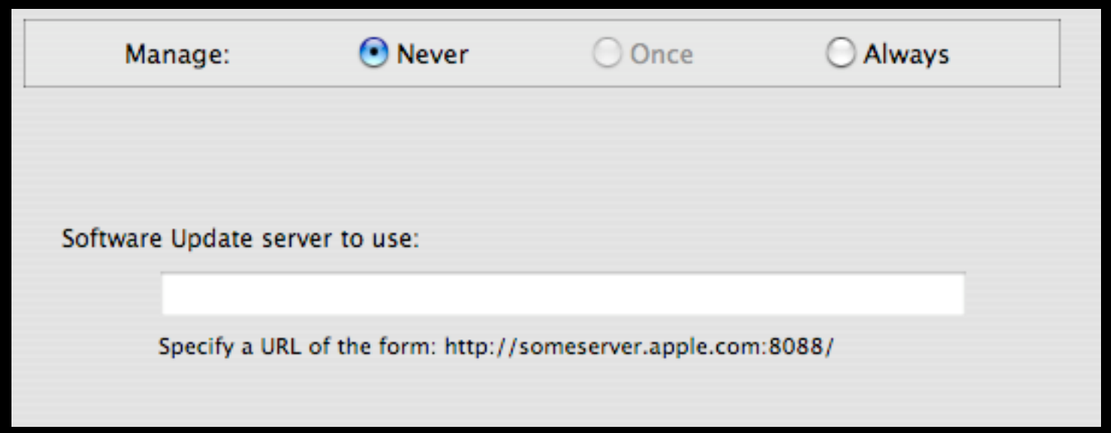
System Preferences



Loginwindow

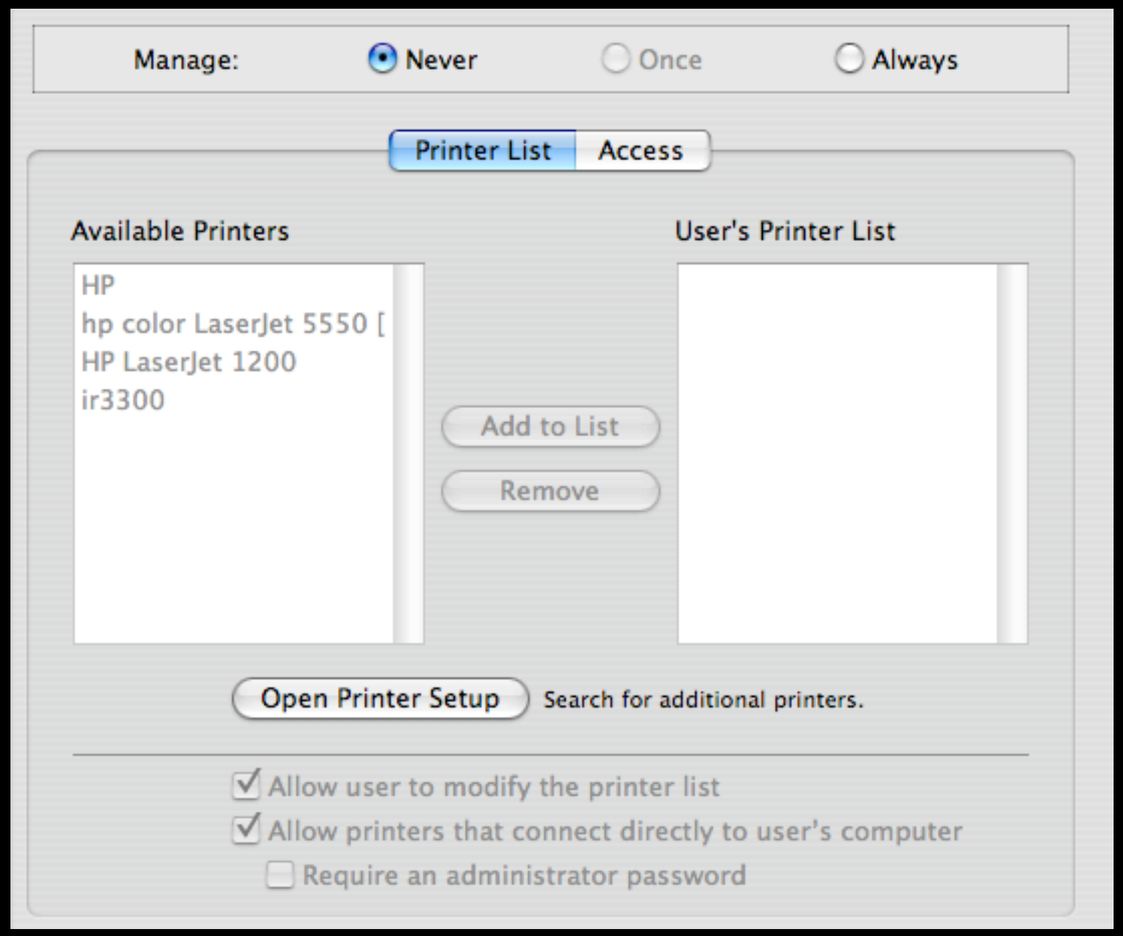


Software updates



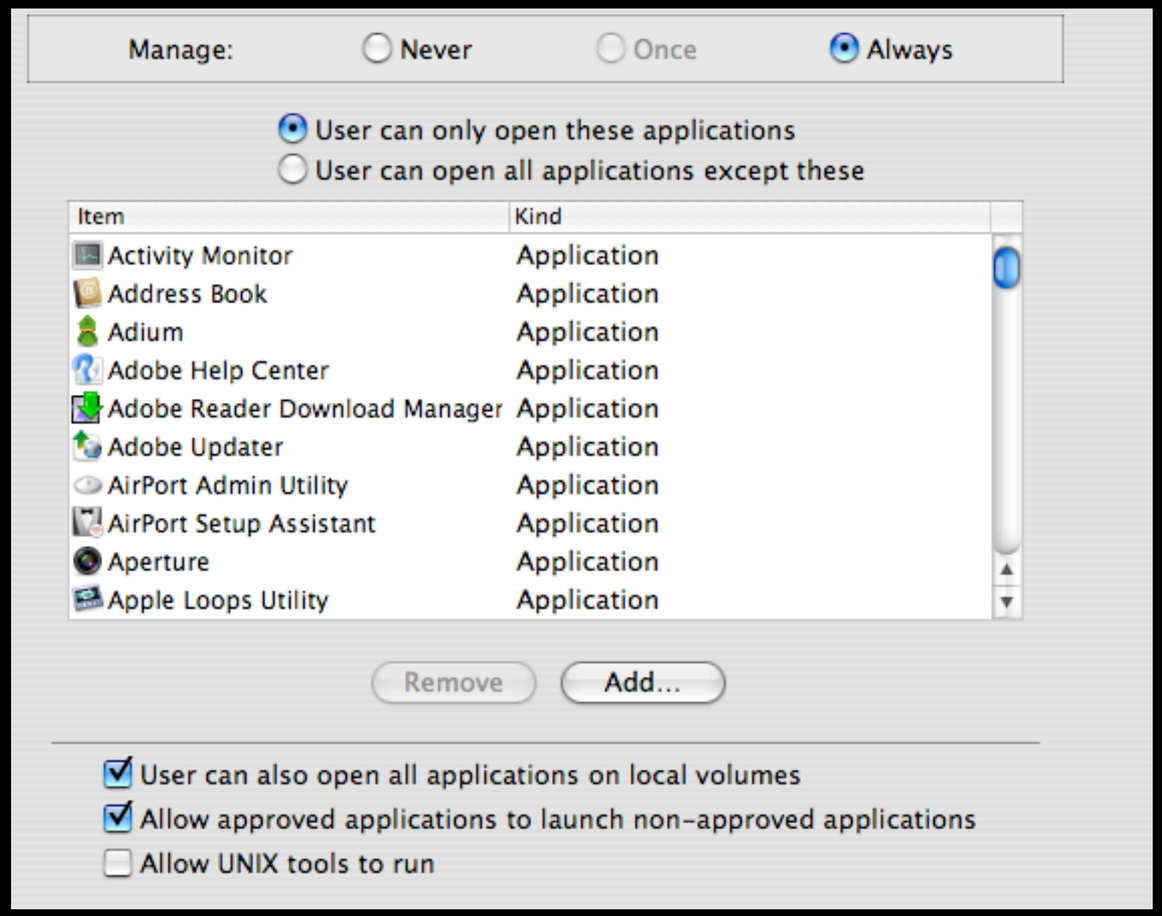
Middle ground

Printers

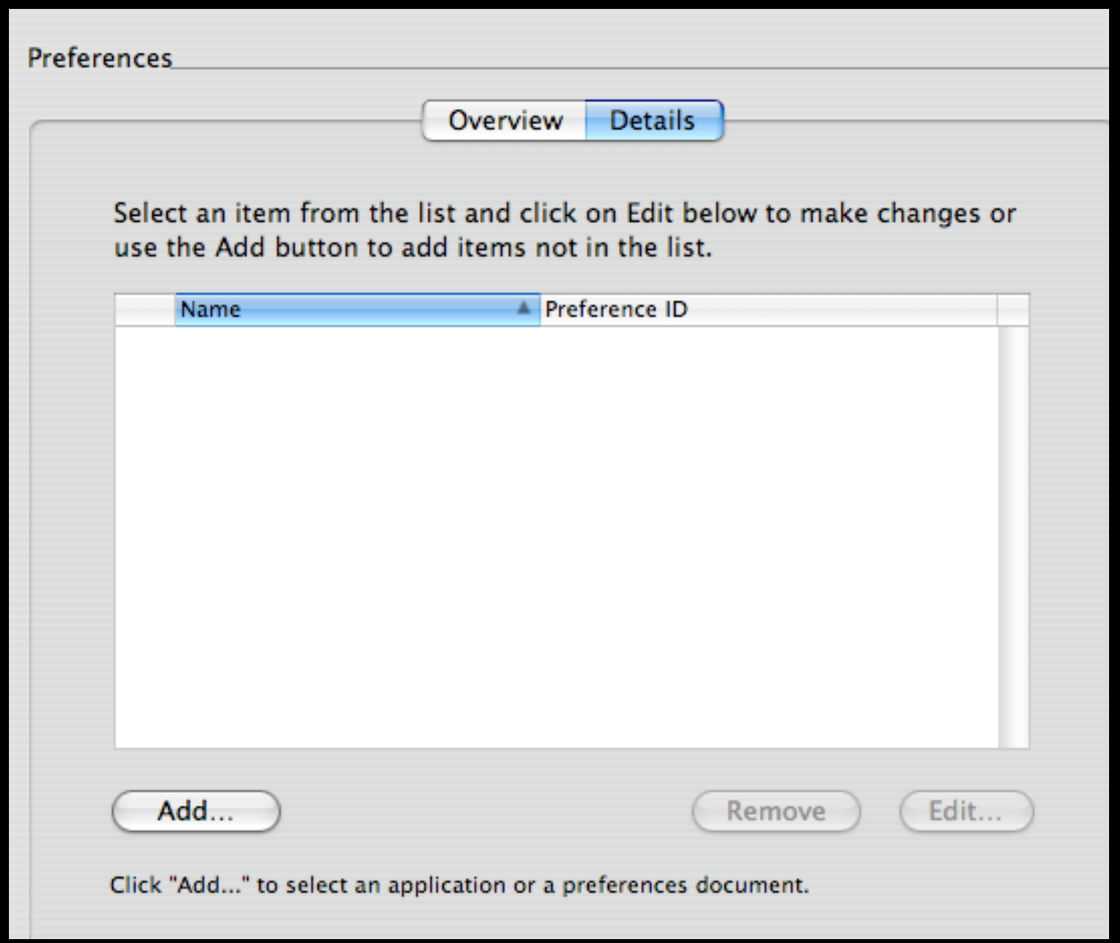

Manage:
Never
Once
Always
Printer List
Access
Available Printers
User's Printer List
HP
hp color LaserJet 5550 [
HP LaserJet 1200
ir3300
Add to List
Remove
Open Printer Setup
Search for additional printers.
Allow user to modify the printer list
Allow printers that connect directly to user's computer
Require an administrator password

Heavy

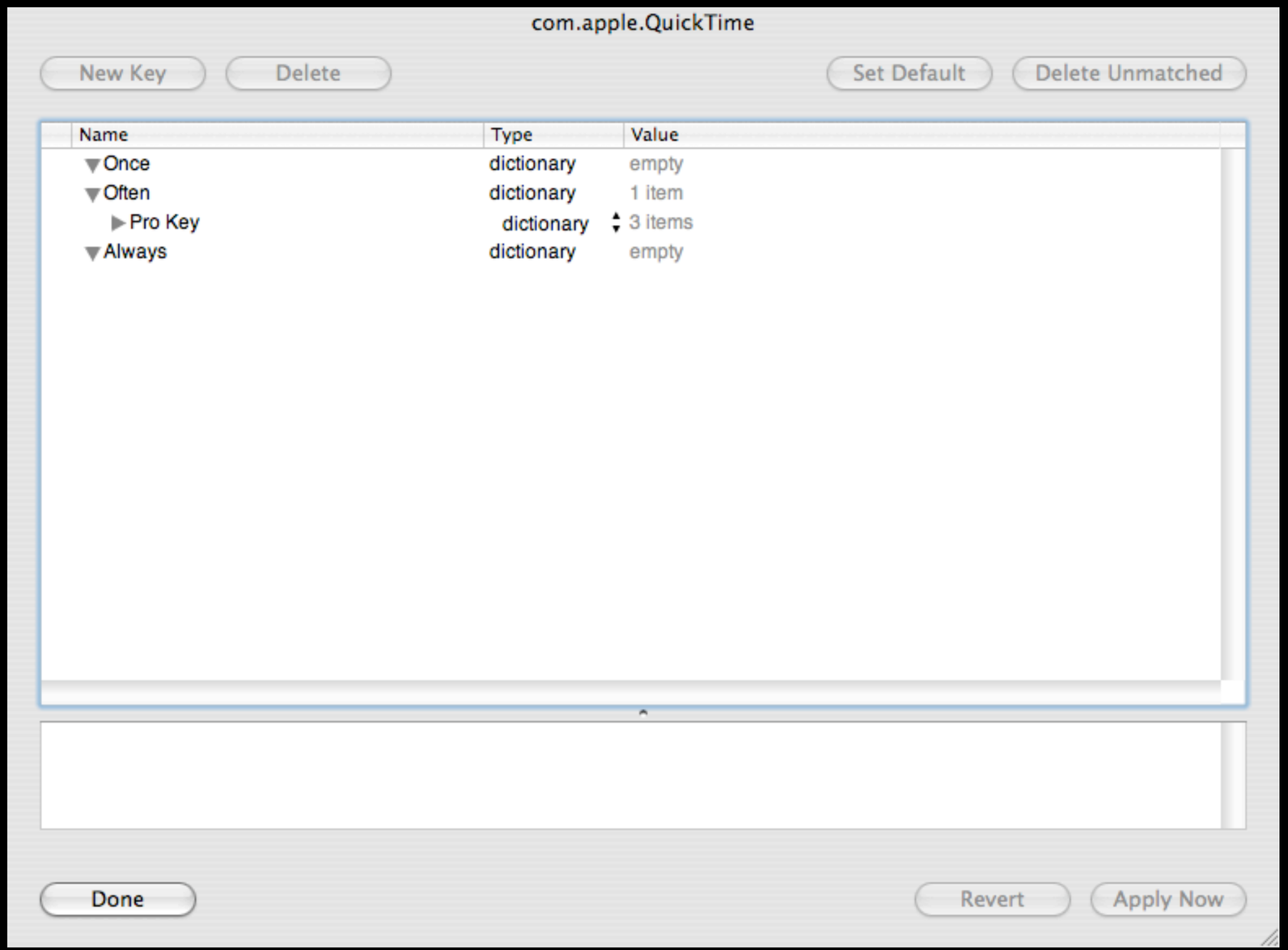
Application management



Manifest inspector



An example

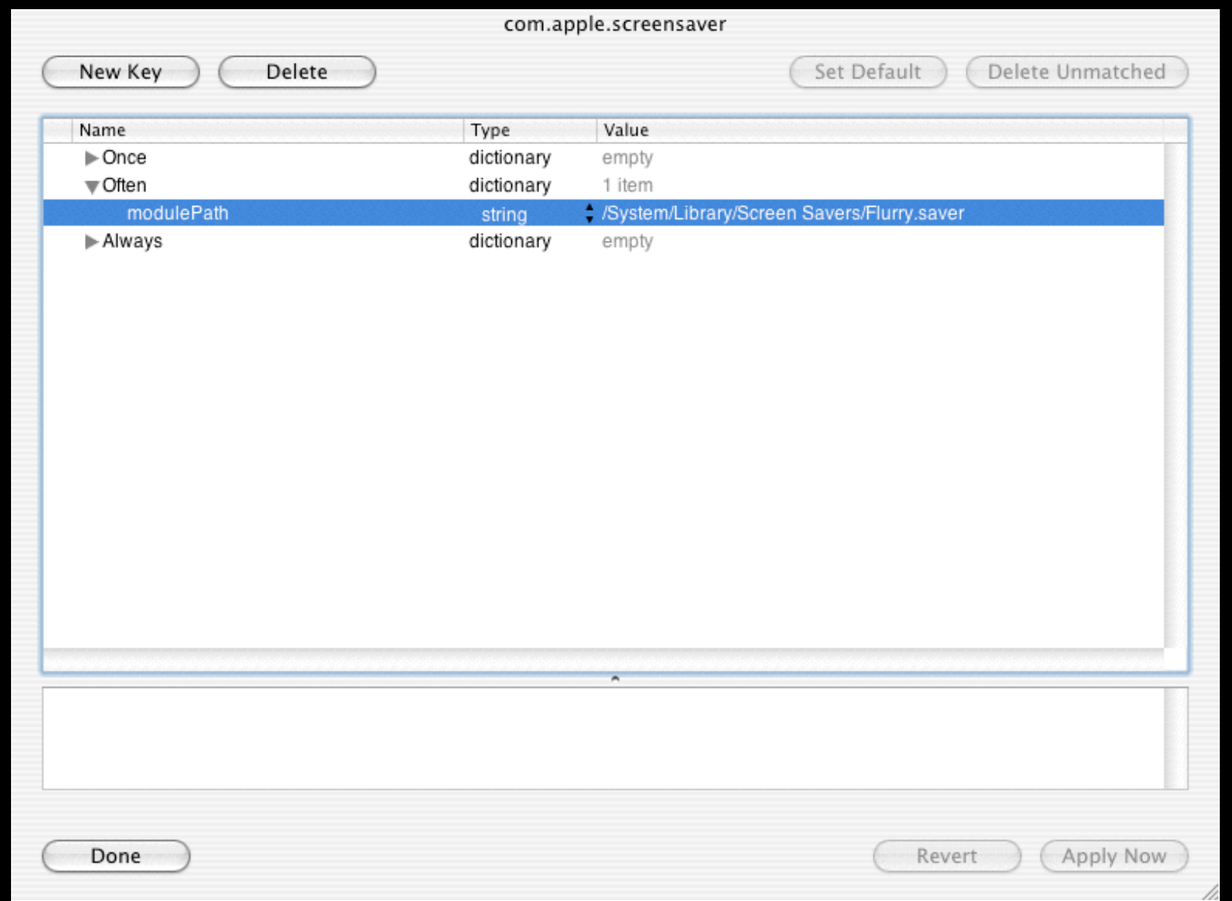

com.apple.QuickTime
New Key
Delete
Set Default
Delete Unmatched
Name
Type
Value
Once
dictionary
empty
Often
dictionary
1 item
Pro Key
dictionary
3 items
Always
dictionary
empty
Done
Revert
Apply Now

Possible candidates

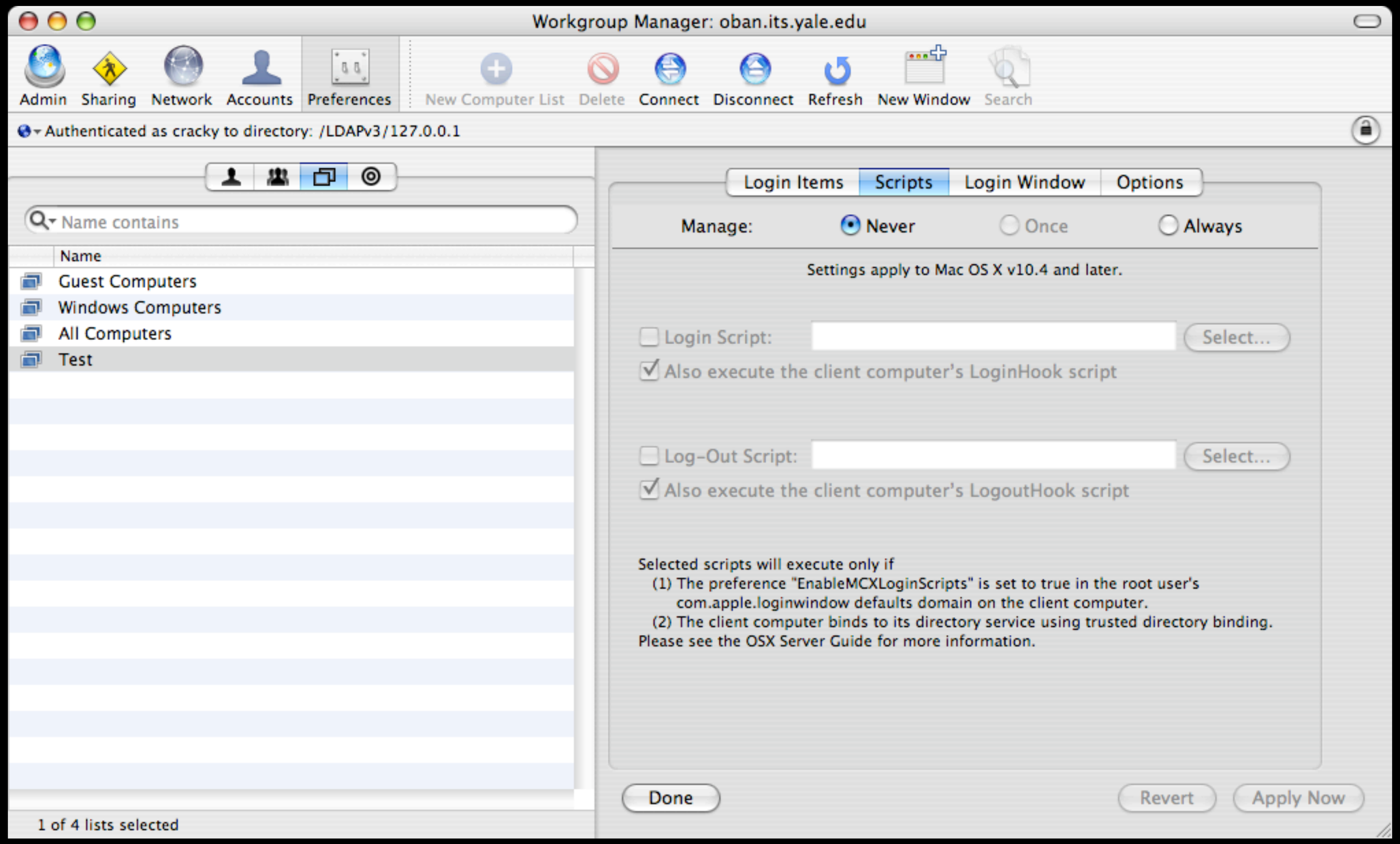
com.apple.desktop.plist

com.apple.iWork.plist

com.apple.screensaver.*



Login scripts



Possible login items to script?

- Microsoft Word settings?
- Printers?
- Dock specific settings?
- IMPORTANT: Must set key on client workstation

Network Directories

a.k.a. Portable Home Directories

Advantages

- Preferences are consistent across multiple machines
  - Language settings
  - Printers
  - System Preferences
- Central backup of user data can be done at the server level
- Machines become expendable (labs, offices)

Disadvantages

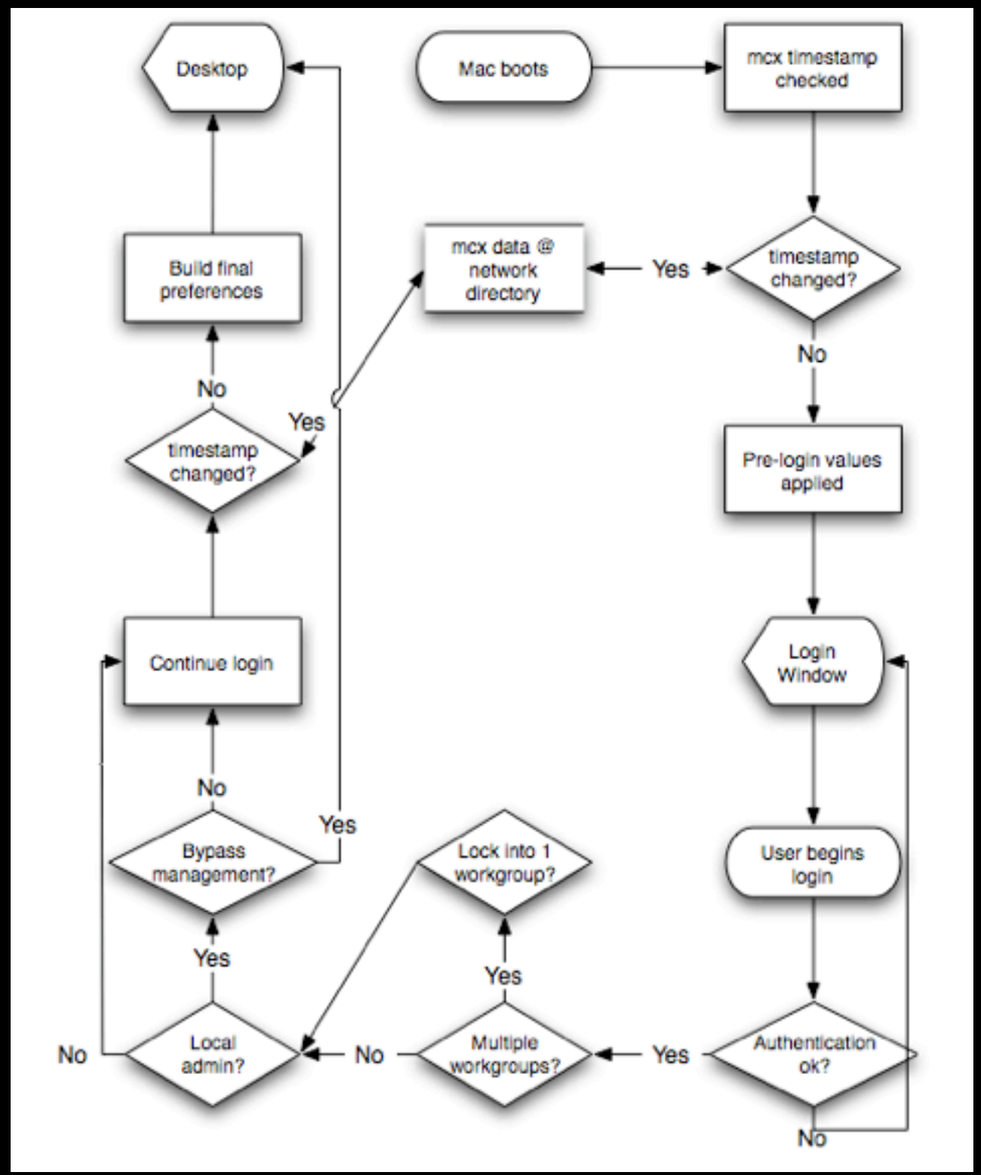
- Some applications are not ‘network-friendly’
  - Office, Adobe CS Suite, etc.
- Disk intensive applications will not perform well
  - iMovie, Final Cut Pro, et al.

- Requires a fairly robust hardware/network combination

- No quota reporting

Troubleshooting

MCX diagram


Desktop
Mac boots
mcx timestamp checked
Build final preferences
mcx data @ network directory
Yes
timestamp changed?
No
No
Yes
timestamp changed?
Pre-login values applied
Continue login
Login Window
No
Yes
Bypass management?
Lock into 1 workgroup?
User begins login
Yes
Yes
No
Local admin?
No
Multiple workgroups?
Yes
Authentication ok?
No

MCXCacher

```
/System/Library/CoreServices/mcxd.app/Contents/Resources/MCXCacher

MCXCacher -U usershortname [-h homepath]
Creates (or overwrites an existing) mobile account on the current
machine for user "usershortname" with optional home path "homepath"

MCXCacher
Performs the pre-login checks and refreshes cache if required.

MCXCacher -u usershortname
Performs the post login checks and refreshes caches -- does
everything that "MCXCacher" does plus caches the current user's
mcx_settings

MCXCacher -f
Flushes the cache (Mobile accounts not removed; but system is unmanaged)

MCXCacher -d
Dirties the cache so that it will be refreshes at the next login
("MCXCacher" call by mcxd)
----------------------

MCXCacher -f will put the machine into an unmanaged state until the
next time it reconnects to the management server, so it's a rather
drastic thing to do. From memory, if you run this command, Mobile
Users won't be able to login at the loginwindow unless the machine
can connect to the management server at that time.

MCXCacher -d does the right thing in the vast majority of cases, and
as John DeTroye just pointed out to me, runs at login/logout and
restart.
```

Compositor.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist SYSTEM "file://localhost/System/Library/DTDs/
PropertyList.dtd">
<plist version="0.9">
<dict>
   <key>PreLoginAllowBundles</key>
   <array>
     <string>com.apple.loginwindow</string>
     <string>com.apple.MCX</string>
     <string>com.apple.mcxloginscripts</string>
     <string>com.apple.SoftwareUpdate</string>
   </array>
</dict>
</plist>
```

CinchDefaults.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN" "http://
www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
   <key>excludedAlways</key>
   <array>
     <dict>
       <key>comparison</key>
       <string>fullPath</string>
       <key>value</key>
       <string>~/Library/Mirrors</string>
     </dict>
....
```

Debugging

- Two caches
  - computer
  - user
- Turning on debugging
  - `defaults write /Library/Preferences/com.apple.MCXDebug debugOutput 3`

Real World:

Ventura County Star

Before:

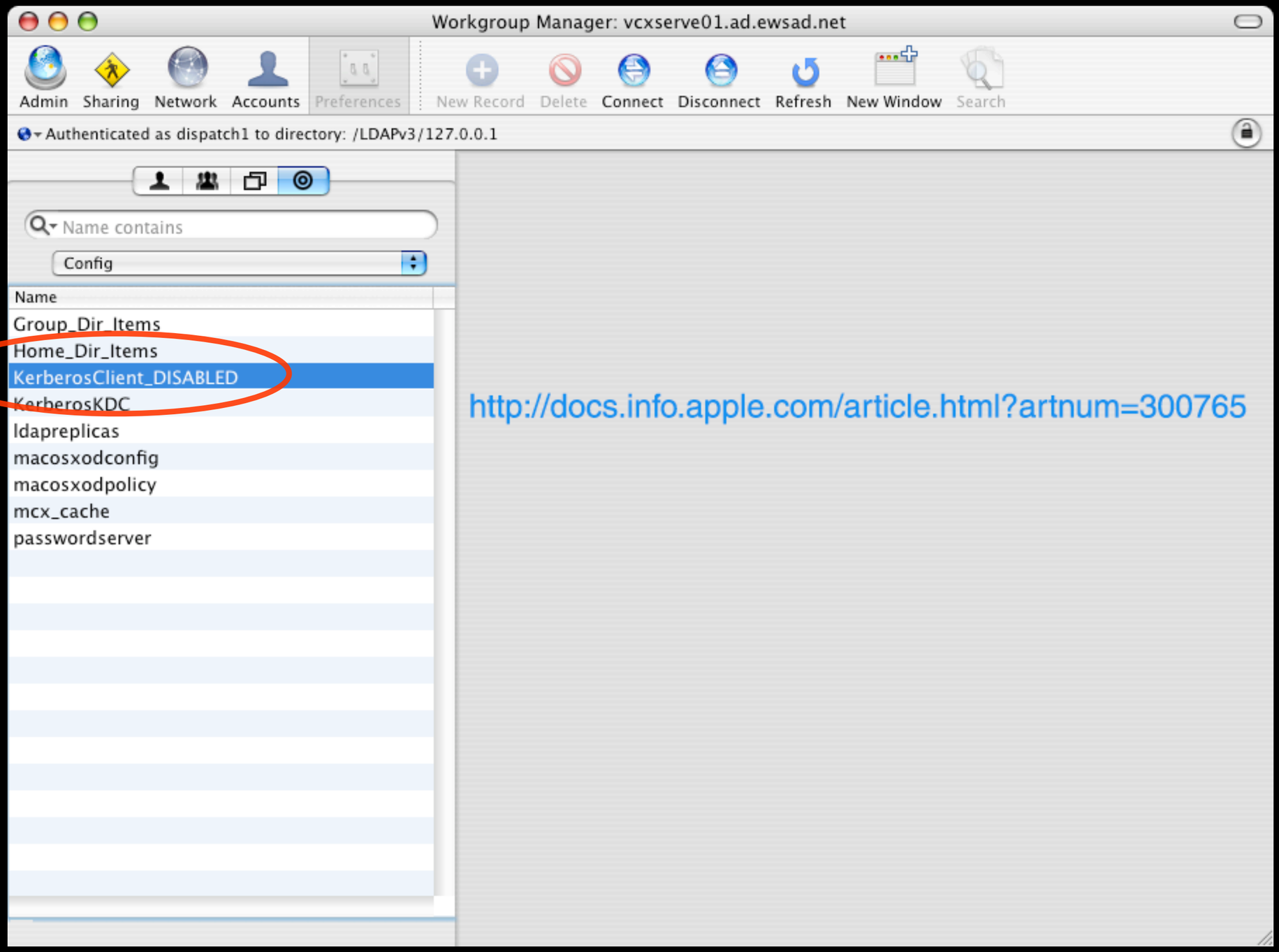
- Mac OS 9
- MacAdministrator
  - User folder of files preserved and copied
  - No preferences copied
  - All control panels locked
- AppleShare IP 6.3
- Windows NT 4.0 (some 2000)

After:

- Mac OS X
- AD-OD Integration (golden triangle)
- Mobile accounts with portable home directories
  - Entire user environment preserved and moved
  - But with exceptions (cache, etc.)
  - Limited access to system preferences
- XServe G5 Open Directory Master
- Bound to AD for authentication

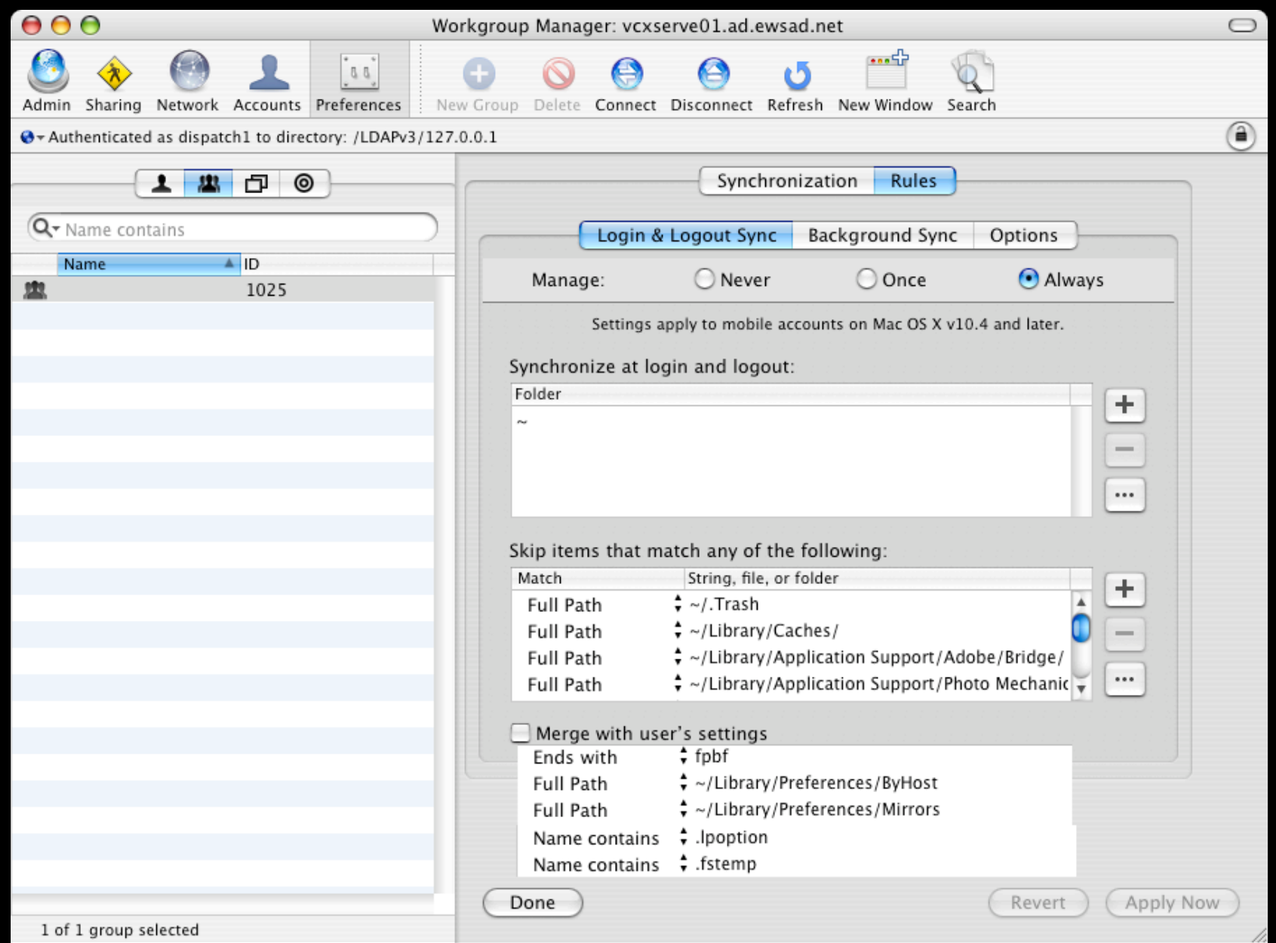
What Happened:

- First time system was loaded:
  - 15-20 minute boot times
  - Random syncing errors:
    - “Your AFP home cannot be synchronized right now because it is on an SMB or AFP home.” Pardon?
- Additional load placed on system:
  - Network failed (lesson: make sure your infrastructure is sound, *ahead of time*)



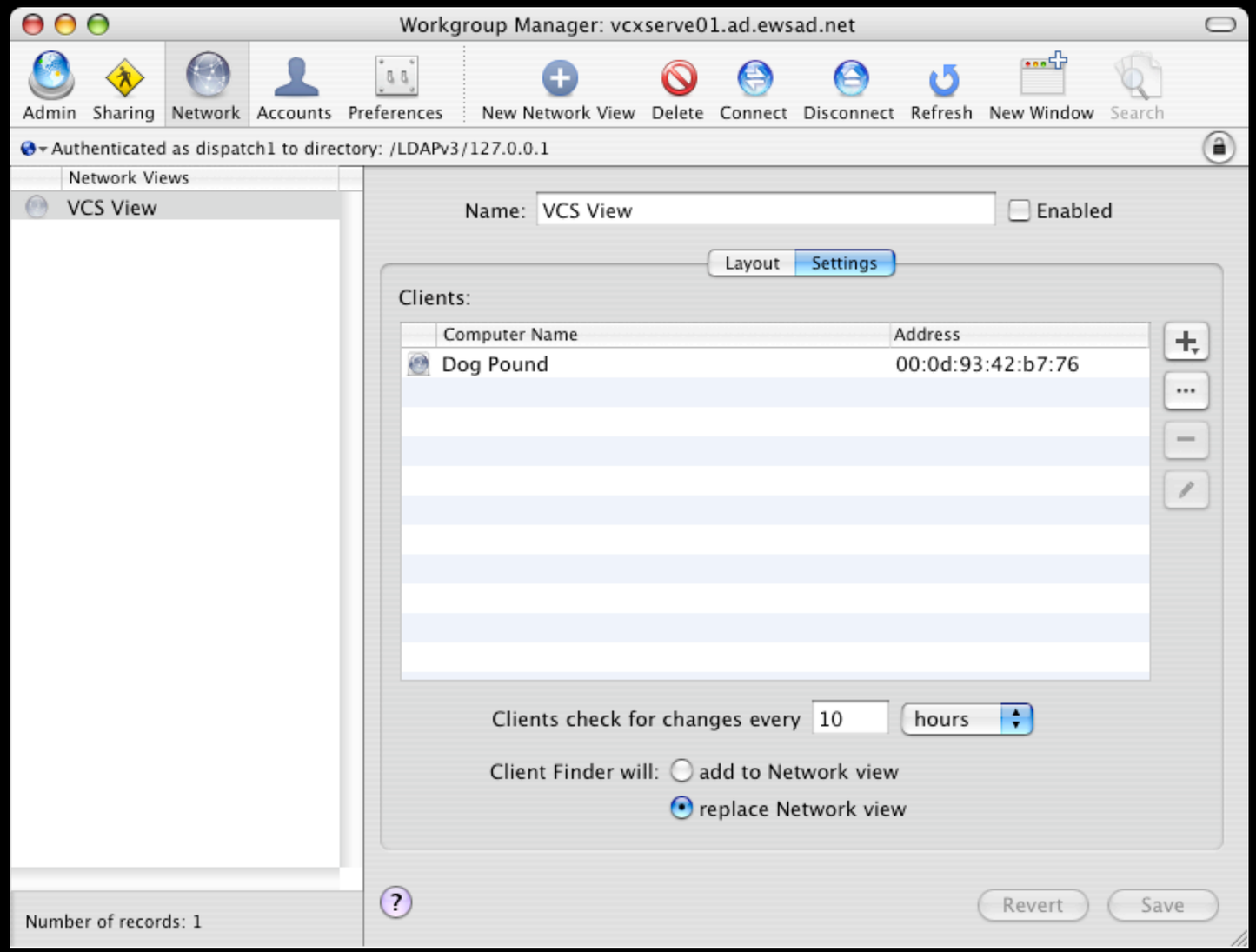
Lessons Learned

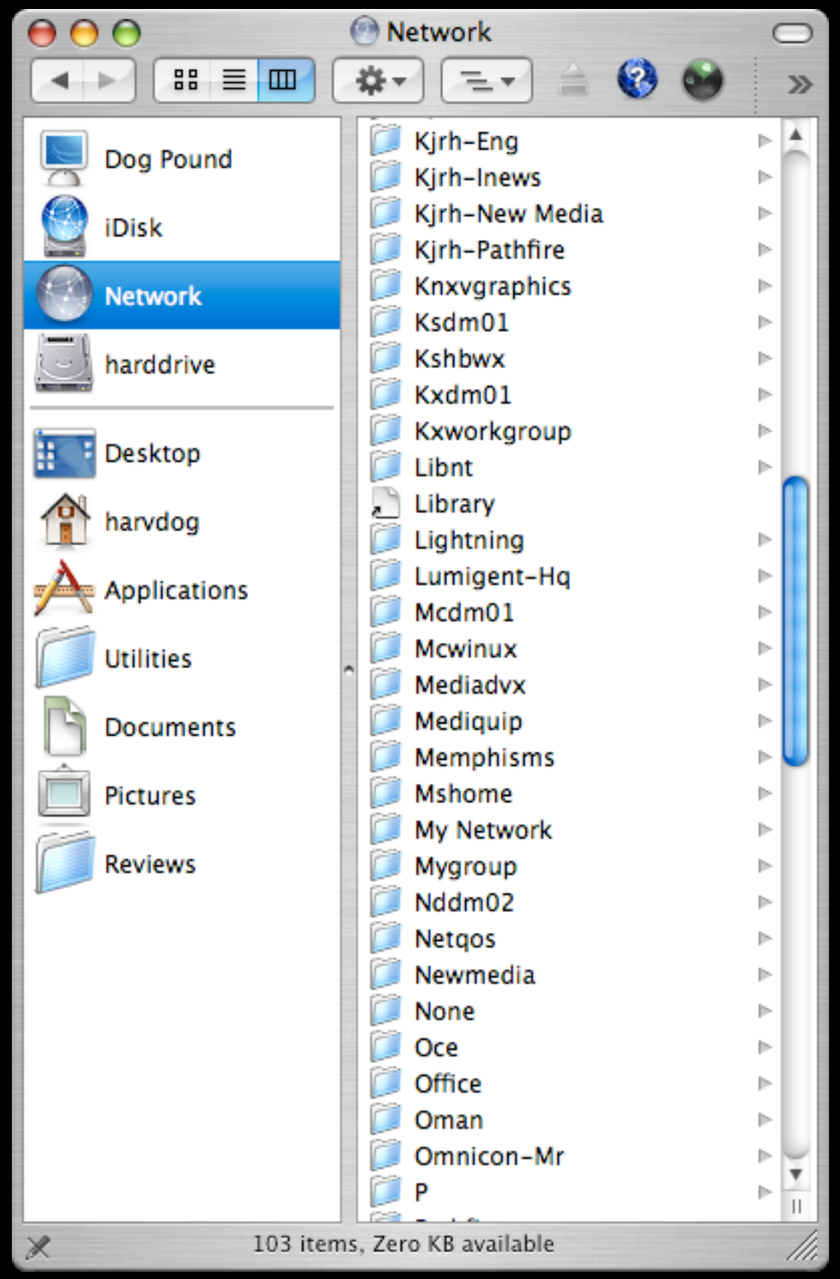
- Take your time
- Don’t lose your notes
- ARD is your friend
- Scour the Library for cache files to exclude



To do:

- Manage Network views
  - What it looks like now
- Creating a view in WGM





To do:

- Migrate OD Master to Intel XServe

- Move!