Mac OS X Server Administrator Topics

Mac OS X Server

2007-05-23

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Contents

Tables

Introduction to Mac OS X Server Administrator Topics

Mac OS X Server combines the most popular open source server technologies with Apple's easy-to-use setup, management, and monitoring tools. Mac OS X Server is based on open standards, so it integrates easily with existing networks. It comes with an extensive suite of services for supporting Macintosh, Windows, and mixed-client environments in businesses, departments, and educational institutions.

This document provides information on solving problems, adding functionality, and performing advanced operations on Mac OS X Server. The topics in this document supplement the more general instructions located in the Mac OS X Server Reference Library.

Important: This is a preliminary document. Although it has been reviewed for technical accuracy, it is not final. Apple Computer is supplying this information to help you plan for the adoption of the technologies described herein. This information is subject to change, and software implemented according to this document should be tested with final operating system software and final documentation. For information about updates to this and other developer documentation, you can check the ADC Reference Library Revision List. To receive notification of documentation updates, you can sign up for a free Apple Developer Connection Online membership and receive the bi-weekly ADC News e-mail newsletter. (See http://developer.apple.com/membership/ for more details about ADC membership.)

Who Should Read This Document?

These topics are intended to assist system administrators providing services utilizing Mac OS X and Mac OS X Server. Readers should be familiar with terminology and methodology related to system administration. Readers should also be comfortable editing configuration files and executing commands using the Terminal.

Organization of This Document

This document contains the following articles:

- "Kerberizing the Mac OS X VPN Server" (page 9) guides you though configuring the Mac OS X VPN server to use Kerberos authentication with a third-party directory.
- "Creating Mail Account Bundles" (page 11) walks you though creating an add-in for Mail to assist users setting up email accounts.
- "Setting Up a Network Library" (page 17) explains how to create a share point for library-based resources such as fonts, Internet plug-ins, and user images that will be available to all network-connected users.
- "Setting Up Key-Based SSH Login" (page 25) discusses how to configure Mac OS X Server to use key-based SSH.
- "Incorporating User Images in Open Directory" (page 27) explains the process of adding user images into Open Directory and describes how the images are used in Mac OS X.

- "Creating Automator Actions for Apple Remote Desktop" (page 29) explains how to create Automator Actions for Apple Remote Desktop 3.
- "Installation and Best Practices" (page 33) discusses the best practices for installing, configuring, and maintaining Mac OS X Server.

See Also

Additional documentation about Mac OS X Server can be found in the Mac OS X Server Reference Library.

Kerberizing the Mac OS X VPN Server

In Mac OS X Server v10.4, the Layer 2 Tunneling Protocol (L2TP) virtual private network (VPN) server can use Kerberos to authenticate users. Kerberos authentication is the only way to have the Mac OS X VPN Server authenticate users against a third-party directory. This article will cover the steps you need to take to enable Kerberos authentication for the VPN server when it is used with a third-party directory.

Currently Mac OS X is the only operating system that has a VPN client that supports Kerberos authentication. There is a known limitation that the `edu.mit.kerberos` file that defines the realms for the Kerberos utility in Mac OS X can contain only one realm when utilizing Kerberos authentication for VPN access. Having only one defined realm is not an issue for most system administrators, because there is typically only one Kerberos realm defined for each top-level domain and its users. This realm must be a Kerberos version 5 realm. This limitation will be addressed in a future update or release.

This topic addresses how to integrate the Mac OS X VPN server into a third-party directory for authentication and does not cover the details of Kerberos. For instructions on using Directory Access to integrate a Mac OS X Server into a third-party directory, see Chapter 7, "Managing Directory Access" in *Mac OS X Server Open Directory Administration*. Contact the vendor responsible for the operating system for more information about the availability of Kerberized VPN clients. For more information about Kerberos, see the MIT Kerberos website.

Installing the Kerberos Keytab

The VPN server, built-in to Mac OS X Server v10.4, needs a Kerberos version 5 keytab for the realm it will serve. The keytab must contain the principle `vpn/exampleserver.exampledomain.com@EXAMPLEREALM.COM` where `exampleserver.exampledomain.com` is the fully qualified domain name of the server. For example, `vpn/vpnserver.apple.com@APPLE.COM` is a valid keytab request. If the server provides additional kerberized services, those principles must be in the same keytab.

1. The Kerberos keytab is located at `/etc/krb5.keytab`. If that file does not exist, move the keytab you receive into `/etc` name it `krb5.keytab`, and move to step 3. If the `/etc/krb5.keytab` file exists run the following commands in step 2.

2. In the Terminal application, run the command `sudo ktutil` to open the keytab utility. You see `ktutil:` as a prompt, all of the following commands are run at the `ktutil` prompt, not the Terminal prompt. Each new line is a separate command.

```
read_kt /etc/krb5.keytab

read_kt /your/new/keytab/file.keytab

write_kt /etc/krb5.keytab

quit
```

3. Make sure the owner of the file is `root` and the file permissions are set so that only the file's owner, `root`, is able to read and write the file. The permissions for `group` and `other` must be set to not allow access to the keytab file.

4. In Terminal, run the command `sudo klist -k /etc/krb5.keytab`, and make sure all of the principles you expect in the keytab are present.

Configuring the VPN Server

Before turning on the VPN Server in the Server Admin application, you must perform the following commands to kerberize the VPN server:

1. In Terminal, run the following command as a single line.

   ```
   sudo serveradmin settings
   vpn:Servers:com.apple.ppp.l2tp:EAP:KerberosServicePrincipalName =
   "vpn/exampleserver.exampledomain.com@EXAMPLEREALM.COM"
   ```

   This command sets the service principal name for the VPN server to match the principle in the keytab you just installed.

2. The next step is to change the configuration of the VPN server so it does not use access control lists which rely on globally unique identifiers (GUIDs) that are not present in the third party directory. To do this, open the file `/Library/Preferences/SystemConfiguration/com.apple.RemoteAccessServers.plist` in an application that can edit property lists. The Property List Editor application is the recommended application to use. If you do not have this application, install the Developer Tools package that accompanies Mac OS X v10.4. You will find the Property list Editor at `/Developer/Applications/Utlities`. In this file there is configuration information for both L2TP and PPTP. You only need to edit the information for the L2TP protocol. Find the key named `AuthenticatorACLPlugins` and rename it to `_AuthenticatorACLPlugins`. Save this file.

3. Start the VPN Server in the Server Admin application. Now that the VPN server is set up for Kerberos authentication, clients with the correct Kerberos configuration file and set up to use Kerberos authentication can utilize the VPN service.

You have now installed a Kerberos keytab, configured the VPN server to not look for GUIDs and set the realm for which the VPN server will use Kerberos to authenticate users. For more information about the VPN server built-in to Mac OS X Server v10.4, see *Mac OS X Server Network Services Administration*

Creating Mail Account Bundles

Starting with Mac OS X version 10.4, Mail has a new user interface for configuring accounts. The interface is a series of panels that prompts the user for information and verifies the information during the configuration process. You can provide users with a mail account bundle that streamlines the account setup process. Information such as the server name and connection options can be preset. Additionally, you can specify whether or not the user is allowed to change the values you have specified. There is also a setting that allows you to direct the user to a URL where they can get information to help with troubleshooting problems on their own. In this article, you will learn how to create a mail account bundle and you will see an example that is an excellent starting point for creating your own mail account bundles.

Creating a Bundle Directory

To create a mail account bundle, first create a new directory that will hold the files associated with the mail account bundle. To avoid name conflicts, the name of the mail account bundle directory should be the reverse-ordered ICANN domain name (for example, `com.apple`) and have the suffix `mailaccounts`. The bundle used to assist .mac users setting up mail accounts is installed in `/Library/Mail/AccountTypes` and is named `com.mac.mailaccounts`.

Next, create a XML file inside the bundle. It must be named `MailAccounts.plist`. This file should be an XML property list file with the keys listed in Table 1 (page 11). All keys are required unless explicitly indicated as optional. See the example at the end of this document for the proper nesting of keys.

Note: Some of the dictionary keys have been deprecated in Mac OS X v10.5. These keys may still be used for mail account bundles in Mac OS X v10.4 but are ignored in Mac OS X v10.5.

Table 1 Keys in `MailAccounts.plist`

Type	Description	Key
`AccountIconName` (deprecated)	String	Optional. The name of an icon file to use for this account. The icon should be 16 by 16 pixels and the file should be placed in the top level of the bundle directory.
`Accounts`	Array	An array of dictionary elements describing the account types. See "Account Dictionaries" (page 12) for details on the format of these dictionaries.
`Identifier`	String	The reverse-ordered ICANN domain name; for example, `com.mac`. This name must be unique.
`Version`	Integer	Currently, the `version` value must be `1`.

Account Dictionaries

Each dictionary in the Accounts array describes an account type that can be configured by the user. The keys in Table 2 (page 12)are supported. Unless indicated as optional, all keys are required for each account dictionary defined in the property list.

Table 2 Keys in each dictionary in `Accounts`

Key	Type	Description
`AccountID`	String	A CFUUID string, generated by running `uuidgen` in Terminal. For more information about CFUUIDs, see *CFUUID Reference*.
`AccountName`	String	The default account name for an account.
`AuthenticationScheme`	Array	Optional. An array of supported authentication schemes. An empty array or missing key causes accounts to be configured using `Password` security. List all authentication schemes that are supported by the account. The first item in the list is the preferred scheme. Supported schemes are: • `Password` • `CRAM-MD5` • `GSSAPI` • `KERBEROS_V4` (deprecated) • `X_KPOP` (deprecated) Both `KERBEROS_V4` and `X_KPOP` are now treated as `GSSAPI` in Mac OS X v10.5.
`AuthenticationScheme-IsEditable` (deprecated)	Boolean	Optional. Indicates whether the user is allowed to change the authentication scheme.
`DeliveryAccounts`	Array	Optional. This is an array of CFUUIDs associated with mail delivery accounts defined in the same mail account bundle. The first outgoing account defined in the array is used by default.
`EmailAddressDomain`	String	Optional. Indicates the domain name associated with the e-mail address. Mail uses this key to match the user's email address to the enclosing account dictionary.
`PortNumber`	Integer	Optional. A non-standard port number to use when connecting to the server. When specified, Mail attempts to connect to the port first before trying standard port numbers.
`PortNumberIsEditable` (deprecated)	Boolean	Optional. Indicates whether the user is allowed to change the port number used when connecting to the server.

Key	Type	Description
`Protocol`	String	Indicates what protocol to use when communicating with the server. Supported values are: • `IMAP` • `POP` • `SMTP`
`SSLEnabled`	Boolean	Optional. Indicates whether SSL should be used when communicating with the server. If this key is not specified, Mail attempts an SSL connection to the server first and falls back to non-SSL if that fails.
`SSLEnabledIsEditable` (deprecated)	Boolean	Optional. Indicates whether the user is allowed to change the setting of whether SSL is used or not. If this key is set to true and Mail fails to connect to the server using SSL, the user is shown the Incoming Security Settings pane of the setup assistant.
`ServerName`	Array	A list of server names to allow the user to pick from. The first server name is the preferred server and is used by default.
`ServerNameIsEditable` (deprecated)	Boolean	Optional. Indicates whether the user is allowed to change the server name.
`SupportURL`	String	Optional. The URL shown in the Mail Account Preferences panel that takes the user to the specified webpage for more help or information.
`SupportURLLabel`	String	Optional. When present, the specified value is displayed instead of the `SupportURL` value in the Mail account preference panel. Clicking on the label opens the URL stored in `SupportURL`.
`UserNameIsEmail-Address`	Boolean	Optional. Indicates whether the user name is the user's full email address.
`UserNameMatches-EmailAddress`	Boolean	Optional. Indicates whether the user name and the portion of the user's email address to the left of the @ sign are the same value.

Localizing Mail Accounts

You can localize values for the `AccountName` and `SupportURL` keys in mail account bundles. This lets each account have its own localizations. For more information on localization, see the Internationalization Documentation.

Using Simplified Account Setup

Mail provides a new feature in Mac OS X v10.5 that allows users to add a new account with only their full name, email address, and password. This feature is known as simplified account setup.

Mail account bundles are automatically used for simplified account setup if the following conditions are met:

- The `EmailAddressDomain` dictionary key must be specified.
- The `UserNameIsEmailAddress` or `UserNameMatchesEmailAddress` dictionary key must be set to true.
- The `DeliveryAccounts` dictionary key must have exactly one CFUUID.

Installation

To install a mail account bundle, you need to place the top-level directory, which contains the `MailAccount.plist` file, in one of the following locations:

- `~/Library/Mail/AccountTypes`
- `/Library/Mail/AccountTypes`
- `/Network/Library/Mail/AccountTypes`

A Mail Bundle Property List Example

The following example is the property list used to create the .Mac mail account bundle shipping with Mac OS X v10.4 and later:

```
<?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>Accounts</key>
    <array>
        <dict>
            <key>AccountID</key>
            <string>5D795DD4-55CB-11D8-A81E-000A957054BE</string>
            <key>AccountName</key>
            <string>.Mac</string>
            <key>AuthenticationScheme</key>
            <array>
            <array/>
            <key>DeliveryAccounts</key>
            <array>
                <string>5C5150FA-55CB-11D8-A746-000A957054BE</string>
            </array>
            <key>EmailAddressDomain</key>
            <string>@mac.com</string>
            <key>Protocol</key>
            <string>IMAP</string>
            <key>ServerName</key>
            <array>
                <string>mail.mac.com</string>
            </array>
```

```
            <key>UserNameMatchesEmailAddress</key>
            <true/>
        </dict>
        <dict>
            <key>AccountID</key>
            <string>5C5150FA-55CB-11D8-A746-000A957054BE</string>
            <key>AccountName</key>
            <string>.Mac SMTP</string>
            <key>AuthenticationScheme</key>
            <array>
                <string></string>
            </array>
            <key>Protocol</key>
            <string>SMTP</string>
            <key>ServerName</key>
            <string>smtp.mac.com</string>
        </dict>
    </array>
    <key>Identifier</key>
    <string>com.mac</string>
    <key>Version</key>
    <integer>1</integer>
</dict>
</plist>
```

Setting Up a Network Library

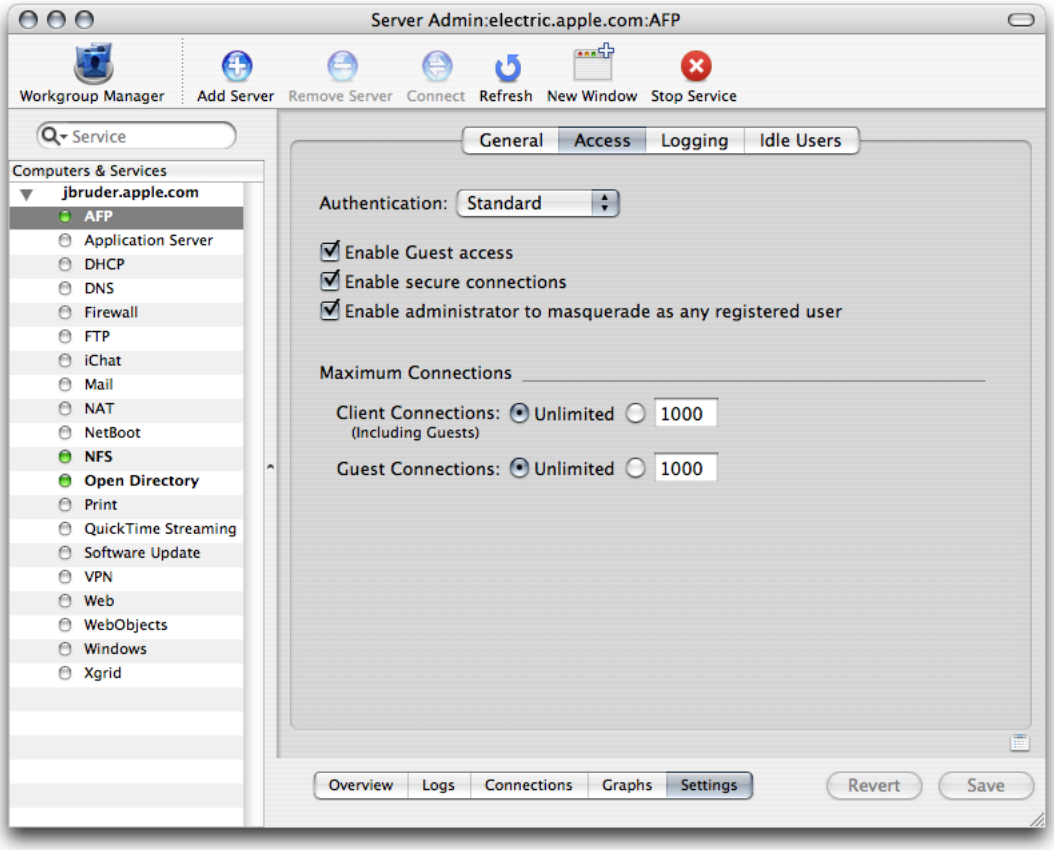
A network library allows for shared resources to be centrally managed and available to all Macintosh users bound to an Open Directory Master server. For example, a resource such as fonts, can be placed in a network library so they are available to end users bound to the Open Directory Master server. This allows users to use the shared resources with out having to install them. This article is a step-by-step guide to create a network library. For more information about the library structure see The Library Directory.

Creating a network library requires you to make some configuration changes to your Open Directory Master so it can provide clients bound to the Open Directory Master the information about the network library. The following steps walk you though the changes required to create a network library.

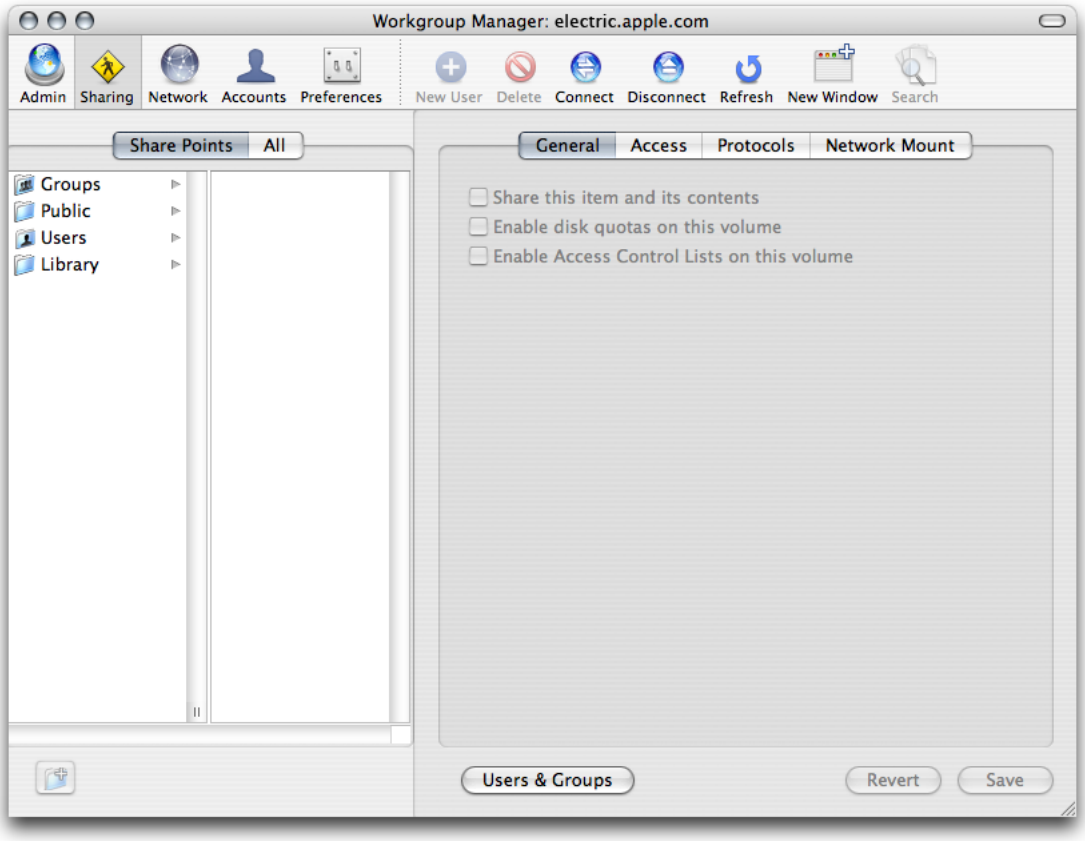
1. On the server where you want the network library to reside, create, in `/Shared Items`, a directory named `Library` . This directory serves as the share point for the network library.

   The newly created library directory is the location where you place any resource that should be shared to all of the clients utilizing the network library. The hierarchy of directories in the network library is the same as in `~/Library` or `/Library`. For example, if you needed to share a font with all network library users, you create a directory named `Fonts` in `/Shared Items/Library` . Then you place the fonts to share inside of `/Shared Items/Library/Fonts`.

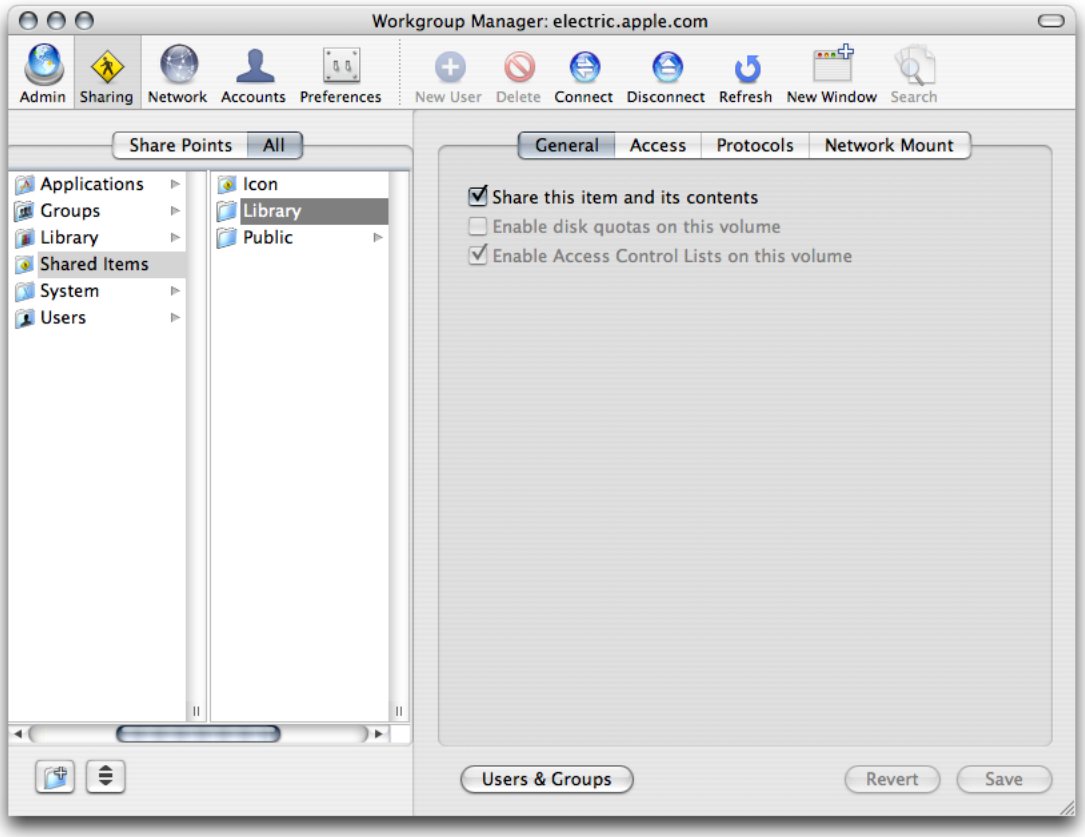
2. Using Server Admin, connect to the server and select AFP in the Computers & Services list. Make sure the AFP service is running. Click Settings, then click Access, and make sure “Enable Guest access” is selected.



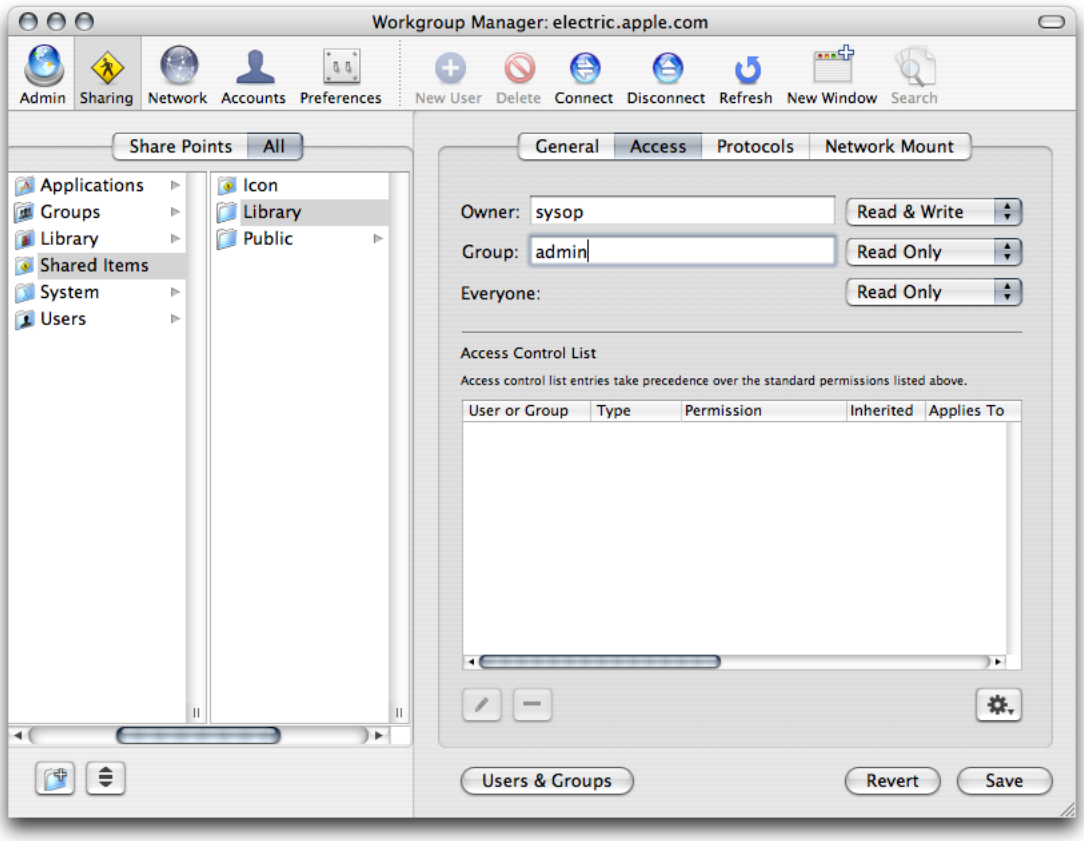
3. In Workgroup Manager, connect to the server and click Sharing.



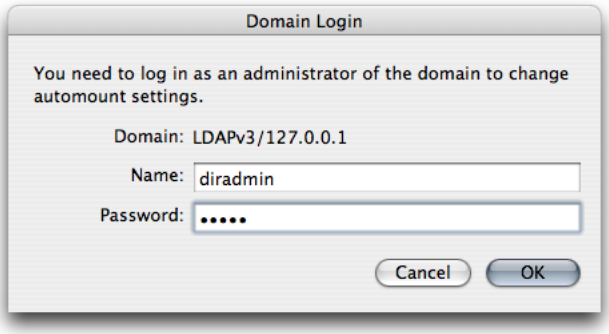
4. Click All and select the directory `/Shared Items/Library`. In the General pane, select the button next to "Share this item and its contents."



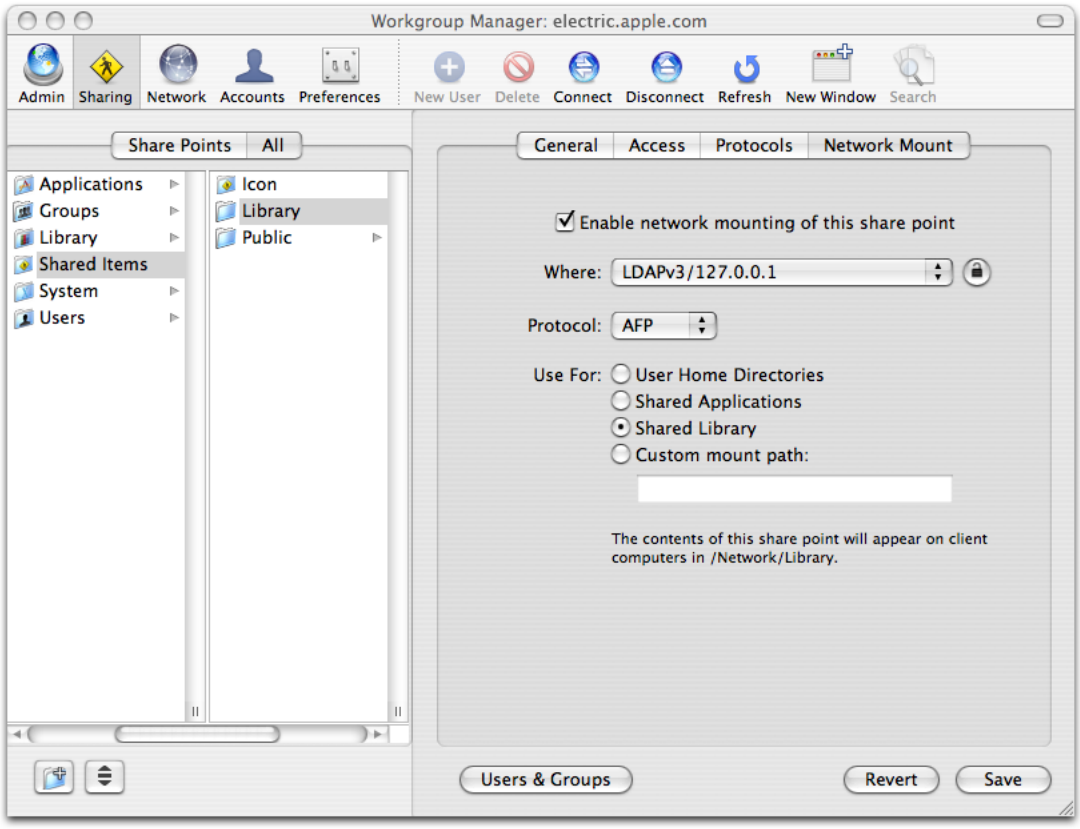
5. In the Access pane, specify the share point owner and group names by typing names into those fields or by dragging names from the drawer that opens when you click Users & Groups. Set Owner permissions to Read & Write, set Group permissions and Everyone permissions to Read Only. Click save.



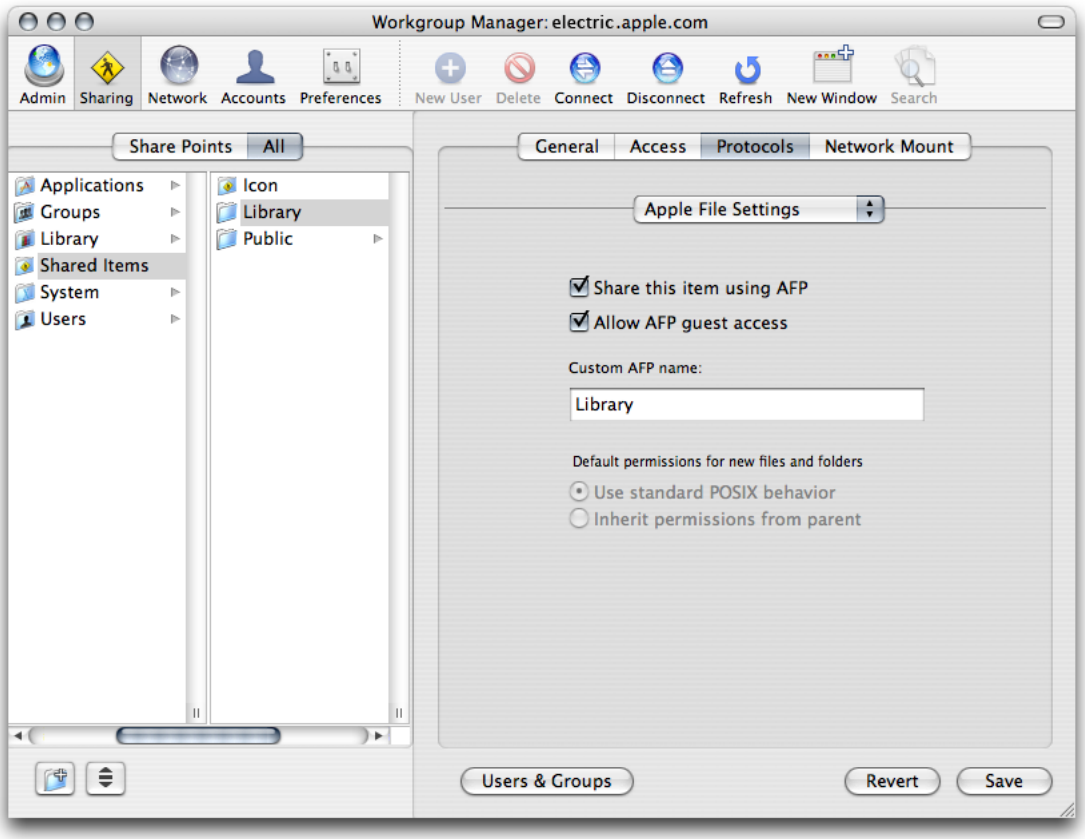
6. Click network Mount and authenticate as an administrator of the directory domain where the network library resides. If the server where the network library mount point is also the Open Directory Master for the network, continue to step 7. If the network library mount point is on a different server then the ODM, use the Where pop-up menu to choose the directory domain in which the network library mount record resides. Then click the lock and authenticate as an administrator of the directory domain.



7. Select “Enable network mounting of this share point” and “Use For Shared Library.” Click All (above the list on the left) and select the directory `/Shared Items/Library`.



8. Click Protocols, choose Apple File Settings from the pop-up menu, and make sure “Share this item using AFP” and “Allow AFP guest access” are selected. (They are selected by default.)



The network library is mounted when the client computers first turn on. Any client turned on before the configuration of the network Library is finished must be rebooted to gain access to the resources in the network Library.

Setting Up Key-Based SSH Login

SSH, or secure shell, is a protocol that allows users to securely log in, administer, or transfer files between remote computers. This article outlines the process of setting up key-based SSH login on Mac OS X and Mac OS X Server.

For more information about SSH see the OpenSSH homepage. Key-based authentication is helpful for tasks such as automating file transfers and backups, and using with fail-over scripts because it allows computers to communicate without a user having to type in a password.

Important: With the power of key-based authentication comes risk. If the private key you generate becomes compromised, unauthorized users will have access to your computers. You need to determine whether the advantages of key-based authentication are worth the risk.

Generating a Key Pair

To set up key-based SSH, you must generate the keys the two computers will use to establish and validate the identity of each other. To do this run, the following commands in Terminal:

1. Check to see whether a `.ssh` folder exists in your home directory by running the command `ls -ld ~/.ssh`. If `.ssh` is listed in the output, move to step 2. If `.ssh` is not listed in the output, run `mkdir ~/.ssh` and continue to step 2.

2. Run: `cd ~/.ssh`

3. Run: `ssh-keygen -b 1024 -t dsa -f id_dsa -P ''`

   This command generates the public and private keys. The `-b` flag sets the length of the keys to 1,024-bits, `-t` indicates to use the DSA hashing algorithm, `-f` sets the file name as id_dsa, and `-P ''` sets the private key password to be null. The null private key password allows for automated SSH connections.

4. Run: `touch authorized_keys2`

5. Run: `cat id_dsa.pub >> authorized_keys2`

6. Run: `chmod 400 id_dsa`

   The permissions on the private key must be set so that the file is not world readable.

7. Run: `scp authorized_keys2` *username@remotemachine*`:~/.ssh/`

   This command copies the public key and the authorized key lists to the specified user's home directory on the remote computer. If you need to establish two-way communication between servers, repeat the above process on the second computer. It is not secure to copy the private key of one computer to

another computer. This process must be repeated for each user that needs to be able to open a key-based SSH session. The root user is not excluded from this requirement. The home folder for the root user on Mac OS X Server is located at `/var/root/`.

Key-base SSH Use With Scripting

A cluster of servers is an ideal environment for using key-based SSH. The following Perl script is a trivial example that should not be implemented. It demonstrates connecting over a SSH tunnel to all of the servers defined in the variable `serverList`, running `softwareupdate`, installing any available updates, and restarting the computer if necessary. The script assumes that key-based SSH has been properly set up for the root user on all of the servers to be updated.

```
#!/usr/bin/perl
# \@ is the escape sequence for the "@" symbol.
my @serverList = ('root\@exampleserver1.example.com',
                  'root\@exampleserver2.example.com');
foreach $server (@serverList) {
open SBUFF, "ssh $server -x -o batchmode=yes 'softwareupdate -i -a' |";
while(<SBUFF>) {
  my $flag = 0;
  chop($_);
  #check for restart text in $_
  my $match = "Please restart immediately";
  $count = @{[$_ =~ /$match/g]};
  if($count > 0) {
 $flag = 1;
  }
}
close SBUFF;

if($flag == 1) {
 `ssh $server -x -o batchmode=yes shutdown -r now`
}

}
```

Incorporating User Images in Open Directory

Adding user images to Open Directory allows applications such as iChat, Mail, Address Book, and Login Window to display custom user images. Custom user images in Open Directory have many practical applications. This article will guide you though the process of creating images for use with Open Directory and through the process of adding these images into Open Directory.

One example of how custom user images in Open Directory add to the user experience is in a managed lab of computers in an elementary school. Younger children might not understand the process of logging into a computer with a user name and password, but they understand clicking an easily recognizable icon or a picture of their face to log in. Another example of how user photos in Open Directory add to the user experience is in a large Enterprise utilizing collaboration services such as email or Apple's iChat server. People using these services will see photos of the person they are communicating with via email or instant messaging.

Guidelines for User Images

User images must adhere to the following specifications to be compatible with Open Directory and Mail.

- The image must not exceed 64 pixels by 64 pixels in size. Ideally, the image should be exactly 64 pixels by 64 pixels because smaller images do not display correctly.
- The image must be a TIFF, and the file extension on the image must be `.tiff`.
- The name of the file must be the user's short name in Open Directory. For example, `jsmith.tiff` is a correctly named file.

Tools such as Automator or shell scripting can reduce the overhead associated with adding large numbers of user images to Open Directory. For more information on creating a set of Automator actions to automate the image creation and installation process see Mac 101 - Lesson 3: Automator.

Installing User Images

It is highly recommended that you copy the user image file to `/Network/Library/Images/People`. For more information on setting up a Network Library see "Setting Up a Network Library" (page 17). If the folders `Images` and `People` do not exist, create them. After copying the image file to `/Network/Library/Images/People`, create a symbolic link to the image using the naming scheme of *`shortname`*`@`*`emaildomain`*`.tiff`. This symbolic link is required by Mail to associate an image with the given email address. For example, a correctly named symbolic link has the form `jsmith@example.com.tiff`. This must be a symbolic link not an alias. For more information about creating simlinks, see *Mac OS X Man Pages*. If the user has multiple email addresses, multiple symbolic links must be created for each email address.

Adding Images to User Records

The following steps outline the changes that must be made to each user's Open Directory record to associate an image with a given user:

1. Using Workgroup Manager, connect to the Open Directory Master for your network.
2. In the preferences for Workgroup Manager, select "Show 'All Records' tab and inspector" and click OK.
3. From the user account list, select the user whose custom image you want to add.
4. Click the Inspector tab to bring up the raw data that makes up the user record in Open Directory. Be careful when editing the data presented in the Inspector tab; changing the wrong data can result in user records becoming invalid.
5. Click the New Attribute button.
6. Type `apple-user-picture` in as the attribute name.
7. In the Text field, type in the path to the user image. All paths begin with `/Network/Library/Images/People` and end with the filename you edited above. The path for the example used above is `/Network/Library/Images/People/jsmith.tiff`.
8. Click OK, then click Save.

Clients must be bound to the Open Directory Master for user images to appear. Any client turned on before adding user images to Open Directory must be rebooted for the user images to become effective.

Users are able to change their own images in the Accounts pane of System Preferences, and change others' images using Address Book. These changes take precedence over the images defined by Open Directory and will not affect the image on the server.

Creating Automator Actions for Apple Remote Desktop

Apple Remote Desktop (ARD) 3 is the latest release of Apple's desktop management software, enabling administrators to distribute software, manage assets, and remotely administer Mac OS X systems via a network. ARD 3 takes advantage of innovative technologies in Mac OS X Tiger such as Automator.

Automator is built on the concept of actions - discrete tasks such as opening a file, cropping an image, or sending a message. With Automator users can string together a series of actions into a single Automator workflow document. Much like a script, a workflow document can be executed, triggering each action and passing any data generated by the action to the next action in the workflow.

Using the Automator actions installed with ARD and custom actions you create, you can develop workflows to automate repetitive system administration tasks. For example, workflows can distribute software, create detailed software and hardware reports, and remotely configure systems using ARD.

This article is intended for system administrators and developers looking to create their own Automator actions to supplement the actions bundled with ARD. A basic understanding of ARD, Automator, Xcode, AppleScript and the process for creating Automator actions is assumed. For background information about Automator and creating Automator actions see the *Automator Programming Guide*. For more information about AppleScript see *Getting Started with AppleScript*.

This topic provides examples and sample code for three types of actions. The first is an action that performs a task provided by ARD. The second combines two actions into one workflow. The third example executes a UNIX or AppleScript command.

Example 1: Locking Remote Computer Screens

Apple Remote Desktop can lock the screen of remote computers, blanking their screens and preventing user input. The following sample code, written in AppleScript, allows you to build an Automator action that uses ARD to lock remote screens. In order to reduce the length of some of the lines in the examples a line continuation character (¬) is used. The use of line continuation characters is optional. To create a line continuation character in the AppleScript Script Editor, press Option-Return on your keyboard.

```
on run {these_computers, parameters}
    if the class of these_computers is not list then set these_computers to¬
    these_computers as list

    set task_name to "Lock All Screens"

    set screen_message to (|screenMessage| of parameters) as Unicode text

    tell application "Remote Desktop"
        set this_task to make new lock screen task with properties¬
        {name:task_name, showing output:false, message:screen_message}
        execute this_task on these_computers
    end tell
```

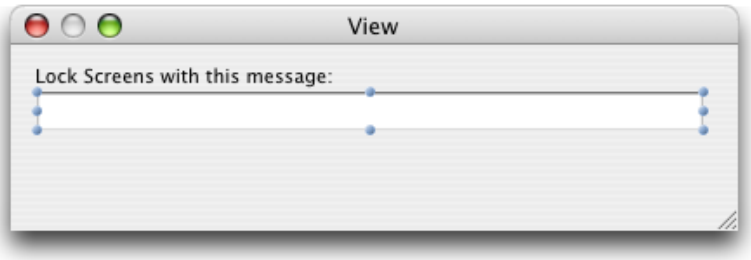
```
    return these_computers
end run
```

This example contains the `on run` hander, which is part of the standard template for all Automator actions and contains the statements that are evaluated when the script runs. It expects two things: a list of computers and a parameter containing the selections made in the action's user interface. For more information about extracting the stored parameters see Implementing an AppleScript Action.

When the `on run` handler executes, it first checks to make sure the `these_computers` input parameter is indeed a list. If it is not then it is made into a list.

The third statement creates a variable called `screen_message` that contains the text that will be displayed on the remote screen when it is locked. This text is extracted from the Automator action's user interface using parameters and is formatted as Unicode text.

For instance, you might design a UI for this action in Interface Builder like this, with the text field bound to a parameter called `screenMessage`:



Once the necessary variables are initialized, Apple Remote Desktop is used (by way of the tell application statement) to create a task called `this_task` as a lock screen task. The task is initialized with variables from the script and values from the script's input parameters and executed.

Finally the handler returns the list of computers it received in the on run handler. Though the script did not generate new data, it should return its original input so the list of computers it received can be passed to the next action in the workflow for additional processing.

Example 2: Combining ARD Tasks and UNIX Commands

This example combines two Apple Remote Desktop tasks (Copy Items and Send UNIX) into a single Automator action. When activated, it will change a remote computer's desktop picture to be the same as the ARD administrator's desktop picture.

The AppleScript in the example makes use of double quotes, which must be escaped to be used in a UNIX shell script. When double quotes are used in your AppleScript, type a backslash (\) character immediately before the double quote.

```
on run {these_computers, parameters}
    if the class of these_computers is not list then set¬
    these_computers to these_computers as list
    set first_task_name to "Copy Desktop Image"
    set second_task_name to "Replace Remote Desktop Image"
    tell application "Finder"
```

```
        set this_image to ((get the desktop picture) as alias)
        set the image_name to the name of this_image
    end tell
    tell application "Remote Desktop"
        set this_task to make new copy items task with properties¬
        {name:first_task_name, showing output:false,¬
        location:current users desktop folder, copy items:¬
        {this_image}, bandwidth limit:0, conflict¬
        resolution:replace, should open:false, ownership:current¬
        console user}
        execute this_task on these_computers
    end tell

    set the UNIX_script to "osascript -e 'tell application¬
    \"Finder\"
    set this_image to document file \"" & image_name & "\" ¬
    of the desktop
    set this_image to move this_image to (path to pictures folder)¬
    with replacing
    set the desktop picture to this_image
end tell'"

tell application "Remote Desktop"
    set this_task to make new send unix command task with¬
    properties {name:second_task_name, showing output:false,¬
    script:UNIX_script}
    execute this_task on these_computers
end tell
    return these_computers
end run
```

This example uses the same structure as the first example. In the script the Finder is asked to create an AppleScript alias to the current desktop picture, this alias is then used by the Remote Desktop copy items task. Finally, a UNIX script is used to set the image as the desktop picture. The UNIX command `osascript` executes an AppleScript as if it were a shell script.

Example 3: Combining AppleScript and UNIX Commands

Apple Remote Desktop allows administrators to execute a UNIX command or shell script on remote systems. The `osascript` command allows developers to embed AppleScript commands in a UNIX shell script. These two technologies allow administrators to perform a wide range of tasks that are not provided as standard features of ARD. The following example code shows how to combine "Send UNIX Command" and AppleScript functionality.

```
on run {these_computers, parameters}
    if the class of these_computers is not list then set these_computers to¬
    these_computers as list
    set task_name to "Clean Desktop"

    set the UNIX_script to "osascript -e 'tell application \"Finder\" to¬
    delete every item of the desktop whose class is not disk'"

    tell application "Remote Desktop"
        set this_task to make new send unix command task with properties¬
```

```
        {name:task_name, showing output:false, script:UNIX_script}
        execute this_task on these_computers
    end tell
    return these_computers
end run
```

This example is very similar to the first example, but uses a UNIX script in place of the lock screen task. In this case the AppleScript command will delete every item on the Desktop of remote systems, excluding mounted volumes.

Installation and Best Practices

Because every setup of Mac OS X Server is different, you should understand as much as possible about the process before deciding on a plan for implementation. This article discusses and recommends accepted best practices that should apply to many, but certainly not all, situations.

This article is intended as an overview of the best practices for system administrators responsible for large-scale deployments of Mac OS X Server. This article is not a step-by-step manual. Where appropriate or available, references are provided for more information on a particular topic. Also, special attention is paid to large-scale enterprise and institutional needs, because smaller-scale needs are already well covered by existing documentation.

Initial Installation of Mac OS X Server

Even though Server Setup Assistant provides a very robust and useful interface for setting up a server, a better methodology is required when deploying tens or hundreds of servers at any one time.

Many sites are using deployment techniques for Mac OS X Server that are very similar to Mac OS X client systems. Typically a "golden master" server image is created. This golden master sever image includes all relevant site-specific modifications, such as software installs and configuration, but has a dynamic DHCP-based IP address and no directory services configured.

The image is installed on the machines by hand imaging them with tools such as Apple's Disk Utility, which can do disk image restores. This process can easily be run from either an external bootable device, such as a FireWire hard drive, or a bootable DVD. The golden master image can either be stored on the bootable media, or hosted on a remote web or NFS server. For more information about installing images on computers, see *Mac OS X Server System Image and Software Update Administration*.

If you need to image more than a few systems at any one time, the use of a NetBoot or NetRestore server is common practice. Once booted from the NetBoot server, the systems are imaged with the golden master. Many times the imaging is actually accomplished by automated scripts so that no human intervention is required after the machine has NetBooted.

The new multicast Apple Software Restore (ASR) server that is present in Mac OS X Server v10.4 and Mac OS X v10.4 is being used for large scale system imaging. This tool allows the golden master image to be multicasted to as many machines as your network hardware can support at any one time. For more information about the ASR server see the ASR man page in the *Mac OS X Man Pages*.

Current deployments are imaging up to 2000 machines per day with a combination of NetBoot and multicast ASR with minimal administrative staff working on the task.

Regardless of whether you choose a full NetBoot or ASR setup or just use a firewire drive to restore an image onto the new server, you should not have to do full installs from the Mac OS X Server installation DVDs.

Configuration of Mac OS X Server

After the system has been installed, even if coming from the Mac OS X Server installation DVDs, you need to configure the system.

To facilitate mass deployments Apple has created a configuration file framework that allows administrators to pre-create setup files that a newly installed server will use to configure itself.

The Server Setup Assistant allows you to do this on systems installed from the installation media and to create text configuration files that can either be stored in the LDAP directory or connected to the machine by external storage.

Often you will be interested in more customized configurations, so it is common to use a golden master where the initial admin user and other settings have already been configured.

Note that when performing single machine installs through the Server Setup Assistant you should always configure the machine to be a Stand Alone system in regards to directory services. Log in to the machine first, and ensure that your settings for DNS and networking in general are working properly, before you changing the Open Directory configuration.

Also, if you are going through the Server Setup Assistant, you will be given the option to enable specific services that will be immediately available after the machine reboots for the last time during the configuration. With the exception of Apple Remote Desktop, used when the target machine is configured in a "headless" configuration without a video card, you should not enable any of the services.

Since you can enable individual services at a later time with no loss of functionality, it is best to ensure that your server has been correctly set up and then to configure each of the services before enabling them.

Drive Configuration

You should configure all servers with some form of RAID for the boot volume. If all of your data storage for the system is being handled by an external RAID device, considered best practice, then you can easily use the built-in software RAID of Mac OS X Server to mirror the boot drive.

If you need internal data storage, consider a hardware RAID card that will allow for RAID 5 and more configuration options than the software RAID alone will allow.

If you use an Xserve server computer with external storage, you want to fill the first two drive bays with smaller drive modules and then mirror the data for the boot volume. Then configure the third bay with a larger drive. Use it for admin storage, clones of the boot volume, and staging of backups.

Important: A redundant boot volume is a best practice.

While it is possible to install OS X on a UFS-formatted volume, it's highly recommend you use an HFS+ Journaled format.

It's common in many Unix server configurations to map the temp folder, virtual memory space, and `/var` directory to other partitions. This is typically done to either speed up the system or to prevent the boot drive from being filled up with either errant log files or ballooning virtual memory swap files. With the large amount

of performance tuning that has been done with OS X to maximize the caching of the RAM and the kernel itself in addition to the massive increase in hard disk capacity in recent years, there are rarely, if ever, any gains to be had by implementing the remappings. Additionally, it is more typically a machine's processor or network connection that is the bottleneck and not system drive I/O. In fact, you may be putting yourself at a disadvantage by doing this since very few admin tools or third party solutions for OS X expect this kind of configuration.

Cloning the Boot Volume

To account for system-update regressions and potential catastrophic damage to your boot volume, you should regularly clone the boot system onto other media.

In the case of the Xserve with external storage, the third drive bay is commonly configured with a large drive and then partitioned into three volumes. These volumes are used for a nightly backup, a known good backup, and a pre-update backup. Each of these is discussed in more detail below. The boot volume is then cloned to this drive using either `rsync`, `psync`, Apple System Restore, or a number of third-party utilities.

Every night the system clones the boot volume to the "Nightly" partition. This partition is used as a staging area for backups. It is also used as a primary restore volume in the case of a failure or corruption of the operating system. In these cases you can usually boot from the Nightly volume and have your server accessible to clients in a matter of minutes after the disaster. Then, when it is more convenient, you can either clone the Nightly volume back to the boot volume, or repair the boot volume so that it is useable again.

If you do clone your boot volume on a nightly basis, you can then run your backup software against the cloned system instead of your boot drive. Doing so lessens the performance impact of a backup on your server.

The boot volume is then cloned to a "Known Good" partition whenever the system has been running well for some time and has not had recent configuration changes. This volume is used less for restoration purposes and more for an online archive of configuration files.

Finally, the system should be cloned to a "Pre-Update" partition immediately before you apply software updates to the system. This provides a high degree of protection in case the update procedure produces unexpected results. In the same way that you booted from the Nightly volume, you can use the Pre-Update volume to restore a machine to functionality within minutes of detecting an abnormality with your system.

It's important to note that the cloning methodology described here is not a solution to backing up client data. It is preferred to have all dynamic client data be stored on an external RAID system and backed up to tape using any of a variety of backup solutions. The use of the boot volume clones is entirely to ensure the highest availability possible for your system volume, not your client data.

Performing Ongoing Maintenance

Software updates to production systems should be undertaken only after sufficient regression testing. Although the Nightly volume will help in this matter, with large installations it is common to use a network-based file management solution.

Apple's Apple Remote Desktop (ARD) allows an administrator to push software changes to hundreds of machines at the same time, helping you to ensure that they are all maintained in a similar fashion. With the introduction of ARD 3.0, machines that are offline during a software or configuration update are updated or configured when they come online..

If you desire a more automatic solution for this you can look into the open source Radmind, popular in a large number of higher education installations, or commercial options such as FileWave, Casper, and Net Octopus. All of these solutions allow you to do checksums on every file on your server and automatically replace any file that has been modified without permission.

Although these solutions usually have a fairly steep learning curve and infrastructure requirements, using them virtually guarantees that all of your systems are identical in configuration and prevents configuration decay over time.

Document Revision History

This table describes the changes to *Mac OS X Server Administrator Topics*.

Date	Notes
2007-05-23	Updated for Mac OS X v10.5.
2006-06-28	Corrected typos. Added a screenshot to the article "Creating Automator Actions for Apple Remote Desktop."
2006-05-23	Added new article, "Installation and Best Practices."
2006-04-14	Added information about creating Automator actions for Apple Remote Desktop.
2006-02-07	Clarified instructions for creating a network library.
2005-11-09	New document that describes a variety of server administrator advanced operations.