Mac OS X Enterprise Application Management

Best Practices

macosxlabs.org

Richard Glaser, University of Utah

Philip Rinehart, Yale University

June 21, 2004

Contents

1 Abstract

1.1 Preface

This white paper will attempt to outline best practices in the installation, deployment and distribution of Mac OS X software in an enterprise environment. While some topics are specific to an enterprise Information Technology department, small offices and developers can use most topics. Additionally, we will provide information for enterprise Information Technology administrators on tools used to troubleshoot current challenges, and techniques to provide solutions for these challenges. With an understanding of the following topics, developers can gain insights on how to create enterprise friendly software applications.

Richard Glaser is a member of the steering committee that leads the Mac OS X Lab Deployment Project, www.macosxlabs.org. This initiative is intended to provide a collaborative environment for lab administrators to support one another in the deployment of Mac OS X in a lab environment. It is composed of roughly thirty distinguished higher education institutions from around the world. Richard administrates the web, file, email and discussion forums servers and has contributed much of the documentation and content on the project web site. He has presented at MacWorld 2004 on Radmind and numerous other topics for the macosxlabs project. Currently, he is the system administrator for Student Computing Labs that manages 350 Mac's used by students, staff and faculty and provides support and training for campus managers and users.

Philip Rinehart is a member of the steering committee leading the Mac OS X Lab Deployment Project (www.macosxlabs.org) and manages Macs as a support specialist at Yale University. He has been using Macintosh Computers since the days of the Macintosh SE, and Mac OS X since its Developer Preview Release. He presented on Macintosh OS X security at MacWorld 2004, and is an active member of the Mac OS X Lab project. He has contributed to many areas of the project. Before coming to Yale, he worked as a Unix system administrator for a dot-com company.

The authors would like to acknowledge the contributions and suggestions from Doug Willen, Michelle Clifford, Justin Elliott, Debbie Lords, Deborah Cherry, Greg Neagle, and Lance Ogletree.

1.2 Audience

Managing the operating systems and software installed on hundreds of diverse computers across a physically large campus or site creates significant challenges for enterprise information technology managers and staff. The migration from Mac OS 9 to Mac OS X has created some particular new challenges for information technology management and staff in the enterprise environment as much as it has for software developers. Many of these issues can be attributed to the differences between the Mac OS 9 and Mac OS X security models, operating system architecture, and distribution and management tools.

In enterprise Information Technology environments, both large and small, resources are limited and enterprise administrators make software purchasing decisions based on the constraints imposed by these limitations. Software that presents significant hurdles, which does not follow best practices, and causes inefficient use of limited Information Technology resources is less likely to be purchased and deployed. Even when deployed, inconveniences in the maintenance of particular software may lead Information Technology staff to recommend alternative choices when users request either guidance or purchase recommendations.

This white paper is intended for administrators or developers who deploy and develop software in an enterprise environment. The environment can range from a small office to a large enterprise with thousands of Macintosh computers. Also, it is intended for developers that create software for Mac OS X to expose them to the challenges and the tools used to manage and distribute software on multiple machines. In turn, a better understanding of challenges in distributing applications will allow development of enterprise-friendly applications. Lastly, this white paper is intended for Apple Computer, as input on future directions of Mac OS X for enterprise application management.

1.3 Factors in the enterprise

- Licensing

  This section defines and describes software licensing, methods used to license software by developers, and preferred methods for those managing and distributing the software.

- Security

  This section presents the Mac OS X security model, including notable challenges that arise when distributing software such as permissions, special modes and ports

- Installation

  This section covers the challenges of installing and initial setup of software including reviewing, tracking and troubleshooting installers. Additionally, examples of issues are presented.

- Missing Features and Support

  This section covers important Mac OS X software support issues and Mac OS X software that is needed for widespread acceptance and adoption in the enterprise.

- Distribution

  This section presents tools used to manage and distribute software. Additionally, different implementations and their impacts are covered.

- Troubleshooting

  Tools and procedures used to detect multi-user software issues is covered.

- Solutions

  Solutions to multi-user software issues are covered.

1.4 Comments and Questions

Please send comments and questions about this white paper to the Mac OS X Labs project's Contact Us web page. Errata, examples and up-to-date information can be found at Application Distribution Best Practices

2 Serialization of applications

The first area at issue is serialization. What exactly is meant by the term "serialization"? From a software developer standpoint, serialization is the incorporation of licensing that prevents the serialized software from being illegally used on unauthorized computers. In an enterprise environment, there are many ways to address this issue. Some methods have little or no impact and others create unacceptable use of enterprise resources.

Software can be sold in either a "concurrent" or individual licensing model. Concurrent licensing allows simultaneous, i.e. "concurrent", use of software by one or more users, up to the maximum number of users allowed by the license agreement. With concurrent licensing, software may be installed on any number of computers within the enterprise. Frequently, it will exceed the number of licenses available for concurrent use at any instant in time.

Individual, machine specific, licenses are the second way software is sold. This type of licensing limits the distribution of software and in turn limits user access to computers. Also, it is usually more cost prohibitive than concurrent licenses. Additional requirements are often required, such as a node locking license to machine specific hardware. At times, a software developer will sell a supposedly "concurrent" license, or a group of single licenses. However, if the license requires individual entry by Information Technology staff onto each machine onto which it is installed, it is not truly a "concurrent" license as defined above.

An additional concern for Information Technology administrators is piracy of software that is deployed in public environments. As a suggestion, developers should remove application serial numbers from any informational dialog box such as splash screens and

"About" dialog windows. Storage of serial numbers inside plist or preference files stored in user's home directory also allows piracy.

2.0.1 Network Detection

Network detection licensing allows simultaneous use of software by one or more users, up to the maximum number of users allowed by the license. With network detection licensing, software may be installed on any number of computers.

As an example, OmniGroup's software can be licensed in this way. Users purchase a number of licenses. The software is responsible for monitoring and managing the number of licenses in use. This method does not add to the burden of installation, as each software installation is self managing. Additionally, no additional hardware is required.

However, monitoring of license usage should be as network friendly as possible In an enterprise environment, polling for the number of licenses in use should be not more than every 15 minutes.

Concurrent licensing software also should not listen on a network port not defined in `/etc/services`. If a specialized port is desired, it should be registered with IANA.

All ports for concurrent licensing should be documented. In an enterprise environment, firewalls and routers may block the ability of software to perform its licensing verification without this information. If RPC services are required, "use secure RPC for the running services where possible."[1]

2.0.2 Server metering

The first method of serialization can use a server product to meter the software licensing agreement. This model has many methods of implementation. It is often implemented with a concurrent licensing agreement, but it can be used individually or in addition to standard licensing for piracy prevention. When used with a concurrent licensing agreement, the license server is responsible for metering the concurrent licensing agreement.

As an added benefit for enterprise IT, server metering allows easy and granular management of distributed software, reducing barriers for deployment that exist with other serialization methods. Server license management allows user or location specific con-

trol, usage monitoring, anti-piracy, vendor-defined policy, auditing, and detailed version management. These features make server license management a very advantageous tool. Server management software includes products such as Sassafras software KeyServer product and Rainbow Sentinel Technology FlexLM product. Both products allow metering and monitoring of purchased licenses of software. It is important in an enterprise environment that software is compatible with and properly works with server license management software.

As an example of a license server friendly application, Peak 4 now allows the application to be directly controlled by KeyServer for proper metering and use of the software package. The program can also be patched with KeyServer. This ability allows Information Technology staff to add code to the application that forces it to be used only within the parameters of control set by the server monitoring application. The addition of this ability now allows the distribution of Peak 4 to a wider range of machines, increasing revenues for the software developer, and reducing the burden to an enterprise customer. Peak 3 did not follow this model, requiring machine specific information for proper licensing. The net effect of this barrier reduced the total number of installed licenses.

An example of an application that works with FlexLM licensing metering software is Mathematica by Wolfram software. Use of the FlexLM server package allows for proper metering of Mathematica across the enterprise.

However, FlexLM can also be administered in non-enterprise friendly ways. As an example, ArcView's implementation of FlexLM requires a hardware lock for the server. This method can be problematic when the license manager hardware would otherwise serve more than one function. The use of a hardware key may therefore require the use of additional Information Technology resources. ArcView's license manager also compounded the problem, requiring a specific operating system for the FlexLM, limiting options for Information Technology deploying this application. Frequent updates to the FlexLM license server software, potentially creates problems, often not working as documented. Lastly, FlexLM does not have the reporting capabilities of KeyServer.

Some software developers have created their own proprietary license servers. This method is the least preferred, as often the license server is not robust and it requires the

administrator to dedicate extra time for support of an additional license server, or obtain additional hardware for an additional license server. As an example, Sibelius has created a network license server. In its first release, it was not robust, consuming all available memory on the license server. The second version has improved, but is still not as robust as widespread deployed license servers mentioned previously. Use of proprietary license servers imposes an additional burden on enterprise Information Technology, requiring additional server resources, and administrative management beyond existing KeyServer or FlexLM systems.

2.1 Hardware serialization

Hardware serialization, sometimes referred to as "system fingerprinting", is the process of locking an application to a hardware component, either using a stand alone key or an aspect of the computer system on which it is running preventing unlicensed use. A software application can use any part of a computer system's hardware, from the CPU to a monitor. This method of licensing requires that the a code be created manually on each machine onto which it is installed. Each time that a machine must be touched by Information Technology staff adds to the cost of software deployment. Also, in the future if specific computer or components are replaced or upgraded, the license may need re-installation. Deployment of software using this method may become untenable when its difficulties are multiplied across hundreds or thousands of computers deployed in numerous buildings throughout an institution. The result may well result in a decision to not purchase the software.

2.1.1 Hardware Locks

Hardware locks are a piece of hardware attached to specific user machines which have internal set code identifying an application that has a legal license. They are often referred to as "dongles". Hardware locks may give some software developers a false impression of increased security and resistance to piracy. In an enterprise environment, they are one of the least preferred methods of licensing due to their difficulty and inflexibility to administer. Server license management solutions offer a high level of resistance to piracy;

developers can offer predefined licenses that cannot be modified by the administrator and avoid the inflexibility of locks in an enterprise environment. Listed below are some of the issues with locks:

1. Limits Distribution - Lack of flexibility in software distribution in the enterprise, limits its availability of software to specific computers or groups of computers. In a highly utilized environment, this adds severely to the cost of deployment

2. Theft Risk - Locks must be attached to publicly accessible machines exposing them to risk of theft, which then can be used to pirate software

3. Securing locks from theft is quite difficult, as most are quite small

4. Expense - Initial cost and replacement of locks is expensive when compared to other licensing methods

5. Support - Hardware locks require more individual user support, as many are not familiar with this technology and error messages are often unclear. In public areas, users may unplug a lock without understanding impact on future users who do not know that it has been unplugged

6. Ports - Shifts in hardware locks technology, i.e from ADB to USB, can be problematic as hardware is replaced. Since most locks do not allow for pass through use, there may be insufficient ports to accommodate all peripherals and locks, necessitating additional equipment and the need to secure that equipment from theft to supply the needed ports

7. Single point of failure - If the lock fails, there is no alternative for using the software working except a time consuming exchange with the vendor

8. Physical limitations - Smart hardware locks, such as the iLok or eMagic proprietary hardware locks must be authorized for either new or additional versions. It requires an administrator to either unsecure the locks and authorize them from a centrally located machine, or requires authorizing the lock on each authorized machine

9. Administrative overhead - Separate authorizations for each smart lock requires record keeping, physical identification of each lock, and the serial numbers, adding significant administrative overhead to maintaining the application's usability

Notable software developers that use this method include Steinberg Cubase , LightWave, and QuarkXPress. WINNMR is no longer used at some institutions due to difficulties related to its reliance on hardware locks.

2.1.2 Media verification

Another method of hardware serialization uses additional media to verify the validity of the installation. Again, problems arise in an enterprise environment as each machine must be touched.

As an example of an application that uses media verification, Final Cut Pro 3 requires media verification information for proper licensing. With the Final Cut Pro 4 volume license agreement, the application can now be directly controlled by KeyServer for proper metering and use of the software package. The addition of this ability now allows the distribution of Final Cut Pro 4 to a wider range of machines, increasing revenues for the software developer, and reducing the burden to an enterprise customer. Additional software that uses this method includes Reason.

2.1.3 Challenge/Response

The last, and least preferred method of hardware serialization is challenge/response license verification. Why is this the least preferred method? In a large enterprise environment, each machine must be touched after software installation. In comparison to the previously discussed serialization method, hardware locks; challenge/response licensing cannot be automated in an unattended manner. Hardware locks can, in at least a limited distribution.

The challenge/response license method is used in two different ways:

1. The administrator on the machine runs the application or a helper application which generates a challenge based on hardware characteristics of the authorized machine.

This challenge is sent to the vendor by electronic means. The vendor then generates a response that matches the challenge code and sends it to the administrator. Often, a delay of one business day or more is required for processing. An example application that follows this procedure is Bryce.

2. The administrator enters a serial number and key which is then verified by 'phoning home' to a server at the company. If the machine has no network connectivity the machine cannot be authorized to use the software. Practica Musica and Graphing Calculator are examples of software that use this method.

These methods of administering software create an additional difficulty, record-keeping. Most challenge/response solutions are designed for a single-user machine. In an enterprise environment, a single administrator is required to complete the entire challenge/response transaction. This requirement becomes especially burdensome in a managed environment as if the specific administrator is not available the machine cannot be authorized. Unlike hardware locks, each machine using challenge/response must be touched again to install the verified response. Additionally, if any hardware is replaced on the machine, the challenge/response key may be invalidated. Over the course of a semester, a single application could require multiple verifications. In a small Information Technology environment, where staff resources are limited, this method becomes impractical for widespread deployment.

3 Security

With the rise of malicious code, virii, and worms targeting known vulnerabilities, security, both in regards to the network and platforms on it, is a more significant concern for all enterprise deployments. The resultant downtime, the expense incurred by this downtime, and increased concern arising from governance and regulatory compliance, e.g. HIPAA, Sarbanes-Oxley, FERPA, requires the enterprise to focus on computer security. In addition, with Mac OS X, inroads have been made in previously unavailable environments, such as the federal system and military, where security of client operating system is of critical importance. While Mac OS 9 was a "closed" operating system with little or no access to

internal code and a command line, Mac OS X is an open source operating system based on Unix. Access to a wide variety of open source software has increased the ability to compromise a system that was not available in Mac OS 9. Lastly, the change in the security model from an originally developed single user environment to a newly developed multi-user environment creates security considerations with enterprise software installations and distribution.

3.1 Permissions and File Modes

In Mac OS X each file system object, a file or folder, is owned by a user and is associated with a group. For each object, specific permissions can be granted to the owner, group and to all other users. There are three basic permissions: read, write, and execute. Some installers or applications set these basic as well as other special permissions incorrectly or insecurely. These incorrect permissions are especially notable in a multi-user enterprise environment in which untrusted users must be allowed access to the machines, such as in publicly accessible computing facilities, libraries or kiosks in public spaces.

3.1.1 SUID

One of the main areas of file system security on Mac OS X systems is Set-UID (or SUID) and Set-GID (or SGID) programs. Normally, an application runs with permissions of the logged-in user. Only files and directories that have the correct user permissions are affected. However, in certain situations an application may require group or root user privileges. Improperly configured installers and applications may incorrectly use SGID or SUID. If permissions of the application are set with the SUID or SGID bit, the application runs with either the permissions of the application owner (SUID) or the permissions of the application group (SGID). Frequently, SUID application are owned by 'root', and will run as the root user, heightening security concerns since the root user has complete access to the operating system and file structures. This security concern is amplified when installers use these file modes.

Why are SUID or SGID applications or installers undesirable? If the application has a programming error, buffer overflow, or other security issue, it may be exploited by a

hacker who can use these elevated permissions to affect secure areas of the system. Additionally, files and folders may be overwritten or modified unintentionally. In an enterprise environment, any SUID or SGID application requires reevaluation due to security concerns.

A few of the command line utilities that have the SUID bit set with an owner of root include:

```
/sbin/rdump
/sbin/restore
/sbin/route
/sbin/rrestore
/usr/bin/atq
/usr/bin/atrm
/usr/bin/batch
/usr/bin/chfn
/usr/bin/chpass
/usr/bin/chsh
/usr/bin/rlogin
/usr/bin/rsh
/usr/bin/su
```

The binaries listed above can have their SUID bits removed, without undue harm. As stated in the Unix Security Checklist from the Australian Computer Emergency Response Team, " Remove this bit from programs that do not require elevated privileges to function successfully."[1].

An example of a software application that improperly implemented SUID is Virtual PC. Virtual PC is a popular x86 virtual machine emulator capable of running several Windows operating systems under Mac OS X. Virtual PC provides a set of services for managing network sharing capabilities under Mac OS X. These services are implemented using a SUID file mode with root permissions, with the component program VirtualPC_Services, located in the application package. VirtualPC_Services creates a log file upon startup in /tmp/VPCServices_Log. However, it does not check for several unsafe conditions prior

to creation of these temporary files. As a result, any user could create a symbolic link that would cause the contents of another file to be overwritten or a new file to be created.

N.B. An update for this vulnerability is available from Microsoft. This example illustrates the security concerns for Information Technology when developers use SUID or SGID file modes.

3.1.2 World writable directories

World writable directories present a large security risk to Mac OS X. If an application or installer modifies files or folders that grant write privileges for everyone, known as world rights, outside of subfolders in the user's file space `/User/<username>`, files or directories can be replaced or modified by a potential attacker and used to compromise the system. Also, applications that save user files as world writable could allow another user to inadvertently delete those files.

A developer may need world write privileges to allow an application to create a temporary file or directory, usually in same directory as the application. If temporary space is needed for program operation, it is best to use `/tmp`. Temporary files can be written in this directory by all users, but cannot be affected by other users.

Why is this directory acceptable? On Mac OS X, as on all *nix systems, a special permission, file mode, called the "sticky" bit can be set on a file or folder. If this is set for a folder, an unprivileged user may not delete or rename files owned in that folder by other users. For example, this bit is set for the `/tmp` directory.

N.B. Files saved in the /tmp directory are removed on startup. Developers may want to implement end-user options for backup or recovery of files. These files could be saved in the user's file space, in case the Mac is rebooted. Ideally, it is an end-user option and not a requirement of the software application. Environments implementing network home directories have limited disk space for user files, and may not be able to use a user's home directory for temporary file storage.

Other reasons for incorrectly using world rights is unfamiliarity with the Mac OS X security architecture or installers which incorrectly set permissions. As an example, Apple's Installer application can set permissions of a pre-existing directory to the version in

the archive overwriting the previous directory permissions. Apple advises, "This suggests you should select the Overwrite Permissions checkbox only if you are certain that you know better than the user what the directory permissions should be." [2] For more information, see Apple Software Distribution Concepts. This setting in PackageMaker should be used *very carefully*.

3.2 Unnecessary open ports

If a specialized port is desired, it should be registered with IANA. Any open port may be blocked by firewalls or routers in an enterprise Information Technology environment. Additionally, an open port may be used as an attack vector by an intruder.

3.3 Startup items and root daemons

Use of either Startup items or root daemons create a security concern for enterprise Information Technology. Many times this process creates and modifies file or folders insecurely, or overwrites existing data. Additional resource requirements are used to test and evaluate code which implements either startup items or root daemons.

As an example, the LoginWindow process in Mac OS X is launched at startup. The process creates the folder `/Library/Logs/Console` with world writable permissions. It then creates a second directory `/Library/Logs/Console/<username>` owned by that user. On a publicly accessed, multi-user machine, enterprise administrators do not want to allow writing to this directory space.

3.4 Tripwire

Many enterprises check file system integrity on an installed system, often referred to as tripwire. These programs are used to monitor and manage the computer's file system. Tripwire establishes a baseline 'snapshot' of the file system, recording file system properties such as owner, permissions, modify time, and content hashes. This information is stored in a secured database. When an integrity check is run, it gathers the same information on the monitored file system and looks for any differences. Any deviations are written

to a report file and based on a policy. Based on the results, a system is checked for signs of compromise.

Software should be developed that functions in a secured tripwire enviornment. If the application writes or modifies the file system areas either monitored or managed by tripwire, it will create false positive intrusion readings creating more work for the enterprise administrator.

As an example of an application that creates files in non-standard locations is Dreamweaver MX 2004, build 7.0.1.2181. On launch, the Macrovision root SUID tool located at `/Library/Application Support/Macrovision/AuthenticationService` creates files in

```
/cores
/private/var/root/
.Trashes
/private/var/db/netinfo
```

Additionally, these files are generated randomly, complicating the management of any managed Mac OS X client system. An example transcript follows:

```
/.Trashes/.4C435602-8295-11D8-B17A-003065908C38
/.Trashes/.4C4DE190-8295-11D8-B17-66DA-65908C38
/.Trashes/.4C4E8FF1-8295-11D8-B174C4DB365908C38
/.Trashes/.AnHymoexdWtcPK3Mi12xzDL9aRPRItInformation TechnologyVIvu
/cores/.4C459B7A-8295-11D8-B174C44B365908C38
/cores/.tuUSiPaDqcNMMAHyQH9p6H8-111EExl0SKUI
/private/var/.Macrovision
/private/var/.Macrovision/.MVBackup
/private/var/.Macrovision/.MVBackup/0.dat
/private/var/.Macrovision/.MVBackup/256.dat
/private/var/.Macrovision/.MVBackup/257.dat
/private/var/.Macrovision/.MVBackup/258.dat
/private/var/.Macrovision/.MVBackup/2986409984.dat
/private/var/backups/.14:57:54\b-\b29.03.2004\b-Vj4ufJ6W.conf
```

```
/private/var/backups/.14:57:54\b-\b29.03.2004\b-zcfMtO4v.conf
/private/var/backups/.14:57:54\b-\b29.03.2004\b4C6A1CLzx.conf
/private/var/cron/.14:57:54\b-\b29.03.2004\b-LR8leKxR.conf
/private/var/cron/.14:57:54\b-\b29.03.2004\b-Wv8FTNUp.conf
/private/var/db/.4C44F695-8295-11D8-B17-66FE-65908C38
/private/var/db/.7wtpXllukAC9J9h01z4eJM1LGnAq2sPqpKaJ
/private/var/db/netinfo/.14:57:54\b-\b29.03.2004\b58-821M9S.conf
/private/var/db/netinfo/local.nidb/.zv6zFMEqM8hANlyX5O6B9YYnXt9k41b4ZAwY
```

The creation of these files and folders in tripwire managed file space may not be allowed in an enterprise environment.

4 Installation

Based on the above security considerations, the creation of well behaved and Information Technology friendly installers becomes even more critical for the enterprise administrator. Smaller departments may not have the resources to properly troubleshoot installers. Certain practices should be followed to ease and simplify installation and, in turn, distribution of software. These practices include:

1. Log Files - Installers should create detailed log files including file permissions, directory permissions and attributes.

2. Admin Permissions - Installers should avoid requiring administrative or root permissions to perform installation.

3. Install Location - Install only in standard locations on the file system as defined by Apple and noted in this paper.

4. Preference Scopes - Software should support preference scopes which allow application preferences to properly work when stored in user, host, or network locations. This support allows software to be flexible and implemented in different environments.

4.1 Log Files

Application installers should, if possible, create detailed log files. Log files are useful to determine permissions, attributes, installation locations, modifications, and support files. Logs should be created in user file space, ~/Library/Logs. Installers should not create logs at the root level of the file system.

One tool for creating log files is Apple's Package Maker. A log file is created in ~/Library/Receipts,fulfilling some of the requirements listed above. Using the command line tool lsbom, an administrator can examine the contents of the package. A binary file, i.e. package.bom, is located inside the pkg archive. The lsbom tool interprets the content, and produces output formatted such that each line contains the path of the entry, its mode (octal), and its UID/GID. The tool does not list possible modifications made by installer scripts. Additionally, it does not support third party installers such as Installer VISE and VISE X or InstallAnywhere. Other Mac OS X unix package management systems do not support bill of material files, requiring use of another installation tracking method.

4.2 Administrative or root rights

If an application does not install in protected system areas, the use of administrative or root user rights is unnecessary. The unnecessary use of this requirement may prevent installation of software by a non-administrator. In many environments non-administrative users are allowed to install applications in their user file space. As a second consequence, this unnecessary requirement can lead to careless use of administrative credentials. This issue gives opportunities for hackers to create installers used to compromise systems.

4.3 Non-standard location

Many installers install files in non-standard locations. Installers that do not follow recommended guidelines on installation create an increased burden for enterprise Information Technology staff when tracking and troubleshooting installations. Apple specifies, 'Typically, most applications in a Mac OS X system are installed in /Applications to make them

available to all users of a particular computer system." [3] For more information, see Mac OS Installation and Integration.

4.4 Broken links

Software applications can incorporate Finder aliases or symbolic links. Symbolic links use Unix file paths, while aliases use id numbers. Either moving or renaming an application or its parent folder can invalidate an incorrectly formed symbolic link. In turn, an application may not function properly if it depends on the symbolic link. Software should not use symbolic links with absolute paths as the application may be moved or renamed. In versions of Mac OS X before 10.2, aliases located a file or folder using its unique identity first and its pathname second. Beginning with Mac OS X 10.2, aliases reversed the search order, using the pathname first and unique identity second. If a file is moved and replaced with an identically named file, aliases to the original file now point to the new file. Similarly, if you move a file on the same volume, without replacing it, aliases use the unique identify information to locate the file. This behavior mimics Unix symbolic links.

As an example, with an application `MyApp`, if the following symbolic link is created: `MyApp -> /Applications/MyAppFolder/MyApp`, the software application can never be moved from its original location. In this situation a relative symbolic link would not work either, i.e. `MyApp -> ../MyApp`, as if the application is renamed the link is no longer valid.

Symbolic links rely exclusively on path information to locate a file. If you move a file somewhere on the same volume without replacing it, symbolic links to the file break while aliases do not. The only way to fix a symbolic link is to delete it and create a new one. [4]

Aliases can also be broken. As an example, if the following alias is created: MyApp ->/Applications/MyFolder/MyApp, and the software is moved or renamed, the alias can break during a cloned hard drive operation. This behavior is caused by the alias using the file inode, which is different on a cloned system.

For more information, see Symbolic Links and Aliases

4.5 Path limitations

Paths in any installer have the potential to create havoc on a Mac OS X system. Mac OS X has the unique ability to have files and directories with spaces in the name. If the installer does not support spaces in the name, files can be deleted and named improperly.

4.5.1 File name limitation

File names for an application should be able to handle any special character or modification without breaking installers. File names should not have any limitations greater than the defined by Mac OS X. In an enterprise environment, Information Technology administrators often change applications names for ease-of-use. As an example, "Mac OS X" is often removed from an application name.

4.5.2 Directory location limitations

Applications should not require installation at the root level of the hard drive. Additionally, applications should not limit the placement of applications to certain directories. Installers should allow directory names that contain application files to be renamed, as well as support of special characters including spaces

For example, Painter 7 installs the application at the root level of the hard drive. When attempting to upgrade the application, Painter 7 must still be installed at the top level of the hard drive. It cannot be upgraded if it is not. In an enterprise environment, installation directories cannot be limited in this way.

5 Missing Support and Features

One of the decisions made at many institutions deploying Mac OS X is the availability of specialized Mac OS X software. If this software is not available or does not satisfy enterprise security policies, alternative software may be evaluated. Many of these applications have open source alternatives. These open source applications can replace their commercial counterparts, but often may not be as easily used. With the X11 distribution supplied

by Apple as part of a Mac OS X installation, applications can also be run on remote platforms which support the missing software. This solution is not ideal, as it may not always be feasible due to a variety of conditions. As a result, revenue may be lost by the software developer.

5.1 Legacy application support

Many enterprise users bring documents that have been created with non-Mac OS X and legacy applications. It is vital that Mac OS X updated applications open files from prior versions of the software. Additionally, it is equally important that applications have the ability to open foreign application files.

As an example of legacy application support, Microsoft Word cannot properly open WordPerfect files. In an enterprise environment, this inability requires additional utilities such as MacLinkPlus. A second example is QuarkXPress. QuarkXPress can open older files which have been saved in the Classic version of the program, but once saved from Quark Xpress 6, previous versions cannot open the older file easily. The cost of deployment is magnified, as all distributions of QuarkXPress have to be upgraded to avoid this conflict. Additional training of users, and higher replacement costs is also required.

As a recommendation for interoperability in the enterprise, software applications should support standard file formats. Software applications should maintain backwards compatibility as long as it is feasible.

5.2 Human Resources software

In many universities, PeopleSoft is deployed for the management of human resources. The poor application support by this vendor can make the deployment of Macintosh computers in departments unfeasible. Many institutions have consumed additional resources and created solutions for limited support not supplied by the vendor for Mac OS X functionality. Some of these solutions include the use of Virtual PC, terminal services, or the use of a Citrix client. Small departments in Universities do not have the contacts or resources to aid in the fixing of many of the problems with applications.

5.3 Web-based instructional software

WebCT is a popular web based course management suite. Faculty can update course information, hold discussions or accept homework over the web. WebCT is fully functional on Mac OS X using a supported browser like Internet Explorer 5.2. Microsoft has announced that this browser will no longer be supported. Safari is now the default browser on Mac OS X, and is not supported by WebCT.

5.4 Geographic Information Systems Support

Geographic Information Systems Support (GIS) is sorely missing in Mac OS X. GrassGIS, an open source alternative, is not as powerful as the leader in the field, ArcGIS which does not currently support Mac OS X. This application is critical to many Social Sciences fields.

5.5 Backup software

Backup software support in Mac OS X is critical. Retrospect is a GUI backup application with a good variety of scripting and backup options. Restoring from backup is straightforward and generally reliable, though it can be slow. Veritas is another possibility. The Veritas NetBackUp client for Mac OS X is command line only, but simple enough for those without much command line experience. There is no Mac OS X Server Veritas application, so the backups must be done to a Windows or Unix server. Restoring is not simple nor fast, but it does restore files as it should.

5.6 Lack of any Mac OS X alternative

Other applications which currently lack any Mac OS X alternative are:

- AutoCAD
- SAS
- Minitab

6 Distribution issues

In an enterprise environment, one of the considerations for deployment of software is its functionality or impact on distribution and maintenance. Information Technology asks two primary questions:

1. "Will it work properly with the infrastructure or impact the methods of maintenance and distribution?"

2. "Does it work with common distribution and maintenance tools?"

Below are some of the most commonly used tools in distribution and maintenance.

- Apple Software Restore:

  Apple Software Restore (ASR) is a command line tool located in `/usr/sbin`. In the enterprise, it is used to quickly clone and distribute cloned hard drive images. Block or file restoration is supported; it can be run locally or with network protocols. Usually, it is used for initial setup or routine re-imaging. Beyond the command line tool, there are many graphical front ends to ASR that can create, distribute and offer additional features. As powerful as ASR is, it does not support software that requires hardware specific information.

- Apple Remote Desktop:

  Apple Remote Desktop (ARD) is a network administration tool that offers many different options to Mac OS X enterprise administrators. In the enterprise, it is used to patch machines with its package installation feature. Also, it can be used to set the machine to NetBoot and be re-imaged. Currently, one challenge with ARD is that package installation cannot be tracked from the ARD administrative computer.

- NetInstall

  NetInstall is a network imaging application that places a simple yet powerful GUI on top of Apple's Apple Software Restore command-line utility. Many institutions use this utility in combination with Apple NetBoot technology. It provides nearly all the

power of the command-line tool, and provides an abstraction from the command-line that many education administrators find comforting. The ability to configure pre-scripts and post-scripts allows admins to fully customize and automate the distribution of software images.

- Radmind

  Radmind is a file system management suite allowing remote administration of Mac OS X clients. At its core, radmind operates as a tripwire, detecting changes in the file system. Additionally, it allows the enterprise administrator to reverse the changes made to the file system. It is used in the enterprise for flexibility and granularity in deployment of Mac OS X clients beyond a cloned hard drive such as adding or removing software. Additionally, it supports individual machine configurations, including the use of hardware specific files.

- RsyncX

  RsyncX is an implementation of rsync with HFS+ support and configuration either through a command line, i.e. terminal or graphical user interface. In the enterprise, it is beneficial to use because it transfers the differences between two sets of files across the network. It is usually implemented with file servers, but can be used to manage the file system of Mac OS X clients, often for initial setup. RsyncX supports hardware specific information via setbymac and setbyip.

- Ditto

  ditto is a command line tool which supports copying files and folders optionally preserving resource forks and HFS metadata. It is often used in managing and distributing pre-created template user directories with scripts. It does not support some attributes, such as file creation dates, stored in preference files, which may be necessary in an enterprise environment.

6.1 Network Home Directories

Many enterprise and lab environments are now supporting Network Home Folders, commonly known as "roaming profiles". The profile travels with the user from machine to machine. In some cases, only the user's /Library folder automatically resides on the network versus the entire home folder. The user directories may reside on a Mac OS X based server or on a server running any network file system. These network file systems include: Apple Filing Protocol (AFP), Samba (SMB), Novell, Network File System (NFS), and Andrew File System (AFS). Often the network file space is accessed from multiple platforms.

Considerations that must be taken into account when developing software for use with network file systems include:

1. Disk Quota - Since the space on the server is utilized by many users, the amount of space or disk quota to each user is given is limited. Even when the amount of space is not specifically set with a quota, the infrastructure is designed with the assumption that not every user will need an excessive amount of space.

   Using the Mac OS X Preferences Hierarchy will minimize the impact on a network file system. An example of a problem is Macromedia Flash MX 2004, which requires up to 75MB of files to be stored in each user's `/Library` folder in order to run. Assuming the users quota is even sufficient to accommodate this amount, it can quickly lead to the need for terabytes of additional storage.

2. Temporary Space - In a network file system, the need for temporary space management is especially critical as the network may not be able to support sustained network throughput required by many audio or video applications. Applications should allow the administrator to set an alternate volume or partition.

3. Host Preferences - In a network file system, host preferences are preferred, as they will not require distribution for each user network home directory.

4. Single-forked file systems - Sometimes home directories are stored on a server that may use a single-forked file system. If application preferences or application files require a forked file system, the application may not function properly.

5. Hard coded paths - Applications that use hard coded paths for preferences may prevent application deployment with network file systems, due to the dynamic nature of each users' home folder.

Many of these issues exist with local home folders, but a network file system magnifies their importance.

6.2 Resource forks

The next major issue with software distribution is the use of forked file systems within Mac OS X. Many applications under Mac OS X store metadata in a "resource fork", not in the "data" fork of the file. Many network storage systems currently used in enterprise environments do not support resource forks. For network file systems, Mac OS X applications will split the resource fork using "._" as a prefix to the file. Propagation of these files on non-forked file systems may cause cross-platform issues. Issues include:

- Inability to mount a network share from a Windows platform machine. In some cases, propagation of the "._" will render the network share unmountable.
- User confusion about the correct file to open. As two files exist, users may not understand which file is the proper data file.
- Corrupted downloads from a web-based file system

6.3 Non-standard serialization

Use of preference files which must be in the user's home directory may not be feasible in an enterprise environment. As an example, QuickTime Pro is licensed per machine, but the license file must exist in each users home folder. In a networked home folder environment, the administrator must copy the license file as a user logs in, and delete it as a user logs out to fulfill licensing requirements. Also, in environments with multiple local home directories, adding or removing a QuickTime Pro license is problematic and requires additional scripting to resolve.

6.4 Scratch space management

In the enterprise, applications can require the ability to control scratch space. Scratch space is defined as temporary space used for creation of temporary files by an application. Applications should allow the administrator to choose an alternate volume or partition through use of the preferences interface. On a publicly shared machine, temporary files have the potential to render a machine useless, requiring the administrator to either remotely delete work, or login locally if the problem is severe enough. As an example, iMovie does not allow the enterprise administrator to choose a directory for temporary file creation.

6.5 ByHost preferences

The ByHost Preferences folder is used by some applications for initial setup or customization. The ".plist" files inside the ByHost folder use hardware information from the machine that created them. They are usually created in `~/Library/Preferences/ByHost` with the format `com.company.name.product.123456789012.plist`. When applications depend on machine specific information, this adds a burden on distribution and deployment of the software. Examples of programs that follow this practice are Maya and EndNote. Each places hardware-specific files in `~/Library/Preferences/ByHost`. Placement of preferences in this location makes the application available only to that user on the original model machine where the application preference was customized. In a multi-user enterprise environment, where users must be allowed access to machines in public computing facilities, libraries or kiosks in public spaces, use of global preferences is preferred.

For more information, see section 8.5

6.6 Auto launched application preferences

Installers for some applications will modify one of two files:

```
~/Library/Preferences/loginwindow.plist
/Library/Preferences/loginwindow.plist
```

to add items to the "AutoLaunchedApplicationDictionary". This modification presents a number of problems in an enterprise environment:

First, If the /Library/Preferences/loginwindow.plist file is modified, the application will be auto-launched only for the user under which the application was installed. Generally, this behavior is not the desired result. As an example, the installers for the Canon LiDE scanners add a login item to /Library/Preferences/loginwindow.plist which monitors the buttons on the scanner. This application is only launched for the user that installed the software; other users will have non-functioning scanner buttons unless this login item is added to their /Library/Preferences/loginwindow.plist file.

Second, if the file /Library/Preferences/loginwindow.plist is modified, the application will be automatically launched for all users of the machine, but the challenge then becomes how to manage this in an environment where each workstation may have a different mix of applications. Solving this requires either some creative scripting by the administrator or the willingness to manage many different versions of the /Library/Preferences/loginwindow.plist file, or visiting each machine for specific configuration.

Applications that modify either /Library/Preferences/loginwindow.plist or /Library/Preferences/loginwindow.plist include Palm Desktop and Norton Anti-Virus.

6.7 File Attributes

Some applications use configurations that depend on file attributes such as the creation date. Currently, these are not fully supported by tools used to deploy and manage Mac OS X file systems and applications. Applications that depend on these unsupported file attributes will cause distribution issues, requiring software re-configuration and loss of application default settings.

For example, if standard tools are used to distribute applications and manage user file space, Microsoft Office X will display the message "Configuring Components". Tools and applications used to distribute Microsoft Office X, and management of user file space do not preserve the creation date of file system objects in Mac OS X. Microsoft Office X checks the creation date of this file /Library/Preferences/Microsoft/Carbon Registration Database. If the creation date does not agree with the time of installation, the message will appear, causing users to wait for unnecessary reconfiguration.

7 Troubleshooting

Detecting installation and software irregularities is one of the larger challenges in deploying any software. The first step is to create a baseline installation. A baseline installation can be defined in many ways; but in this context, a baseline installation is best defined as an installed Mac OS X system that is in a known working state. For example, many enterprise administrators install the base Mac OS X system. The base installation becomes the launching point for any troubleshooting and further installations.

7.1 Review the Installer

If possible, when beginning to troubleshoot an application, an enterprise administrator can first review the installer for potential problems. This technique will only work with PackageMaker packages and Drag and Drop installs. It will not work with any installs created with a third party product.

7.1.1 Installer

Apple's Installer application, which is used to install package installers, can also can provide limited information about the files being installed by the package installer. Simply open up the package installer inside the Installer application, then select `Show Files` command from the File menu. This will display a dialog with limited path, file and directory information.

The installer application also provides an installation log, showing all steps of the installation. Any scripts run as part of the installation will appear in this log, such as permission corrections, or temporary file creation. Three options are available using this log, Show Everything, Show Errors and Progress, and Show Errors Only. This feature is also accessed from the File menu.

7.1.2 Pacifist

Pacifist allows inspection of packages created by PackageMaker prior to installation of any package. It can be examined before installation to determine its affect on the baseline

installation. As an example, the enterprise administrator can extract parts of a package installer such as GarageBand. On reviewing the installer, an administrator can observe that it installs the application in the directories `/Applications` and `Application Support`, and MIDI Drivers in `/Library`. The administrator can also view permissions on all installed files.

Pacifist can also be used instead of the normal Installer, which can be helpful if the administrator experiences problems in the software installation package. Pacifist also includes the ability to verify existing installations, comparing the files of an installed package with the files listed in the package installer. In addition, it will detect missing files, files that have different permissions or checksums, and can update prebinding information, either for the entire hard disk, or for an individual folder.

N.B. Pacifist does not allow an enterprise administrator to examine any scripts that are run as part of the installation process by Installer. Other methods must be used to view installer scripts.

7.1.3 lsbom

The command line tool `lsbom` can be used to examine package contents as well. From the terminal, change to the `/Contents/Resources` directory within the installer package directory. In this directory, there are many files that can be examined. The `lsbom` tool can be used to examine the `*.bom` file for file installation locations, as well as permissions. In this directory, scripts may exist as well, which can be examined for their effect on the installed system.

7.2 Tracking

After examination of the installer, if applicable, the next step in troubleshooting an installation is to install the software application, launch the application, enter either registration or license information, and create default user settings. At this point, the enterprise administrator can use tools to examine affected areas of the clean installation and software initial setup.

7.2.1 File Buddy

File Buddy, is a popular and very powerful file utility for Mac OS 9 and Mac OS X. It has multiple features including viewing file and folder info, editing file and folder info, finding files, finding and fixing orphaned aliases, removing unused files and much more. It also allows the administrator to capture a snapshot of a hard drive before and after applications installation and setup to detect differences. It uses file and folder attributes such as modification date and size; however, it does not use checksums when gathering differences. A checksum is a value that is computed based on the contents of a set of constant data such as files created or modified by an installer. Checksums are used to detect data integrity, which may not be caught using modification date and file size attributes. Additionally, it does not support owner and group file permissions. Ideally, due to security and accuracy concerns, the enterprise Information Technology administrator uses checksums when tracking installation and distributing software.

7.2.2 logGen

logGen is a freeware command line utility. It can be used to detect file system changes after a preference change or package installation. It accomplishes this by utilizing a number of methods, but primarily uses the modification date and a checksum of each file. This tool is only compatible with Mac OS X 10.3 and above, due to some perl modules that are missing in earlier versions. To use the tool, take a baseline snapshot of the file system using the following command:

```
sudo /usr/local/sbin/logGen before_snapshot.dat
```

Next, install software and enter either application registration or license information. Create default user settings and run the tool again.

```
sudo /usr/local/sbin/logGen after_snapshot.dat /
before_snapshot.dat > changes.txt
```

Next, the data is compared between the after_snapshot.dat and before_snapshot.dat files and the changes are summarized and saved to changes.txt. This tool is very useful for determining files used to build Mac OS X packages as well

7.2.3 Radmind

Radmind is a suite of command line tools with corresponding GUI tools that can be used to determine what was installed by a particular installer. After initial setup of a managed client with the basic system, i.e. baseline, you can track the additional installations and setups , i.e. overloads, with radmind. As an example using the radmind command line tool fsdiff:

```
fsdiff -C -c sha1 -0/var/radmind/client/loadset.T /
```

After checking file system differences, the administrator can use the Transcript Editor or favorite text editor to review a transcript of the complete install. The setup, including permissions, privileges, and files modified by the installer, can be examined.

7.3 Common Problems

7.3.1 File Permissions

File permissions are often created in a non-enterprise friendly way. There are many tools to examine permissions created on files and directories by installers. As mentioned previously, Radmind (section 7.2.3) can be used to determine file permissions. Many tools also exist to examine file permissions, including:

1. The Finder

   In a limited way, the Finder can show permissions on file folder and directories. It cannot show the execute bit. If the Finder is on a computer with directory access, this option may require additional time as the computer determines file ownership.

2. GUI File Permission Tools

   If file permissions need to be modified from a graphical environment XRay, and FileXaminer can be used to examine each file. Additionally, more obscure file per-

missions can also be set using the application, including the visibility of a file or directory.

3. Command Line Options

   To examine permissions from the command line, issue the following command: `ls -lo`. All permissions will be listed in human-readable form, as well as the special BSD file permissions. To find world writable folders, issue the following command: `sudo find -x /Applications/Myapp.app -perm -0777 ! \( -type d -perm -1777 \) ! -type l -print`.

   The above command will only find files and directories that are world writable inside the application package MyApp.app. World writable files installed outside of the application package will not be found. It also skips any symbolic link that is world writable. SUID files can be found with the following command: `sudo find -x / -type f -perm -04000 -print`. SGID files can be found with the command `sudo find -x / -type f -perm -02000 -print`

   To examine file creator types and attributes, the command tool `GetFileInfo` installed with the Developer Tools is used.

   A script to check for broken symbolic links can be found at Unix Review. Listed below is the script in it's entirety. If the root file system is specified, this script *will* attempt traversal of network directories, even if not mounted. The script can be found in Appendix A

As mentioned previously, Set-UID, Set-GID, and world writable directories may need modification in an enterprise environment. To review, see section 3.1.

7.4 Test

After fixing common problems with a software application, it should be tested for any additional problems. Testing should occur as a non-administrative user, not as the administrator of the machine as problems may not appear for an administrator. Generally, global features such as printing, help, spell-check, and saving files should be tested. In some

cases, application specific features should be tested. As an example, each application in iLife has interdependencies that should be tested.

7.5 Distribute

After the software application has been thoroughly tested, it should be tested using the distribution methods chosen by the enterprise administrator. If all goes well, the application can be distributed.

7.6 Complex installations

If after the previous two phases, the software application does not work, command line tools may be needed to isolate the specific problem.

7.6.1 lsof

The first tool in an administrator's troubleshooting toolbox is `lsof`. Entering the command:

```
lsof +D /Library/Application\ Support
```

will watch a particular directory for any files that are opened by an application. For administrators uncomfortable at the command line, Sloth provides a GUI to this utility.

7.6.2 fs usage

The next tool, fs_usage, is one of the most useful tools in determining an application's file system requirements. The following command:

```
fs_usage -w -f filesys [executablename] | grep -v CACHE_HIT
```

will show all file system objects accessed by the command executable. The output will contain files needed by the application, including preference files, as well as any other files used by the application.

7.6.3 ktrace

Kernel tracing `ktrace` can be used to observe many types of information generated by any application. From the manual page, options include:

```
-t trstr
The string argument represents the kernel
trace points, one per letter.
The following table equates the
letters with the tracepoints:
c     trace system calls
n     trace namei translations
i     trace I/O
s     trace signal processing
u     userland traces
w     context switches
```

To observe input/output operations, enter

```
ktrace -ti -p 67
```

This command will trace all input/output operations on process 67. Other options can be used to determine problems with the application. Output is written to a machine readable file, which is then read with the `kdump` utility.

8 Solutions

Many of the problems listed above can be overcome using some of the following solutions. These solutions are intended to be general guidelines, and should help the enterprise administrator to overcome problematic applications. These solutions are also instructive to the software developer, showing the methods that are used to overcome software problems.

8.1 Installers

Some installers will not run in an administrative context, and must be run with super-user *root* privileges. A number of tools exist which can be used to provide root authorization, without enabling the root user. Among these tools are Pseudo and the command line tool sudo.

8.2 Security solutions

Some application's security problems can be worked around with the use of a *shadow* file. This solution can be used for the Classic environment, or any other application that needs full write access to a protected area. The command: `hdid /Classic.dmg -shadow ~/Library/Management` will mount the Classic disk image as a shadow file. This file can then be mounted, and used, but will not save any information once logged out. Any application can take advantage of this method, allowing for more secure use. Mike Bombich has also provided an application to accomplish this task ShadowClassic

Other applications can take advantage of this method. As an example, the script below will run VirtualPC as a 'shadow' file.

```
do shell script "/usr/bin/hdid ' /
Applications/Virtual PC 6.0.1/Windows 98 SE.dmg'
 -shadow VPCShadow'"
delay 5
tell application "Finder"
open file "Mac OS X:Applications:Virtual PC 6.0.1"
end tell
```

N.B. Shadowfiles created with this command can be quite large. They should be deleted at logout.

8.3 Startup and Logout scripts

Startup and Logout scripts can be used to clean up directories which cannot be solved using shadow files. As an example, the latest version of MacLinkPlus Deluxe which requires

world rights to the directory `/Library/Application Support/DataViz` as well as a number of aliases. Without world rights, the user will be prompted for serial information each time the application is used. A solution from the University of Utah is listed in Appendix B. For more information about login and logout hooks, visit the macosxlabs Script Archive

8.4 Alias and symbolic links

Some applications can use aliases and symbolic links to give users access to files needed to run the program. Move the required files to a user-writable location. Place aliases to these files in their original locations, and try launching the application. If aliases do not work, try creating symbolic links via the command line,

```
ln -s /path/to/source/file /path/to/target/file
```

Try launching the application and see if it behaves normally.

8.5 Global placement of preference files

In Mac OS X, a hierarchy of folders designed to contain and maintain various settings for users exists. Ideally, an application should use this hierarchy, instead of restricting or specifying a specific preference location. By using global preferences, deployment of any application becomes more flexible.

Preferences commonly reside in one of three scopes: the user scope, `/Users/<username>/Library/Preferences` which limits it to a specific user; the host scope `/Library/Preferences` which limits it to users of a machine; or the network scope, `/Network/Library/Preferences`, which makes it available to all users of a network.

Preferences stored at the User preference level can be relocated to a higher level of the preference hierarchy. This relocation allows the application to be used by all users of the machine. For example, the Slick iMovie plug-ins can be installed in the global directory, and it will be available for all users.

The routines for retrieving preferences search through the preference domains in the order shown in Table 1 until they find the requested key. Thus, if a preference is not found in a more user-specific and application-specific domain, the routines search the more global domains for the information. For more information, see User Preferences Hierarchy

Table 1: "Mac OS X Preferences Hierarchy"

Search order	User Scope	Application Scope	Host Scope
1	Current User	Current Application	Current Host
2	Current User	Current Application	Any Host
3	Current User	Any Application	Current Host
4	Current User	Any Application	Any Host
5	Any User	Current Application	Current Host
6	Any User	Current Application	Any Host
7	Any User	Any Application	Current Host
8	Any User	Any Application	Any Host

8.6 File Attributes

A solution for software applications that depend on file attributes is using the `GetFileInfo` and `SetFile` tools. These tools are available as part of Apple Developers Tools. They are located in the `Tools` directory.

The first step in checking attributes is a clean install. After the clean install, the administrator can check the problematic file using the `GetFileInfo` tool. An extended set of information is returned with this tool. After installation, an administrator should apply the file system management tool they are using. Next, compare the updated file's attributes to the managed file. Note the differences, and use the tool `SetFile` to set the file's attributes. As an instructive example, Microsoft Office X stores the creation date of the following file, `~/Library/Preferences/Microsoft/Carbon Registration Database`. The following tools do not preserve the creation date, causing the "Configuring Components" message to launch for each user:

- Carbon Copy Cloner
- cp

- ditto -rsrcFork
- CpMac
- radmind
- zip
- tar

The following tools preserve the creation date, eliminating the ”Configuring Components” message:

- Finder
- Apple Remote Desktop
- RsyncX
- Path Finder
- StuffIt (.sit format)

GetFileInfo returns the following information on the above file:

```
file: ''/Users/user/Library/Preferences \
/Microsoft/Carbon Registration Database''
type: ''pref''
creator: ''RgEd''
attributes: avbstclinmed
created: 09/26/2002 12:03:23
modified: 01/19/2004 17:46:59
```

Note the date returned from GetFileInfo. After file system management, use the SetFile utility with the following command:

```
/Developer/Tools/SetFile -d \
''09/26/2002 12:03:23'' \
''/path/to/copy/Library/Preferences/ \
Microsoft/Carbon Registration Database''
```

This command can be included in a login hook, post file system management script, or maintenance script.

8.7 Repackaging tracked installers

If a software application has been tracked properly, it can be captured and re-created as an Apple installer package. In this way third-party installers, drag and drop applications, and improperly created packages can be recreated to meet enterprise information technology's requirements. Advantages of packages include:

1. Permissions are read when "Repair Permissions" is run from "Disk Utility"
2. Software can be distributed through NetInstall and Apple Remote Desktop
3. A log of the installation is kept in `/Library/Receipts`

PackageMaker documentation provides the basics of creating a software package.

9 Conclusion

This whitepaper has presented challenges and solutions faced in an enterprise environment deploying Mac OS X. With the whitepaper's recommendations , both enterprise information technology administrators and developers have a potential roadmap for the deployment of software in the enterprise. As an enterprise administrator, security policies, efficiency and cost are benefits of using the techniques presented. From a developer's perspective, the guidelines presented in this whitepaper allow for wider deployment of a software product in the enterprise.

Challenges outlined in this whitepaper could be reduced if a built-in package and patch management mechanism existed in Mac OS X. With Mac OS X 10.3 and earlier releases, the official "patch management" mechanism, Software Update, only works with Apple software and does not support third party software. Inventorying installed software requires using the methods presented in this paper, without a built-in global option that works with third-party software.

Another important concern in management and distribution of software is assessing updates and patches. Without a programmatic method, administrators must manually review multiple sources and web sites. If Software Update supported third-party software, inventory, tracking updates and disseminating update information would save administrators time.

Lastly, with many installers, a built-in method of tracking installation does not exist. Package installers can be reviewed, but with use of pre and post install scripts there is no guarantee that the installer uses the ”Bill of Materials” file only.

Many solutions could be used to solve this problem. One solution is the implementation of a tripwire. This would allow separate software installations, providing support for granularity in software distribution that many enterprises require.

References

[1] Australian Computer Emergency Response Team. *Unix Security Checklist 2.0*. web page 2001.

[2] Apple Computer. *Tools: Software Distribution: Authorization, File Ownership, and Permissions*. web page 2003.

[3] Apple Computer. *System Overview: Installation and Integration*. web page 2003.

[4] Apple Computer. *The Mac OS X File System: Aliases and Symbolic Links*. web page 2003.

[5] University of Utah. *Application Deployment Problems and Solutions*. web page 2004.

A LinkCheck script

```
#!/usr/local/bin/perl -w
#
# SccsId[] = "%W% %G% (Link check Perl program)"
#
#-------------------------------------------------------------------------#
#                            linkcheck.pl                                 #
# ----------------------------------------------------------------------- #
#        Program documentation and notes located at the bottom.           #
#-------------------------------------------------------------------------#

   BEGIN { $diagnostics::PRETTY = 1 }

   $SIG{'INT'} = sub {print "\nExiting on $SIG{'INT'}\n";exit $SIG{'INT'}};

   use File::Find;
   use Getopt::Std;
   use Cwd;
   use POSIX qw(uname);
   my $host =  (uname)[1];

   use vars qw($opt_a $opt_H $opt_h $opt_l $opt_r $opt_v);
   my $options='aHhlrv';
   exit_usage("Invalid option!\n") unless (getopts($options));
   show_documentation()                if ($opt_H); # Full documentation
   exit_usage()                        if ($opt_h); # or usage brief.
   exit_usage("Filesystem required.\n") if ($#ARGV < 0);

   if ($opt_v)
   {
      use diagnostics;
   }

 #-----------------------------------------------------------#
 # Eliminate all but local filesystem searches right away. #
 #-----------------------------------------------------------#
   my $local_fs;
   my @search;
   foreach (@ARGV)
   {
      if ($local_fs = 'df -lk $_')
      {
         push(@search, $_);
      }
```

```
      else
      {
         print "File system $_ must be local to $host, not NFS mounted.",
               "\nSkipping $_.\n";
         $_ = "";
      }
   }

#---------------------------------------------------------------#
# Ignore find command's stderr output (eliminates "Permission #
# denied" and most--not ALL--other bothersome messages).      #
#---------------------------------------------------------------#
   open(OLDERR, ">&STDERR");
   open(STDERR, ">/dev/null") or die "Can't redirect stderr: $!";

   my $q = 0; # Found counter
   my $r = 0; # Removed counter
   find sub # [Anonymous] subroutine reference (called a coderef).
   {
      return unless -l "$_"; # Skip all but links.
    #-------------------------------------------------------------#
    # Skip nfs mounted links, and /proc and /cdrom pathnames.  #
    #-------------------------------------------------------------#
      return if (
                  (lstat("$_"))[0] < 0
                ||
                  $File::Find::name =~ /\/proc/s
                ||
                  $File::Find::name =~ /\/cdrom/s
                );

    #-------------------------------------------------------------#
    # Skip link if it's not on a local filesystem as well.     #
    #-------------------------------------------------------------#
      my $dir = cwd;
      return unless ($local_fs = `df -lk $dir`);

      $! = 0; # Clear error message variable
      return unless defined(my $target = readlink("$_"));

      my $error  = "$!";
         $error  = "($error)" if (defined($error) && $error ne "");

      my $ls_out = ($opt_l)
        ? `ls -albd $File::Find::name 2> /dev/null`
```

```
      : "$File::Find::name -> $target";

    chomp($ls_out);

    unless (-e "$target") # Unless the link is OK, do the following.
    {
       $q++;
       print "Broken link: $ls_out $error\n";
       if ($opt_r)
       {
          print  "rm '$File::Find::name'\n";
          if (unlink("$File::Find::name") == 0) # Zero = none deleted.
          {
             print "Unable to remove $File::Find::name!\n";
             return;
          }
          $r++;
          print "Removed '$File::Find::name'\n" if ($opt_v);
       }
       return;
    }

   #-------------------------------------------------------------#
   # Return unless user requests list of all links (-a).        #
   #-------------------------------------------------------------#
    return unless ($opt_a);

    if    (-f "$target") { print "Linked file: $ls_out $error\n"; }
    elsif (-d "$target") { print "Linked dir:  $ls_out $error\n"; }
    elsif (-l "$target") { print "Linked link: $ls_out $error\n"; }
    elsif (-p "$target") { print "Linked pipe: $ls_out $error\n"; }
    elsif (-S "$target") { print "Linked sock: $ls_out $error\n"; }
    elsif (-b "$target") { print "Linked dev:  $ls_out $error\n"; }
    elsif (-c "$target") { print "Linked char: $ls_out $error\n"; }
    elsif (-t "$target") { print "Linked tty:  $ls_out $error\n"; }
    else                 { print "Linked ???:  $ls_out $error\n"; }

    $error = "";
    return;
  }, @search; # find sub

#---------------------------------------------------------------#
# Restore stderr.                                               #
#---------------------------------------------------------------#
  close(STDERR) or die "Can't close STDERR: $!";
```

```
   open( STDERR, ">&OLDERR") or die "Can't restore stderr: $!";
   close(OLDERR) or die "Can't close OLDERR: $!";

   print "$host: Found $q broken links.  Removed $r.\n";
   exit 1;


#=============================================================================#
#            S U B R O U T I N E S  /  F U N C T I O N S                      #
#                      (in alphabetical order)                                #
#-----------------------------------------------------------------------------#
sub exit_usage # Exits with non-zero status.                                  #
               # Global vars:   $main::notify                                 #
               #                $main::support                                #
#-----------------------------------------------------------------------------#
{
   my $fn_name = "exit_usage";
   my $txt     ;

 #---------------------------------------------------------------#
 # Assign to private variable, $notify either $main::support or  #
 # $main::notify (takes $main::support over $main::notify).      #
 #---------------------------------------------------------------#
   my $notify;
   if (defined($ENV{LOGNAME} )) { $notify = $ENV{LOGNAME}; }
   else                         { $notify = $ENV{USER};    }

   $txt =  "Usage:   $0 -$options fs ...\n";
   $txt =  "$_[0]\n$txt" if ($#_ >= 0); # Prefix message arguments
   $txt .= "\n         -a = Display All links."
        .  "\n         -H = Displays full documentation."
        .  "\n         -h = Gives usage brief."
        .  "\n         -l = Long list (e.g. 'ls -al')."
        .  "\n         -r = Remove broken links (use with caution)."
        .  "\n         -v = Verbose output."
        .  "\n         fs = Required filesystem for search."
        .  "\n              (multiple filesystems may be specified)\n"
        .  "\nPurpose: Search filesystem (descending directories) for"
        .  "\n         broken links, optionally displaying all links"
        .  "\n         (-a) and/or removing (-r) them.\n";

 #---------------------------------------------------------------#
 # If running interactively, then give'm usage, else notify      #
 # program support person(s) because a cron'd job called usage.  #
 #---------------------------------------------------------------#
```

```
   print "$txt";

   exit 1;
} # sub exit_usage

#-----------------------------------------------------------------------#
sub show_documentation # Display program documentation at bottom.      #
#-----------------------------------------------------------------------#
{
   my $n = 0;
   foreach (my @doc_lines = <main::DATA>)
   {
      print "$_";
   }
   exit $n;
} # sub show_documentation


__END__ # Documentation section follows:
#=======================================================================#
#                      D O C U M E N T A T I O N                        #
#=======================================================================#
#                                                                       #
#      Author: Bob Orlando                                              #
#                                                                       #
#        Date: April 29, 2002                                           #
#                                                                       #
#  Program ID: linkcheck.pl                                             #
#                                                                       #
#     Purpose: Search local filesystem or systems                       #
#              (descending directories) for broken                      #
#              links, optionally displaying all links                   #
#              (-a) and/or removing (-r) them.                          #
#                                                                       #
#       Usage: linkcheck.pl -aHhlrv fs ...                              #
#                           -a = Display All links.                     #
#                           -H = Detailed documentation.                #
#                           -h = Usage brief.                           #
#                           -l = Long list (e.g. 'ls -al').             #
#                           -r = Remove broken links                    #
#                                (use with caution).                    #
#                           -v = Verbose output.                        #
#                           fs = Required filesystem for search         #
#                                (multiple filesystems may be           #
#                                specified)                             #
```

```
#                                                                      #
#    Examples: linkcheck.pl /        # Lists broken links (short list) #
#              linkcheck.pl -l /     # Lists broken links (long list)  #
#              linkcheck.pl -a /home # Lists all links in /home.       #
#              linkcheck.pl -r /usr  # Removes broken links from /usr. #
#                                                                      #
#     Returns: Zero on success.                                        #
#              Nonzero in failure.                                     #
#                                                                      #
#----------------------------------------------------------------------#
```

B Logout Script

Logout script from University of Utah[5]

```
#!/usr/bin/perl
@dataviz_ignore_files = ("MacLinkPlus CM Helper", "sysKas.5560",
"MacLinkPlus  Deluxe", "MacLinkPlus Modules");
$dataviz_erase_folder = "/Library/Application Support/DataViz";
# Empty DataViz world writable folder.
& removeAllFilesInFolderExcept($dataviz_erase_folder, /
@dataviz_ignore_files);
########

sub removeAllFilesInFolderExcept {
my ($pathToFolder, @allowedFiles) = @_;
if (! -d $pathToFolder ) {
print "folder doesn't exists";
return;
}
@currentFiles = getFileFolderList($pathToFolder);
@killMe = ();
$repeat1 = @currentFiles;
$repeat2 = @allowedFiles;
for ($i = 0; $i < $repeat1; $i++) {
$iDie = 1;
for ($j = 0; $j < $repeat2; $j++) {
if($allowedFiles[$j] eq $currentFiles[$i]) {
$iDie = 0;
}
}
if ($iDie) {
push(@killMe,$currentFiles[$i]);
}
}
$repeat1 = @killMe;
chdir $pathToFolder;
for ($i = 0; $i < $repeat1; $i++) {
print "/usr/bin/chflags -R nouchg \"$killMe[$i]\"\n";
print "/bin/rm -rf \"$killMe[$i]\"\n";
system "/usr/bin/chflags -R nouchg \"$killMe[$i]\"";
system "/bin/rm -rf \"$killMe[$i]\"";
}
}
```

C URL references

IANA

http://www.iana.org/assignments/port-numbers

KeyServer

http://www.sassafras.com

FlexLM

http://www.rainbow.com/products/sentinel/index.asp

Steinberg

http://www.steinberg.net

LightWave

http://www.newtek.com

QuarkXPress

http://www.quark.com

Reason

http://www.propellerheads.se

Bryce

http://www.corel.com

Practica Musica

http://www.ars-nova.com

Graphing Calculator

http://www.pacifict.com

Virtual PC

http://www.microsoft.com

Microsoft

http://www.microsoft.com/technet/security/bulletin/MS04-005.mspx

Apple Software Distribution Concepts

http://developer.apple.com/documentation/DeveloperTools/Conceptual/SoftwareDistribution/Concepts/sd_permissions_author.html#//apple_ref/doc/uid/20001769/TPXREF3

Installer Vise

http://www.mindvision.com/Products.asp

InstallAnywhere

http://www.zerog.com/products_ia_09.html

Mac OS X System Overview - Installation and Integration

http://developer.apple.com/documentation/MacOSX/Conceptual/SystemOverview/InstallIntegrate/chapter_11_section_4.html

Symbolic Links and Aliases

http://developer.apple.com/documentation/MacOSX/Conceptual/BPFileSystem/Concepts/Aliases.html

Painter 7

http://www.corel.com

PeopleSoft

http://www.peoplesoft.com

GrassGIS

http://grass.itc.it

AutoCAD

http://www.autodesk.com

SAS

http://www.sas.com

Toast

http://www.roxio.com

MacOpener 2000

http://www.dataviz.com

Radmind

http://www.radmind.org

RsyncX

http://www.macosxlabs.org/rsyncx/rsyncx.html

Maya

http://www.alias.com

EndNote

http://www.endnote.com

Pacifist

http://www.charlessoft.com

File Buddy

http://www.skytag.com/filebuddy

XRay

http://www.brockerhoff.net/xray/

FileXaminer

http://www.gideonsoftworks.com/filexaminer.html

Unix Review

http://www.unixreview.com/documents/s=1344/uni1029250523965/0208f_script.htm

Sloth

http://sveinbjorn.vefsyn.is/software

ShadowClassic

http://www.bombich.com/software/shadowclassic.html

MacLinkPlus Deluxe

http://www.dataviz.com

Macosxlabs

http://www.macosxlabs.org/documentation

User Preferences Hierarchy

http://developer.apple.com/documentation/MacOSX/Conceptual/BPRuntimeConfig/Concepts/UserPreferences.html#//apple_ref/doc/uid/20002092

Pseudo

http://personalpages.tds.net/~brian_hill/pseudo.html

logGen

http://www.lsa.umich.edu/lsait/AdminTools/osx/software

Apple Remote Desktop

http://www.apple.com/remotedesktop

PackageMaker

http://developer.apple.com/documentation/DeveloperTools/Conceptual/SoftwareDistribution/Concepts/sd_install_quick_look.html

Minitab

http://www.minitab.com