

University of Sydney Faculty of Architecture Network Installers

Jason Thorne
jason@arch.usyd.edu.au

Andrew Winter
andy@arch.usyd.edu.au

The Faculty of Architecture
Sydney University
Australia

Abstract

We have developed and implemented a system that will autonomously install and configure a Macintosh computer to recorded specifications attributed to the machine. The implementation involves the use of unique Apple technologies such as AppleScript and the Apple System Profiler. The system keeps a history of installs, and creates an asset management database.

Introduction

The Faculty of Architecture at the University of Sydney runs a world class design computing environment, comprising of Power Macintoshes, Sun/SGI Unix work stations, and Pentium PCs. Management and support is provided by finite human resources whilst the environment is continually expanding. Installation and configuration of Macintosh machines has been a labour intensive and time consuming task, which prompted us to develop the automated solution which we have called "Network Installers" or NWI for short.

This is a basic outline of how Network Installers works.

- The entire system would be stored in a "build" volume that is shared on the appletalk network.
- The machine(s) to be rebuilt start up from a separate volume or a CD, which contains applications to install/update the system.
- The install updates the system/applications, restarts with the new system, and enables the configure to execute on start up.
- The configure determines file sharing, sets the machine name, adds apple menu items, aliases etc, according to a unique record appertaining to the machine. Once finished, it disables itself from running at start up, and restarts the machine.

Software Tools Used

AppleScript and Scripting Additions

Some good things about AppleScript:

- The majority of Macintosh applications and operating system features are scriptable. This simplifies configuration process a great deal, which would otherwise require us to manipulate preference files at a fundamental level.
- It's free and is installed as a standard part of the Macintosh OS.
- The documentation is free, and available at Apple's WEB site.
- AppleScript is easy to learn, and has a 4GL feel that makes complex tasks simple.
- There is a great deal of on-line help, including WEB pages, script down-loads and mailing lists

We also use free third party scripting additions. Scripting additions (also known as osaxen) are compiled C/C++ code that enhances the functionality of AppleScript. One such scripting addition is "Jon's Commands" which Network Installers uses for deleting locked files from the trash and getting the hardware address of the ethernet card. For a list of available scripting additions and their functionality, see <http://cgi.scriptweb.com/osaxen/>.

On Guard from Power On software

We use the security software “On Guard” for limiting access to the operating system and applications. On Guard is scriptable and NWI’s AppleScripts can log on and load up a security configuration file according to the intended user of the machine.

System Picker

This is also scriptable, and allows the install script to restart the machine with a chosen system folder. Though this is no longer used in NWI, it does come in handy to restart from a chosen OS folder.

Assimilator

Assimilator Manager creates an application program that compares the existing system to one on a remote volume and creates/deletes/replaces any files necessary to create an identical system on the machine being built. Assimilator can do much more than this, but this is all that is necessary for NWI.

Apple System Profiler

NWI uses the version of this application that came with MacOS 9. It is scriptable, and will return a host of information about the machine, including the type of logic board, the clock speed, the amount of memory and serial number of the machine. Figure 1 is a screen capture of the information given by the application, all of which can be returned to AppleScript from this application.

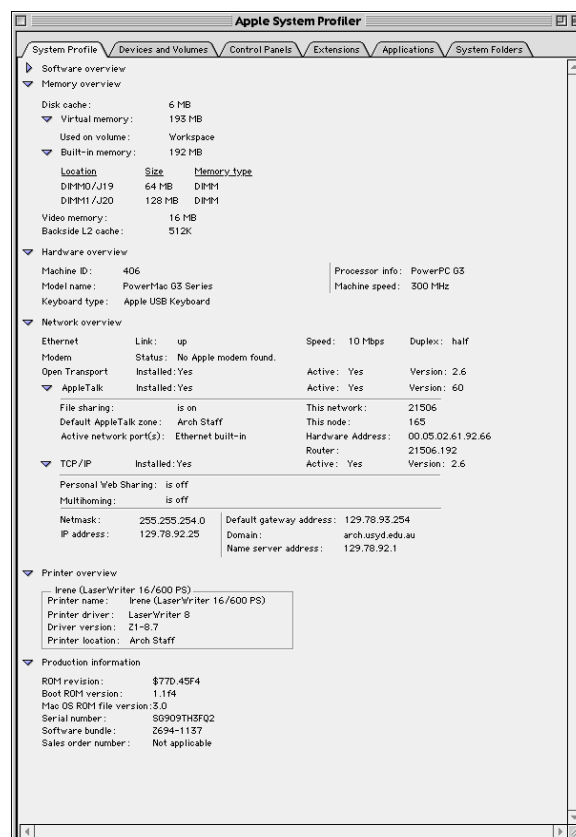


Figure 1. Apple System Profiler Information

The Install

The Build and The Golden System

Before an installation/configuration solution can be used, we need to prepare an operating system and application set to install. The Macintosh allows us to create an operating system folder that will run on any

compatible Macintosh hardware (try doing *this* on Windows 95 or NT). NWI uses just such a system folder to be installed on many types of Macintosh, with all the necessary extensions, control panels and preferences to run the standard set of applications with the standard configuration. We call this the “Golden System”, and all applications that are to be installed in the labs are first installed on the Golden System. Before installing any new applications, I remove the following folders from the Golden System folder, and replace them with empty folders of the same name:

- Apple Menu Items
- Control Panels
- Claris
- Extensions
- Preferences

I then run the install application, and when it has finished, I don't restart the Macintosh, but quit out of the installer application (by force quitting if necessary), then check all of the items installed in each of the folders above, to see if they are necessary, or if they are later versions than those existing in the Golden System. This way I can make sure redundant software is not installed and record what system bits are installed with a particular application.

Down-loading the Build

The build exists on a remote volume, and consists of the Golden System and the standard application set, which we have split these into three groups stored in three folders, namely Applications, Utilities and Communications.

The use of Assimilator greatly reduces the amount of network traffic if the existing system is fairly complete. If this is not the case, say with a new computer, Assimilator will generate a large amount of network traffic. To try to minimise this, as much of the build that will fit is burned on the installation CD. When the install script starts, the user is prompted as to whether the install is a complete rebuild (eg; for a new machine) or an update to the existing system. If the user selects rebuild, the existing system is trashed, and the incomplete system is copied off the CD before Assimilator is run. If update is selected, then nothing is done to the existing system before Assimilator is run.

Recording Events and Configuration Information

The Install script gets the hardware address of the ethernet card, then opens/creates a file on a specific remote volume with the naming convention “<hardware address>-report”. This file is used to note events, and when they occurred. The install script then opens/creates a file with the naming convention “<hardware address>-info”. If this file does not exist, or is incomplete, the user is prompted for the information, if not the user is prompted if they wish to change the existing information.

The Configuration

The configuration script is activated to run on start-up at the point when the install has finished. When the script runs, it opens the remote file “<hardware address>-info” which contains the configuration information recorded at the beginning of the install. If this file does not exist or is incomplete, the user is prompted to update the information, and the script stops. The data elements stored in the configuration information file are:

- The machine ID: A unique five-character identification label. We allocate these and maintain a list manually at present.
- The model of Macintosh: For example, “G3”, “7500”, etc
- The owner type: “Staff” for a staff member or “Lab” if it is a laboratory machine.
- The machine name: The name assigned to the machine. This is how the Macintosh is represented in the chooser.
- The user name: The first name of the staff member, or the type of machine if it is a lab machine. This user is then allowed to connect to the data folder of the machine. In a lab, every machine can see every others data folder via the chooser.

- The staff group: Faculty staff, research staff, etc. Null if lab machine.
- Additional applications: A comma delimited list of key field identifiers for any non-standard applications to be installed. This identifier will be the name of a folder on the remote build, containing the application, control panels, extensions etc that are needed for installation.
- The hardware details of the Macintosh: A delimited list including memory, USB devices, firewire devices, hard disk size and the serial number among other things. The Apple System Profiler returns this information.
- End of record: to indicate that the record is complete.

The Macintosh is configured according to the above mentioned data. The following out lines the changes resulting from running the configuration script.

- Aliases are created, pointing to all application files and aliases installed on the machine. These aliases are then placed in folders in the “Apple menu items” folder, so that the apple menu resembles the “Start” menu on PC Windows.
- If it does not already exist, a folder is created for the user to store data items such as documents. This is called the “Data” folder. Applications are configured to read from, save to and down-load to this area, by virtue of the preferences stored in the Golden System, and permissions set in On Guard.
- Sharing is set on the Data folder according to the type of install. For example, all the Macintosh machines in our labs can copy to or from the data folder on every other laboratory Macintosh.
- The script then configures On Guard, by instructing it to load up the configuration file determined by the type of owner of the machine.

Running NWI

NWI consists of a start-up CD containing

- A limited OS with Appletalk and TCP/IP enabled, which is active when the computer is started from the CD.
- Several scripts including the install script and a copy of the configure script
- The Assimilator application
- Useful utilities
- The AppleScript editor.
- Some bits and pieces that need to be installed in specific situations after the standard install has occurred. These are in the folder “Post Install”.
- As much of the build as will fit.

Figure 2 shows the contents of NWI for OS 9 on the G4.

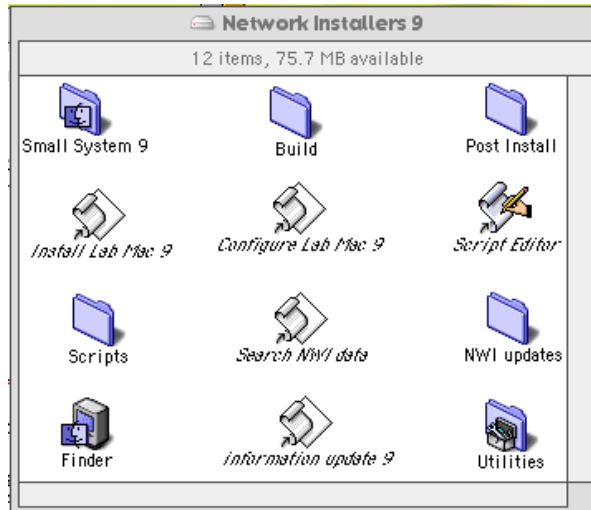


Figure 2. The contents of NWI for OS 9 on the G4

You do not need to start the Macintosh off the CD to rebuild a machine with NWI, however the rebuild seems to run quicker if the system being updated is not active. To start the update/rebuild, it is a simple matter of double clicking the "Install Lab Mac 9" alias. You will be prompted with a choice use the existing configuration data or input new, if the configuration data is complete. If it is not, you will have to enter the configuration data. Once this has been done correctly, the build and configuration run automatically.

Once the machine has been updated and configured, its start-up disk will contain the standard applications, plus an "Additional Applications" folder, where the user can install and run non standard app's, and the data folder where the user stores all their data. Figure 3 is a snapshot of the start-up disk -after NWI has finished-which is called "Workspace" on all machines.

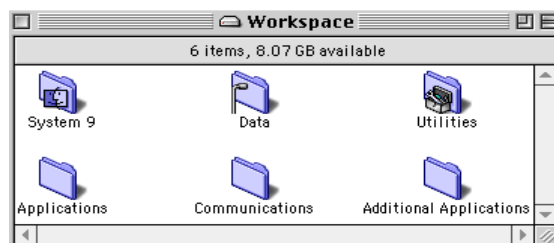


Figure 3. A snapshot of the start-up disk showing the contents of the Workspace

Major Developments

Andrew Winter at the Faculty of Architecture conceived this idea and created the system as I was introduced to it when I started in January of 1999. Andrew had a few ideas on improvements, but generally passed the project on to me, which I have found a very enjoyable and fascinating task. The following lists the different ways that the system has developed.

Password Encryption

The scripts need to use passwords to connect to remote volumes or turn off security on the machines being rebuilt. There was some concern in writing these passwords in to scripts, as there are occasions when they are visible in public places. Passwords are now encrypted and are not stored in readable format.

Common Functions

Functions such as the one to decrypt passwords are stored in one location, and thus all scripts will load the latest edition. If there are any changes to the interface of the functions, then the file name of the function file has an extension, indicating the version of interface it uses. This allows the latest of scripts to run, while older versions are still supported.

Common Variables

Variables, such as remote logins and locations, necessary disk space, etc are stored in one location in a text file on a shared server. The scripts load these variables when they start, and aspects of the install can be changed as simply as editing the text file.

Asset Management

A database is now used, with a unique record for each machine (the key field is the ethernet cards hardware address). The record contains details of:

- The owner, room number where the machine is located, department the machine belongs to, etc.
- A software system installation history; Notes such things as incomplete installs and why they failed, the time and date that the first and subsequent installs have been run, and how long particular aspects of an install took.
- The hardware attached to the machine; USB devices, extra memory, etc.
- The serial number of the machine.

Limitations and Problems

- All applications used by an apple script by the use of a “tell” statement have to be found by the script on start-up, or when loading the script into a script editor. This means that the application name must be hard-wired in the command, and variable names and paths can not be used. This causes a problem in making the configure script generic among different operating systems, as certain control panels change function or name between upgrades. For example, the “Users & Groups” control panel became incorporated in the “File Sharing” control panel in the upgrade from OS 8.6 to 9.
- Both the NWI CD’s start-up OS and the build OS had to be different between the new G4’s and the old beige G3’s. There are 2 current versions of NWI, one installs OS9 on G4’s, and the other will install OS8.6 on B&W and beige G3’s. Due to limitations in AppleScript on operating systems of version less than 8.1, NWI has not been used for those operating systems.
- AppleScripts will not allow the use of global variables, so all parameters have to be passed. This hinders modularisation
- AppleScripts are limited in size which compounds the problem of monolithic code caused by the above point
- The editor that comes with AppleScript is primitive and doesn’t allow for text based searches, and refuses to load scripts over a certain size.
- After market script editors are available that allow for the features missing in the standard editor, but scripts written in these editors may not run outside of the editor as compiled scripts. Some editors also corrupt the saved scripts so that nothing can read them.
- AppleScript does not make it easy to display messages on the screen and allow the script to continue running in the background. As a result, it is not easy to inform the user that things are going on in the background when nothing obvious is happening in the foreground.
- Serial numbers returned by the Apple System Profiler are hard-wired into the logic board. If apple change the logic board on the machine, they do not replace the serial number sticker on the box.

Further Work

It is a kind of solace that this project will never be completed, and it can be expanded in many different ways. Brainstorming new ideas and applications allows a deal of creativity, problem solving and frustration. Some ideas that we have discussed so far are

- We are currently using a 10Mbps shared network. We are considering a switched network to allow for the network traffic of rebuilding a laboratory of machines.
- The problem of network performance may also be reduced when our server moves to AppleShare IP.
- Do away with DHCP IP address allocation and give each Macintosh a unique permanent IP.
- We could have a scheduler regularly run Assimilator.
- Applications that require certain settings could be started by a script, which sets the environment before executing the application, then resets the environment when the user quits the application.
- Scripts that will un/install particular applications at the whim of the user.
- Investigate the possibility of making machines boot from a remote volume using MacOSX server Netboot manager. This would remove the need for a start up CD, and allow the rebuild of machines to be controlled from a single remote server.
- Have the database as a single file, as opposed to one file for each machine.
- Have the scripts generic over operating systems, so that the scripts themselves will determine the operating system to install by the database record, or by the use of a look-up table given the model of machine.