The following paper was originally published in the
Proceedings of the Large Installation System Administration of Windows NT Conference
Seattle, Washington, August 5–8, 1998

AutoInstall for NT: Complete NT Installation Over the Network

Robert Fulmer and Alex Levine
Lucent Technologies, Bell Labs

For more information about USENIX Association contact:

1. Phone: 510 528-8649
2. FAX: 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org/

AutoInstall for NT: Complete NT Installation Over the Network

Robert Fulmer

Alex Levine

Lucent Technologies, Bell Labs

Abstract

This paper describes how we take a bare PC and turn it into a fully functioning NT desktop machine with a standard set of applications in less than 30 minutes. In 1996 we identified OS and application loading as a significant portion of our workload. Loading NT, loading our standard suite of applications, and configuring each of them to our specifications took 4 to 8 hours. This process was prone to errors that were not detected until months later. We automated the process so that an SA with our special boot disk can do the entire process in 30 minutes (60 minutes on a slow network). We do an actual NT installation. We do not clone. With our new system, a single SA can deploy up to 20 PC's per day as opposed to just 1 or 2 per day.

1. Introduction

How many of you are frustrated by the time you waste setting up NT workstations manually when there are so many other tasks demanded of you? We have found a way to automate the OS and application installation for NT so that you can spend your valuable time doing more important things than installing machines.

2. Environment

Three years ago our environment was 90% UNIX workstations and 10% PC's running Windows 3.1. Over the last two years we have migrated to Windows NT and our PC population has grown from 10% to 40%. We currently support approximately 1800 machines of which 800 are PC's. If this trend continues, PC's running NT will dominate the desktop in our area. Anyone who has supported PC's knows this is a scary trend. PC's tend to have a higher Total Cost of Ownership (TCO) than other desktops because, while they tend to be less expensive hardware, they take more work to set up and maintain.

3. The Problem

There are never enough hours in the day. We are always looking for boring, error-prone tasks that can be automated. We began to look into areas where we could save time. One of the first things we looked at was the NT installation procedure. We had a lot of new machines to install and a lot of existing machines that did not fit our standard desktop environment or needed OS upgrades from Windows 3.1 or Windows 95 to NT. We discovered that we were spending a 1/2 day to a day installing the OS and applications as well as performing our standard customizations. In addition, the procedure was tedious and error prone, as long manual procedures tend to be, so we were also spending time cleaning up mistakes and omissions after the machines were deployed. Often a mistake would not be noticed until months later when a customer used an application for the first time. This leaves the

SA dumbfounded, "How did that go unnoticed for so long?"

4. The Options

There were a few decisions we had to make when we started the project. We first had to consider the overall method of loading the box. We briefly considered disk cloning. It has the advantage of being easier to get started. You simply build a disk the way you want it. Then use that as your master and clone the disk via disk imaging software or disk cloning hardware. The cloning hardware had some limitations concerning disk size. The target disk had to be at least as big as the original and you would only get a partition as big as the one on the original disk. When we had to change packages we had the choice of doing an upgrade (not always the cleanest way) or reinstalling the master disk from scratch. We also had to worry about duplicate SID's although there are vendors out there who say they have solved this problem. (see www.sysinternals.com) This procedure only worked correctly if the hardware on the target machine was identical to the hardware on the source machine. Unfortunately we have a lot of different hardware coming in and it usually comes in a few machines at a time. Using disk cloning would have meant rebuilding the disk for each new hardware configuration that came in the door. The biggest concerns here were the type of disk controller and the motherboard architecture. We needed something that would detect hardware on the fly during the installation so that we do not need a separate disk for each hardware platform. The NT unattended installation allowed us to have this feature. We could have one distribution for all of our various types of hardware. The only requirement was that the network card be supported by the NT installation. We could make changes to the distribution without having to rebuild it from scratch each time. Adding applications and applying bug fixes was easier and cleaner. We didn't have to worry about duplicate SID's either. Unsupported network cards were not as big a problem as one would expect either. We discovered ways to add support for new network cards to the original distribution.

Once we had decided which overall method we wanted to use we needed to decide which method to use to start the installation. We considered boot proms for the network cards in our PC's, but this would have added $50-$100 to the cost of each machine and would have meant opening each machine to install the proms. Also the installation process requires several reboots and only the first boot needs to be from the net. There is no way to tell the machine "only boot from the net this time and then revert back to default next boot" as you can with Sun workstations. We also considered Linux boot disks to bootstrap the installation but lacked the knowledge to make it work. We were also hampered by the fact that Linux distributions often lagged behind on driver support for new network cards. We considered putting the distribution on CD. It would have made loads faster without impacting the network but it would have meant burning a new CD for each SA every time we fixed a bug or added new features to the distribution.

We decided to use an MSDOS Lan Manager client on a MSDOS boot disk. The Lan Manager client was small enough to fit on one floppy. The client's configuration files were plain text. And the client was not started in the config.sys file. It was started in the autoexec.bat file. This meant that we could write a program that asked the appropriate configuration questions and modified the client's configuration files. This allowed us to configure the client and then start it with the correct network card and IP settings.

5. The Solution

The manual procedure took 4-8 hours and

required that the SA baby-sit the machine and answer questions throughout the procedure. The automated procedure we designed takes at most 60 minutes (30 minutes on a fast network). The SA need only be present for the first 5-10 minutes to answer questions and the last 10 minutes to install the latest sound drivers and perform a few local customizations that we haven't automated yet.

The procedure for an Autoinstall is basically two steps. Boot from a Lan Manager floppy and answer some questions and remove the floppy once the installation has begun. The SA is then free to go off and do other tasks while the process continues. When the process is done, a few drivers are installed and the machine is ready to use.

6. How It Works, An Overview

The MS-DOS boot disk permits the SA to partition the hard disk. It then asks the SA a few questions about what Ethernet NIC is being used and the host's IP configuration (IP address, netmask, etc.). At that point, the workstation connects to an NT server and runs "part 2" of the script that is on the network drive. This program gives a menu of applications that can be installed; the default is "all". The SA chooses which applications to install. Then SA removes the floppy and leaves until the procedure is done.

This second script continues by formatting the hard disk, bringing down the appropriate distributions from the server, and running the "winnt.exe" installation program that installs the OS.

Applications are loaded last. As part of the OS installation, Data is put into the registry that causes the machine to login automatically (autologon) and run a command of our choosing by putting the command line in the RunOnce registry entry. After the OS is completely installed, control is returned to our final script. This last script installs our applications and performs local customizations. Each application has a different way to automate its installation. In general, most can be provided with an installation script file that answers the questions.

Once the process is complete, the machine is fully usable. However, with our current configuration the machine does not have the latest driver for the audio card installed.

7. The Details

The installation has four phases: the initial DOS portion of the installation, the text mode phase, the initial GUI mode phase, and the autologon or second GUI phase. The initial DOS phase is where the SA boots from the floppy and fills out all of the information that needed to install the system and then kick off the installation. The initial installation then copies all of the files needed to finish the installation and reboots. During the text mode phase the installation begins copying files to their final location and does most of its hardware detection. During the initial GUI mode phase all of the rest of the OS files are copied into place. The network card is detected and the NT networking gets started for the first time. This is the phase where the machine joins the domain. This is also the phase where the video card is detected and the drivers for it are installed. At the end of this phase the patches are applied to the OS and the settings are put in place to start the second GUI phase of the installation. During the second GUI phase all of the applications are installed and all local customizations are applied.

7.1. The Boot Floppy

There are three major pieces that are key to making this work: the boot floppy, the OS distribution, and packaging the applications.

The main idea here is to collect all of the data we need up front and then package everything in such a way that no questions need to be answered during the installation. The floppy and the initial scripts become the most important part of this procedure. The floppy needs to ask for enough information to get the machine on the net and logged into the server. The program that collects this information then rewrites the configuration files for the Lan Manager client to use, starts the client and logs into the network. Once the system is logged into the server we can run scripts or programs that ask for all of the rest of the information we need to finish the installation.

The biggest problem we had with the floppy was fitting the Lan Manager client, the programs and drivers on one floppy. We wanted support for multiple network cards on one disk but the original floppy, which was based on Windows 95's DOS, didn't have enough space to fit everything we needed. We managed to fit it all by doing three things. First, we switched from Windows 95 to an MS-DOS 6.22 boot floppy. It had smaller boot files. Second, we switched from a BASIC program (which required a 200-KB interpreter) to C++ program. Third, we started using a ramdisk and put all of the files we could in a zip file which gets uncompressed to the ramdisk. These changes gave us more space on the floppy and gave us increased performance. When we were done with these changes we had enough uncompressed files on the floppy to start the ramdisk and run the one network driver that is started from config.sys. All the initial autoexec.bat did was set a variable so we knew where the ramdisk was and then uncompress the zipfile to the ramdisk. The autoexec.bat then called a batch file on the ramdisk that ran our C++ program, started the Lan Manager client and logged on to the network. In other words, once the batch file in the ramdisk ran, we didn't even need to have the floppy in the drive anymore.

7.2. The Distribution

The second important element in the process is the OS distribution. Scripts are needed to collect whatever information wasn't collected from the floppy portion of the installation and then start the OS installation. The key to this portion of the setup is choosing the right settings in your unattended.txt file. The unattend.txt file was a file in which we could specify all of the settings for the machine that were needed to install NT without having to answer any questions. We chose certain settings that we wanted to be the same for all of our workstations and then used our initial questions to fill in the settings that we wanted to set on a per workstation setting.

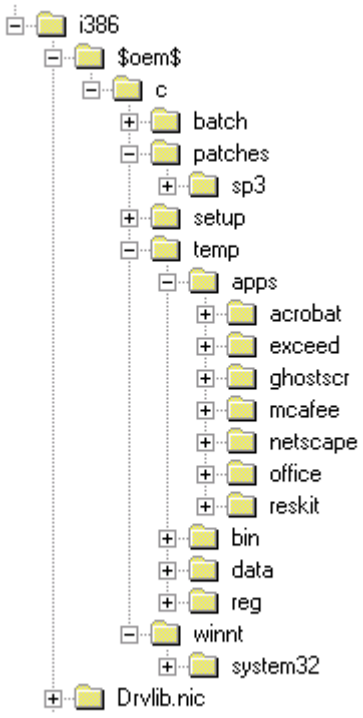
The common settings included things like installation type, CD-key, file system conversion, file system expansion, local administrator password skip, End User License Agreement (EULA) skip, and others. We wanted all of our workstations to run on NTFS so we chose to make NTFS conversion a global setting. We also chose to expand the file system to fill the disk. We chose to leave the local administrator password blank so that it would not be asked for in the middle of the installation. We set the password after the installation was done. We chose to skip the EULA for the same reason. We set the license key in the default unattended.txt because we have a site license for NT and so have one universal license key. Most people will probably want to ask for the license key as part of the initial installation. Because we are using the cmdlines.txt later in the installation and are extending the file system to fill the disk, we need to set the installation type to OEM. We also set the default display settings here. Since we wanted every machine to be part of a domain, we set the flag to join the domain. Which domain to join is set on a per machine basis since we have multiple domains. The basic idea is to create a default unattended file and put our network-wide

settings there. We then use that file as a template to create a specific file for each machine. We thought about using Universal Difference Files (UDF's). These are files that let you specify settings that override the unattened.txt file. UDF's could have been used to specify machine specific settings but some of the settings that we wanted to be able to set on a per machine basis could not be included in a UDF. We quickly discarded the idea and wrote a script that would generate a machine specific unattended.txt file from the information gathered and the template unattended.txt file.

We ran into a couple of interesting problems worth noting. One was a bug which would make the machine pause after the file system conversion if the file system was being expanded. This was done by design, but we don't know why. There is a patch on Microsoft's website that must be applied to the distribution and an extra setting that must be added to the unattended file to fix this bug. You can find this patch and information about the new setting at ftp://ftp.microsoft.com/bussys/winnt/winnt-public/fixes/usa/nt40/hotfixes-postSP2/setupdd-fix.

We also discovered that the environment that programs run in during the first GUI portion of the installation is not what one is normally accustomed to getting. We could not make any assumptions about PATH setting and other such environment settings. We also could not make any assumptions about what services were installed and running. As a result, some of the things we tried to do did not work as expected. We ended up making sure that we used full paths and keeping the cmdlines.txt file as simple as possible. All we did there was patch the OS and set up the automatic login feature and the account that we use to do the installation of all the apps and local customizations.

We tried to minimize the size of our distribution, so the only directories that we copied from the CD to our installation server were the i386 directory and the driverlib.nic directory under the i386 directory. If we had wanted to install any of the personal web server features we would have had to copy the entire i386 directory (including all of its subdirectories). We also uncompressed the files in our distribution because we found that the installation went faster if the target machine did not have to uncompress the files as well as copying them. There are also other modifications that we ended up making later that required that INF files be uncompressed so that they could be modified easily.



View of Our Current Distribution Structure

When doing an OEM installation the installation process looks for a directory called OEM in the i386 directory and copies anything in that directory to the hard disk of the target computer during the initial phase of the installation. If you create a directory called "C" under OEM then anything under that

tree will be copied to the "C" drive using the directory tree created under the OEM tree. We used this feature to copy the service pack and hotfixes over to the target machine along with batch files and executables which were not part of the original distribution that we needed during the installation. The installation process also looks for a file called cmdlines.txt in the root of the OEM directory and runs any commands found there at the end of the first GUI phase. We use this file to run the batch file that installs the patches and hotfixes. We also set up the automatic logon here and the account that we used to install all of the applications and local customizations and to set up the RunOnce registry entry that starts the application installation on the last boot.

All of our applications are part of the OEM tree which means that all of the applications get copied automatically during the initial phase of the installation. We set up a separate tree for all of the applications that we had packaged. We set it up so that each application had an appropriately named subdirectory and that one of the files in that subdirectory would be a script file which would run the package for the application and install any local customizations. One of the pieces of information we ask at the beginning of the installation was which applications the installer wants on the machine. The list of available applications was generated from the list of subdirectories in the application tree on the distribution. When the installer was done selecting applications, a master script was generated that ran the installation scripts for all of the selected applications.

We could add an application to the distribution in three simple steps. First, package the application. Second, write a script that will install it completely. Third, place it in a subdirectory with the appropriate name in the applications directory tree.

The RunOnce registry setting was the heart of the second GUI phase of the installation. Whatever command we put here was run during the final GUI phase. We wanted this to be a script that performed all application installation and local customizations. This script ended up being fairly short. We tried to package as much as we could into application packages so that when we selected to install nothing, we got a completely clean machine. We wanted this mostly for testing purposes and for packaging applications using SMS Installer. This script did two really important things. First, the script called the application installation script that we generated when the applications were selected in the beginning. Second, the script saved the resulting user profile to the machine default user profile. This assured that any new user that logged into the machine would get the settings needed to be able to run the applications that were installed on the machine.

The user profile is a directory structure plus a registry hive, a binary file which stores many of the per user application settings. The directory structure can simply be copied to the default user profile. Although the registry hive is just a file in the user profile directory, it is locked by the system and cannot be copied. Therefore, we needed to do a registry dump on the autologon user’s registry key and then copy the resulting file into the default user profile. There were tools in the resource kit that allowed us to dump a registry hive. However we could only get the username from the user’s environment. The registry key for the user was named by SID rather than user name. We needed to write a utility to get the SID of the currently logged on user. We wrote a wrapper program that called a resource kit tool with the right parameters and parsed the output for us.

7.3. Packaging the Applications

The third key element to this procedure was

packaging the applications so that they would not stop and ask any questions. We discovered four different ways to accomplish this: the InstallShield method, the STF file method, the vendor specific method and SMS Installer method.

7.3.1. The InstallShield Method

We found that a large majority of applications these days are using a setup program called InstallShield. This type of installation program was easy to automate. The setup routine could be supplied with the -r switch that causes it to record all of the answers given during the installation in a log file. We could then copy this log file to the application source directory and use the -s switch that specifies a silent mode installation. This mode reads all the answers to all the questions from the log file that was just created and installs the application without asking any questions. We used this method on McAfee Antivirus for NT.

7.3.2. The STF file method

We found a method that was Microsoft specific and worked on most of their big packages, such as Microsoft Office 95, Microsoft Office 97, and the resource kits. During the installation these applications generated an STF file that was designed to record the installation so that the application could be reinstalled or uninstalled partially or completely at a later date. With the saved the STF file we could give switches to the setup program telling it to reinstall everything and then specify the STF file on the command line. When we did this we found that the setup program repeated the installation that created the STF file. For example, if the STF file was c:\office.stf the command line would be setup /r /t c:\office.stf /q1. The /r switch told the setup program to reinstall everything. The /t switch allowed us to specify the path to the STF. We found that this path had to be an absolute path. The /q1 switch told the setup program to run installation in quiet mode 1. This mode displayed the progress but did not display the dialog box at the end that announced that the installation was successful. We found that the installation either moves or deletes the STF that supplied on the command line. So make sure that a copy of this file, not the original, is used during the package installation.

7.3.3. The Vendor Specific Method

Some vendors have their own setup programs or require hacks to make them work. Netscape appeared to use InstallShield but the silent mode installations did not work. We had to modify their setup.ini to change the default installation path to our standard and add other settings to prevent the installation from asking questions. In a case like this, we found that contacting the vendor was the best thing to do. In most cases they were helpful and explained how to automate their installation process. In cases where the vendor could not or would not help the final method seemed to work well.

7.3.4. SMS Installer Method

Microsoft provided a tool called SMS Installer for sites that have Systems Management Server installed. This tool watches the installation and then finds all of the files, services, registry entries, and shortcuts that are installed during the installation and scripts and packages those changes into a self-extracting archive. When this archive is run it extracts itself and performs an unattended installation of the application that was just packaged. Sometimes SMS Installer missed files and registry settings. These registry entries and files could be added to the package later through the SMS Installer's GUI. We used this to package Hummingbird's eXceed. For those sites that don't have SMS installed sysdiff is an adequate substitute for the SMS Installer. The only caveat is that the sysdiff

package cannot be modified. If sysdiff misses files or registry entries they must be added later. Missing files must be collected and copied manually. Registry entries must be put in a registry script file and incorporated into the registry manually using regedit.exe. Another tool to consider is Seagate's WinINSTALL. It's description made it sound a lot like SMS Installer. We didn't look into it because we had several solutions that were free.

7.4. Adding Support for New Hardware

One of the things that we are still working on (as we write this paper) is adding new drivers to the distribution. We found ways to add PCI network cards and SCSI drivers. We think we found a working solution for video cards but are still testing it. Microsoft did provide ways of specifying OEM hardware in the unattended.txt file but using this method would have meant asking yet another set of questions at the beginning of the installation. We chose to try to integrate the drivers into the actual distribution so that they are auto-detected during the installation. Microsoft did NOT SUPPORT this solution. They would not help us when we had problems making it work. We found that most hardware vendors tried their best to be helpful and provided some insight into what was happening during the installation's two hardware detection phases. After getting information from the vendors and poking around in the setup files, we found we were able to figure out what was going on. Once we understood the driver installation process it became much easier to find a procedure to add each type of driver to the original NT distribution. The solution we chose took some extra work but seemed to boil down to a set modifications to INF files, script files that the NT installation runs, and in some cases modifications to the initial system registry hive in the distribution.

8. Future Enhancements

We have plans for several future enhancements to AutoInstall. We are working on a way to install drivers that the NT installation does not auto-detect. We think we may be able to use SMS Installer to create a package for each of them so that they can be installed automatically. We are planning on moving the applications tree out of the OEM directory tree so they can be copied selectively. This way only the applications chosen by the installer will be copied. We are also planning on adding DHCP support so that our installers don't have to answer as many questions as long as a DHCP server has settings for the client computer.

9. Suggestions for Vendors

There are many things that vendors could do to make this automation process easier. A set of tools to add new drivers to the NT distribution would be nice. A standard method of packaging applications would also be a great help. It would also be really nice if applications were less intrusive in the operating system. Having an application install extra files in the system directories does not appeal to us. If the applications were less intrusive, it would be easier to centralize them. NT workstations could become more generic and easier to setup and to maintain. Another nice feature would be a "smart" application which knows what settings it needs to run and adds reasonable defaults to the user's profile and the local machine if they don't exist. They could at least give a reasonable error message upon startup, stating which settings are missing on the current machine so the user can fix them or call an SA if necessary.

10. Findings

We have a similar system for our Sun Solaris (UNIX) hosts so we knew what benefits to expect. As we anticipated, the result was a fundamental change in the way we install and

upgrade hardware on our network. Previously SA's spent most of their day doing installations. Now that has become a minor part of their job. They spend their time focused on other tasks such as attending to customer requests and trouble-shooting.

We had expected that it would be useful to have every detail of every machine be configured the same way. However, we underestimated how useful AutoInstall would be to our front-line support personnel. Our front-line support had been spending a large amount of their time with issues from customers with newly installed machines that could be traced to human error during the manual installation process. This automated installation process nearly eliminated these problems. As a result, our front-line support personnel now can focus on deeper problems. This was a benefit we had not seen in our Sun Solaris (UNIX) environment. On those systems, applications could be centrally installed, unlike NT where applications and their configurations are loaded locally.

Our customers are extremely satisfied with the accuracy of our installations and our ability to roll out machines in higher volume.

11. Conclusions

Automating OS and application installation was difficult to set up initially because we were breaking new ground. Now that the process is in place, it should become more commonplace. The result of this automation has been a productivity boost on three levels.

(1) Our installers can install more machines per week error free. In fact, for the first time, they can install more than one or two machines per day.

(2) Our front-line support personnel no longer receive problem reports from customers with machines that were misconfigured due to human error.

(3) Our customers are happier with their environment because machines are more likely to be installed on time and the machine they receive is configured correctly.

Obviously, the happiness of our customers is the most important gain.

12. Availability of Source Code, Etc.

Contact the author's for code availability.

Robert Fulmer rjf@research.bell-labs.com

Alex Levine alyank@research.bell-labs.com