Automated Systems Setup

By

Eric Tribbe

Submitted to
the Faculty of the Information Technology Program
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

University of Cincinnati
College of Applied Science

June 2008

Automated Systems Center

by

Eric Tribbe

Submitted to
the Faculty of the Information Technology Program
in Partial Fulfillment of the Requirements
for
the Degree of Bachelor of Science
in Information Technology

_____	_____
Eric Tribbe	Date
_____	_____
Mark Stockman, Faculty Advisor	Date
_____	_____
Hazem Said, Ph.D. Department Head	Date

Acknowledgements

I would like to give a special thanks to Dominic Ferreri and Tim McDade along with everyone else at the Administration and Finance IT Department. They have provided me with a good opportunity to work on a project for them. I would like to also thank them for providing help and guidance as I worked through my project.

Table of Contents

List of Figures

Abstract

Automated Systems Setup is the evaluation and setup of a software deployment product called Thinstall. The Administration and Finance IT Department of the University of Cincinnati needed to set up a software deployment solution to easily deploy and manage its wide variety of software applications on their network. The current method of deploying software on new or rebuilt computer systems is too slow. After implementing the Thinstall solution onto the Administration and Finance IT Department's network, setup of new computers or rebuilt computer will save time and ultimately save money.

Thinstall is a virtual software builder that is capable of packaging many different software applications into a simple virtualized package. These packages then may be easily deployed using login scripts in their Active Directory domain. This solution takes advantage of the already established groups and organizational units in their active directory domain to allow for different software configurations. Once deployed, each virtualized application package installs onto a client computer and runs exactly like a normal installed application. Each application can be easily updated to newer versions or can be patched with the latest updates then repackaged and redeployed to each client computer.

Automated Systems Setup

1. Statement of the Problem

The Administration and Finance IT department manages about 800 users along with 750 computers campus wide. It manages desktop support for multiple departments within the university. Each year the Administration and Finance IT department orders and receives, in bulk, about 125 to 150 new computer systems in a year. It recently implemented a four year rotation with all computer systems that it manages. Using this rotation it found that operating costs can be reduced. For each computer system purchased they add a four year warranty which allows no additional costs to hardware if there is a system failure. As well as the new computers being ordered each year, a number of computers need to be swapped out and rebuilt several times over their 4 year life cycle due to hardware and or software failures.

The problem that the department is facing is the software management with each of the computers. When a computer system gets replaced, the software configuration needs to be copied to the new computer. If the computer breaks down suddenly, a replacement computer needs to be put in place and a copy of the user's software configuration needs to be loaded onto the swapped computer.

Another need of the Administration and Finance IT Department is the need to track software assets that are installed on each of the computer systems. According to Tim Mcdade, their last audit showed that 10 copies of Microsoft Visio were installed on their computer systems, but he knows for a fact that there are several more copies that are installed on more computer systems (6).

2. Description of Solution

The solution to this issue was to set up an automated system setup service. This service included a software management package which has the ability to deploy software to a computer system from a centralized management console to computer systems campus wide. It also had to have the ability to track, and audit the deployed software and purchased software licenses so that software compliance can be made. Also this solution integrated with Active Directory so that the Administration and Finance IT Department could deploy certain software packages based on where the user is located in the Active Directory structure.

2.1 Virtualized Software

Virtualized software is a new concept in Information Technology. Virtualizing software essentially separates the operating system from the software. With virtualizing software no changes are made to the operating system. Normal programs can modify a system's registry or operating system files. Virtualized software puts a virtual layer between the operating system and the software (2). Virtualizing software also has the ability to virtually eliminate conflicts between the different software available. In the past when some applications were installed locally on a computer, a file conflict could occur or one application could overwrite a particular operating system file with an older version while another application needed the newer version. With virtual software these conflicts are eliminated because each application is run in its own environment.

Another aspect of virtualized software is that different versions of the same application can be run at the same time. One example is a help desk running Microsoft Office 2003 and Microsoft Office 2007 on the same computer system.

Virtual software has the ability to inter-operate between different operating systems. A computer system that is running Windows XP can be upgraded easily to Windows Vista without the administrator having to worry if the software would be compatible with the new version of Windows (2).

2.2 Thinstall

The main solution that the Administration and Finance IT department was interested in is the Thinstall solution. This solution is appealing because it requires no additional infrastructure to be built. Thinstall is an agent-less, server-less, virtual software solution that takes advantage of a custom-built login script deployed through Active Directory (7). Thinstall creates a virtualized software package that is built into a single executable file

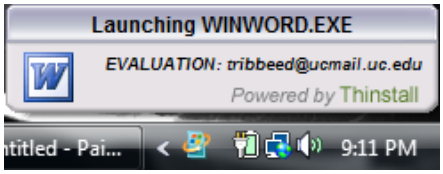

Figure 1 Thinstall VOS

Thinstall also packages its virtual layer operating system into every package. What this means is that no additional software needs to be installed in-order for the virtual software to run properly. Figure 1 shows the virtual operating system at work when a virtual application runs. Having the virtual software operating system built into each application allows for the applications to be very versatile. Each of applications can be placed onto a USB flash drive for portability. This allows a user to carry around all the applications easily and have all the user's customized settings for each application (7). The Thinstall solution is a very non-network

intrusive application. Thinstall can be considered a virtual software builder but the way Thinstall builds the packages allow the program to be very versatile.

Thinstall also has the ability to stream applications from a network location. What this means is for example Microsoft Office 2007 can be placed on a central network server and each client can run the application from that location without having the application package on each of the client computers. Thinstall will only transfer the files and settings necessary to run the application and stream additional files and settings when the application needs it (1).

3. User profiles

3.1 Administrators

The Administrators will have access to the package building tools that are available for each solution. The Administrator will also manage the active directory groups and which members belong to each application group.

3.2 End User

The end user will be where the applications are deployed to. The standard setup for the end-user will be the deployment agent and none of the package building tools.

4. Design Protocols

The Login Script code was written in VBScript. VBScript is a lightweight version of Visual Basic that is used to create dynamic Web-pages and computer scripts (5). VBScript was used to gather certain information about the computer that the script is executing on such as installed software, and the currently logged on user. VBScript is also used to execute install commands to install the selected software packages onto the client computer and send a list of installed

software to a database on a Microsoft SQL server. The overall process for the script is that the script will get the name of which user is logged onto the computer. Then it will get a list of all the Thinstall groups the user belongs too and then download the assigned application to the computer. After the files are downloaded then the script will register the application with the user and uninstall any software from the computer that the user is not assigned to (See Figure 2).

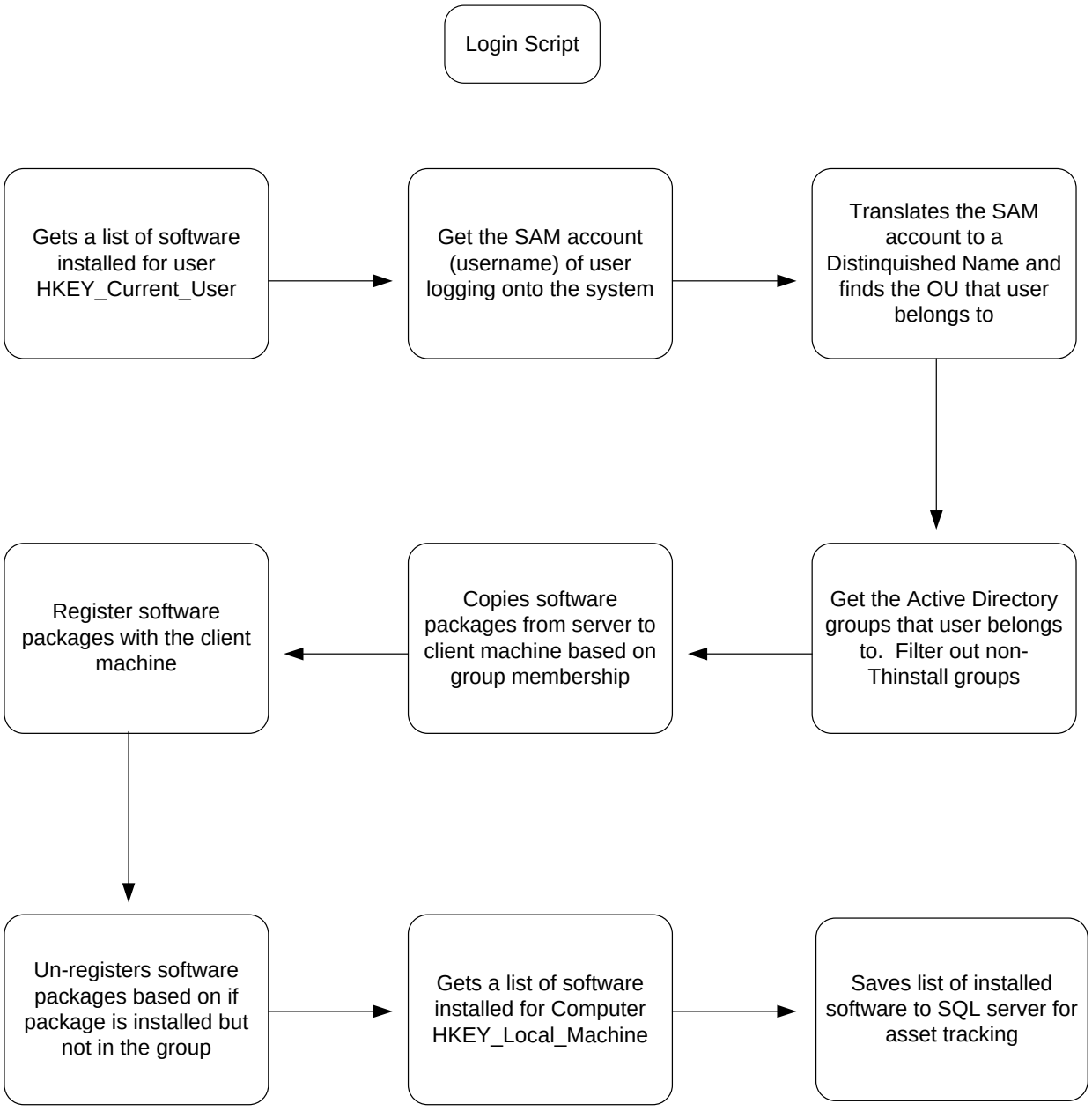

Figure 2 Login Script Flow Chart

5. Deliverables

The following is a list of deliverables for the project. The overall task for the project was to evaluate and set up a software delivery solution for the Administration and Finance IT Department at the University of Cincinnati.

1. Choose a software deployment system that fits the needs of the Administration and Finance IT Department.
2. Provide a working prototype of the chosen system
3. Provide documentation on the set up of the system
4. Implement system on to network and configure client computers.

6. Design and Development

6.1 Budget

The expenses accrued during the research and development of the project was nothing. All of the items are free due to the use of existing hardware and trial software. Labor costs for the project are figured into the budget but are not being charged. All labor for the project is strictly volunteer work (See figure 3. page 7).

Expenses			
Item	**Description**	**Retail Cost**	**Actual Cost**
Server	Provided by AFIT	$1,080.00	$0.00
VMware Server	Free	0.00	0.00
Windows Server 2003 R2	Provided through MSDN software	999.00	0.00
Microsoft System Center Configuration Manager	Trial Software	0.00	0.00
Novell Zenworks	Trial Software	0.00	0.00
Altiris	Trial Software	0.00	0.00
Thinstall	Trial Software	0.00	0.00
Labor @ $40	150 hours	6,000.00	0
	Total:	**$8079.00**	**$0.00**

Figure 3 Development Budget

6.2 Timeline

The timeline for the Senior Design project is split between three quarters. For each quarter several milestones were accomplished. Figure 4 shows the milestones for the project and each quarter's accomplishments are described in detail below.

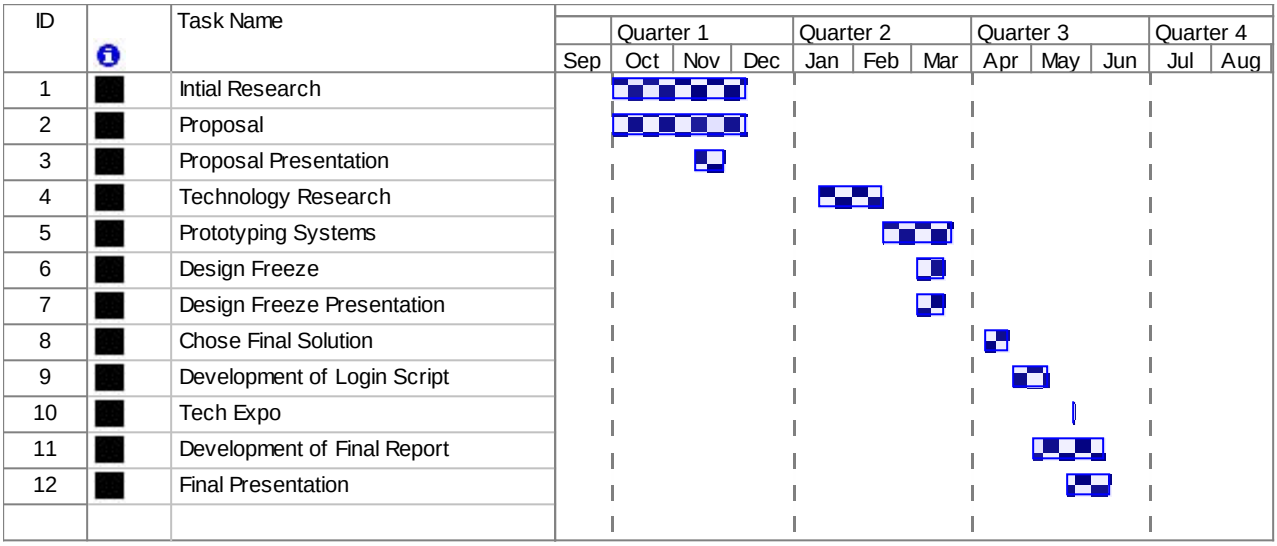

Figure 4 Development Timeline

6.2.1 Senior Design I Accomplishments

The following are the accomplishments made during Senior Design I

- Gathered Requirements of project
- Increased knowledge of software and operating system deployment solutions
- Research deployment solutions
- Developed proposal and presentation

Senior Design I was the research phase of the project. During this time research was done on the concept of deploying software from a central management point. The different types of technologies that was available at the time was explored. These technologies included

the Microsoft Systems Configuration Center, Novell's Zenworks, Symantec's Altiris Solution, and Numara's Deploy IT solution.

During this time appointments were made with the Administration and Finance IT department several times to research their needs and what they wanted out of the project. After gathering the requirements some initial testing was done to find out if the products fit the requirements of the Administration and Finance IT Department.

6.2.2 Senior Design II Accomplishments

The following are the accomplishments made during Senior Design II

- Set up test systems of each product
- Eliminated several of the products for use
- Discovered and tested another product that was available
- Chose two of the products for final use
- Documented the design freeze and presentation

Senior Design II consisted of a more in-depth testing of each of the possible solutions. During this time test systems were set up for each product. At that time it was determined that several of the solutions were not going to fit the requirements giving by the Administration and Finance IT Department. Both the Microsoft System Configuration Center and the Novell's Zenworks were determined to have more features than needed along with not enough features to support the requirements of the project. During the research a new solution was discovered. The new solution, Thinstall, was quickly tested and determined that it would fit the requirements of the Administration and Finance IT Departments. Along with the Thinstall

solution Symantec's Altiris was also determined to fit the requirements stated by the Administration and Finance IT Department.

6.2.3 Senior Design III Accomplishments

The following are the accomplishments made during Senior Design III.

- Chose Thinstall as final solution
- Scripted login script
- Tested final design
- Wrote final paper
- Presented final solution

A few weeks into Senior Design III a choice was made to develop the Thinstall solution over the Altiris Solution. It was determined that there was a need to develop a login script to handle the deployment and management of software applications. The development of the login script took around three weeks to write. During this time testing was done to make sure that the script would work correctly on the Administration and Finance IT's domain.

7. Hardware and Software

For the evaluation and testing of each of the possible product solutions there were several hardware and software components needed.

7.1 Hardware

- Computer system with at minimum two gigabytes of memory: For testing of each deployment solution.

7.2 Software

- Windows Server 2003: Required to load various server components for each deployment solution
- Windows XP Professional: Required for client testing for each deployment solution
- VMware Server: Used to test each deployment solution in a virtual environment
- Microsoft SQL Server: Used to test the software asset tracking portion of the script.
- Notepad++ : Used to write the login script

8. Proof of Design

8.1 Building Packages

In order to be able to use Thinstalled packages, a software application needs to be captured from a clean computer system. To capture an application, Thinstall goes through process that captures all the files that were created or modified during the install and packages them together. The Thinstall Setup capture program is a very user friendly application with a very easy to use user interface (See figure 5).

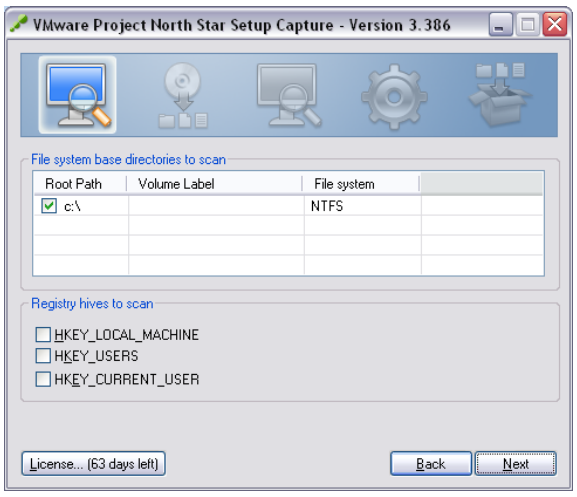

Figure 5 Thinstall: Scan Settings

8.1.1 Pre-Install Scan

The first step in the process is the pre-install scan. The pre-install scan makes a list of all the files currently on the hard drive and catalogs locations and other file attributes. The pre-install scan also scans and documents the system's registry. Figure 6 shows the pre-install scan in progress.

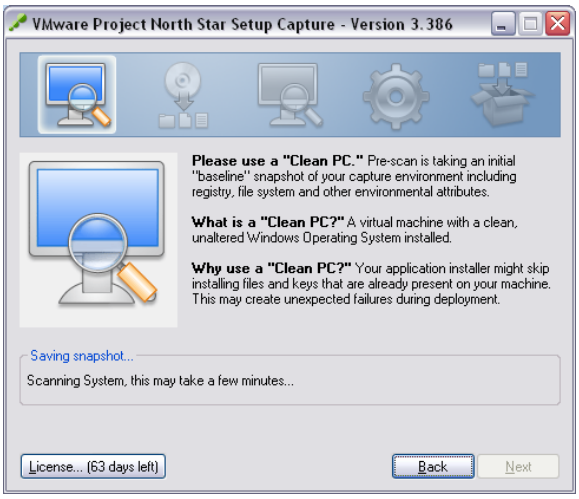

Figure 6 Thinstall: Pre-install Scan

After the preinstall scan takes place the installation of the software application would take place. There are no special steps that need to be taken to install the application. Installation is done by following the instructions from the manufacturer of the application.

8.1.2 Post-Install Scan

After the software application is installed, a post install scan occurs. The post-install scan does the same scan as the pre-install scan does by scanning both the file systems and the registry but this time catalogs all the files created by installing the software application.

The next step in the Thinstall process is to compare the pre-install scan to the post install scan and look for any differences in the file system and the registry. Once the new files and the registry keys are discovered by Thinstall, the user can then select the primary data container and the entry points into the Thinstalled program.

8.1.3 Data Container

The primary data container is the file that contains all the files captured by the Thinstall capture program. It holds the Virtual Operating system which allows a user to access both the virtual file system and the virtual registry. The entry points are the executable files that can be set up to access the primary data container. After the post-install scan runs the Thinstall Capture program displays the list of the executables found and checkboxes next to the names Below the list is a box to select the primary data container for the program (See Figure 7).

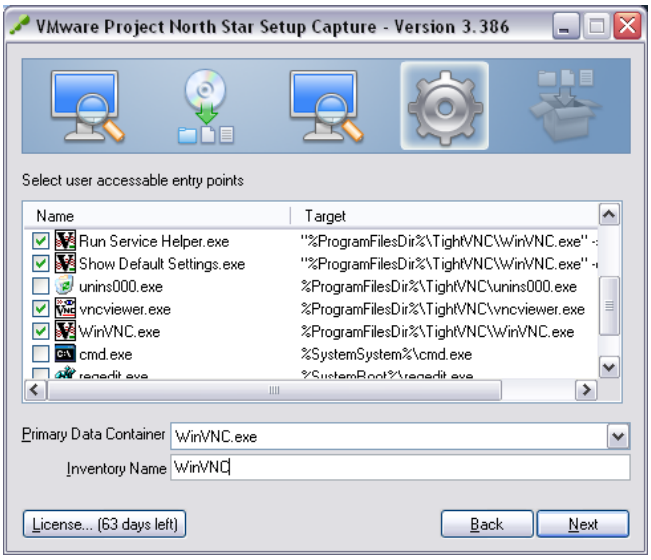

Figure 7 Thinstall: Entry Points

8.1.4 Group Permissions and Sandbox

Once the primary data container and the entry points are set, Thinstall then allows the user to set different configuration settings. The different settings are: setting Active Directory groups; setting the sandbox to a different location; and application sync. The first setting is an option to permit only certain active directory groups to access the package (3). By default everyone is allowed to access and run the Thinstalled program, but settings can be made that only a certain group such as administrators are only allowed to access and run the program (See Figure 8. page 14). This is an important feature that is later used to control users that may have access to the Thinstalled packages but is not necessarily assigned to run the application.

The sandbox is the location where any changes to the Thinstalled application are saved. The original Thinstalled application files are never changed so any changes to the application or any configuration settings to the program are saved. The configuration setting for the sandbox mode is where the sandbox folder is located. By default the sandbox folder is in the *C:/Documents and Settings/<username>/Application Data/Thinstall,* but the location can be changed depending on what the program is going to be used for. The Sandbox folder can be stored in the same directory as the application folder (See Figure 8. page 15). What this allows users to do is to make the application mobile. Someone can load the application onto a USB flash drive so that when he/she plugs the flash drive into any computer and runs the application they have access to the program and any special configuration settings he/she made to that program. The sandbox folder can also be located on a network drive to allow multiple users access the program at the same time and to get the same configuration settings as everyone else (3).

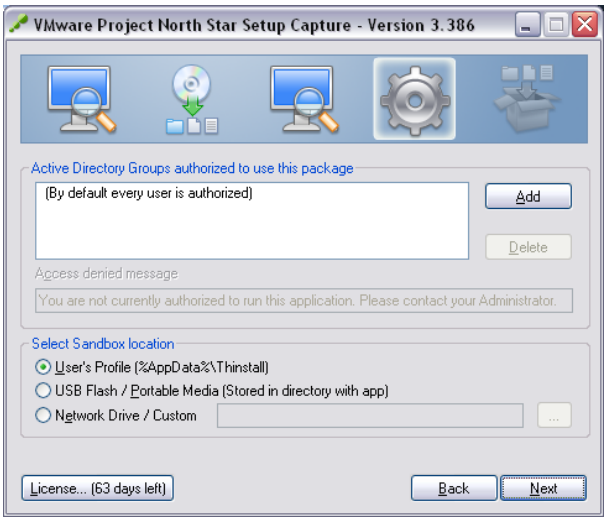

Figure 8 Thinstall: Sandbox Selection

8.1.5 Application Sync

The Application Sync setting allows for administrators to set a secure update path to push out updates to any remote computers running the software. The Application Sync feature in Thinstall is set up to update packages once they are deployed to user. Every time the package is ran the application will check the URL specified to see if any updates are available and download them if necessary (See Figure 9).

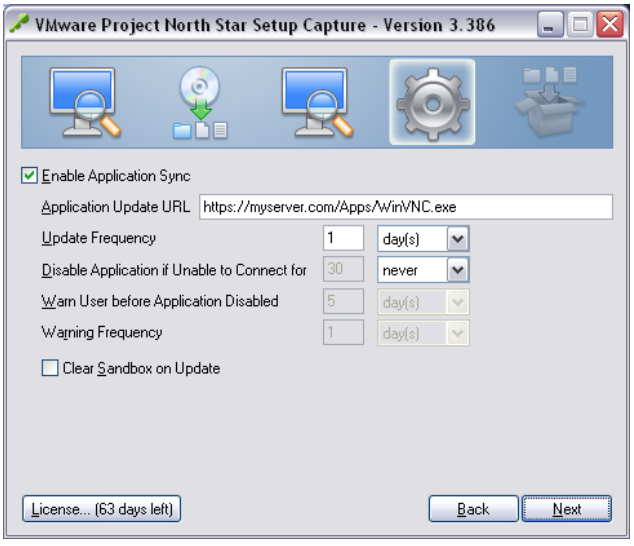

Figure 9 Thinstall Application Sync

8.1.6 Package Building

Once any optional settings are made the Thinstall application builder then creates a directory in *C:/Program Files/Thinstall.VS/*Captures and copies any of the application files to that folder so that a final Thinstall package can be built. The Thinstall Capture Program automatically moves the discovered application files from the pre and post-install scans earlier in the program (See Figure 10).

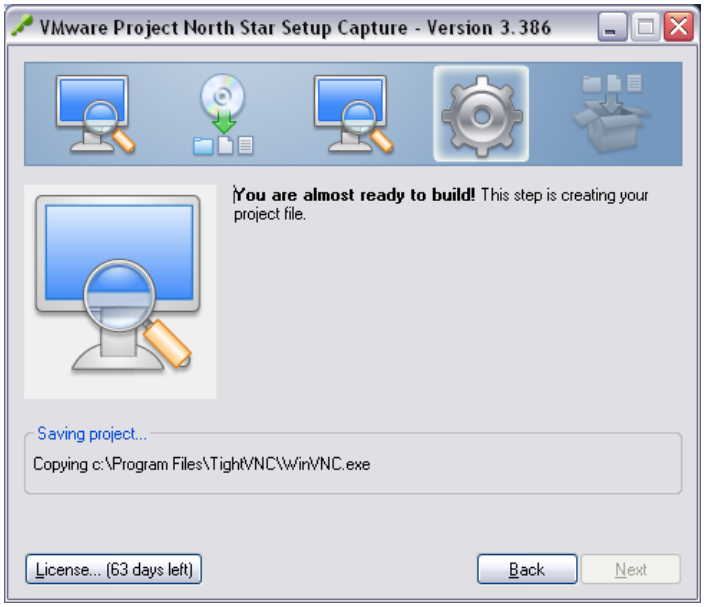

Figure 10 Thinstall: Post-install Scan

An important feature that Thinstall uses when capturing and building a package is its use of environmental variables for its folder locations. Environmental variables are basically shortcuts for an Operating System. For example, instead of a program using *"C:/Documents and Settings/MyUsername/Desktop"* to access a user's desktop the program would just have to user *"%Desktop%"* to get to the same location. Thinstall takes advantage of all of the common environmental variables that are already in place by default. When it creates the folders for the

files it found during the setup capture process and creates the folder paths based on the environmental variables rather than the absolute path of the file (See Figure 11).

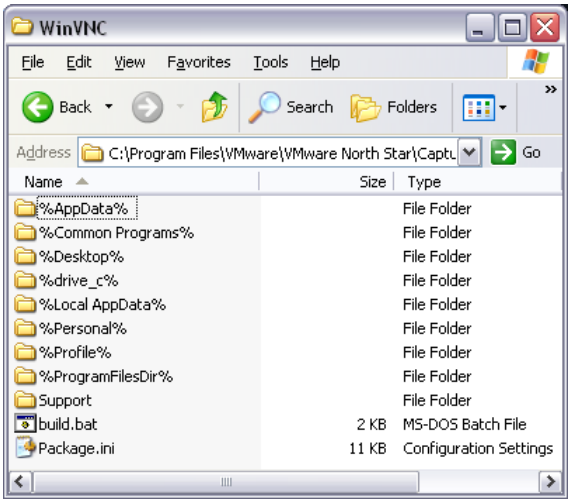

Figure 11 Thinstall: Capture Program

After the setup capture program is finished the next step is to build the actual Thinstall package for the software application that the user wants built. To build the Thinstall package a batch file, named build.bat, along with the configuration ini file, named package.ini, is located in the *C:/Program Files/Thinstall.VS/Captures/<application name>/.* When the build.bat file is run a command prompt window is displayed with the status of the process. When the build.bat runs there are several steps that it does to build the package. The first step is building the virtual file system and virtual registry within the executable file that it creates (See figure 12. page 18). After the virtual file system is created it then copies over all the application files to the file system. The last step is to add the tiny Thinstall virtual operating system to the package.

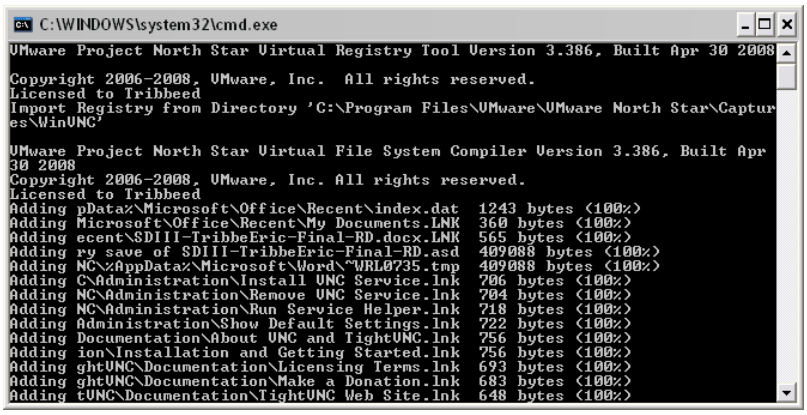

Figure 12 Thinstall: Virtual Package Builder

8.2 Login Script

Once the packages are built the Administration and Finance IT Department need to deploy each package to particular computer systems that they assigned to it. The method chosen to deploy the applications was to use a login script using VBScript code. The VBScript is automatically executed when the user logs into a domain account. The script will find what user is logging into the system, which user based software the user has installed and get the groups that the user is a member of from the Active Directory domain controller. Once the script has determined that information it will copy and install only the software that was assigned to that user. The VBScript will also have an update function to update any of the packages that have new versions, or have been repackaged. The update function built into the script might be obsolete in the next versions of Thinstall. As discussed previously a recent build of Thinstall includes an Application Sync feature which allows administrators to set a central server location which allows the Thinstalled package to periodically check for updates from that server location.

8.2.1 Configuration Variables

```
'########################################################################
'Edit these variables to fit the your domain and your server setup
'########################################################################
strDomain = "dc=legend,dc=local"  'LDAP server location
strSvrpt = "\\tribbe-base\Thinstall\" 'Where is the base directory for the Thinstall Files
strLocpt = "C:\Thinstall1\"  'Where you do you want to put the files on the local computer
strRegpt = "\\tribbe-base\Thinstall\"  'Where is the location of Thinreg.exe
strGrpPf = "ThinGrp-" 'Group prefix to identify which groups are for the applications and which
are for other purposes.
StrComputer = "." 'You shouldn't have to change this variable...this specifies which computer to
run certain commands on....more than likely the local computer.
'SQL Server Configuration Settings
strSQLsvr = "tribbe-sql"        'What is the SQL server name; Can be DNS based, or IP based
strSQLusr = "sa"                        'What is the sql server user
strSQLpwd = "pa$$w0rd"      'Password of the SQL server
strSQLdb  = "insoft"            'Database to use with the Software tracking
strTblnm  = "inSoftTb"                  'Table name of the software tracking
```

Figure 13 Script: Configuration Variables

An important feature that this script has is that it can be easily customized for most domains and most file servers. Figure 13 shows the list of variables an administrator needs to set so it would work on different domains. The *strDomain* variable is used in the LDAP connection string so that the script can get certain information from Active Directory such as group membership and distinguished name. The *strSvrpt* variable is used to set to location of all the Thinstalled application packages are located. This was set so that the packages can be shared in one location so domain users have access to copy them to their system. The strLocpt variable is where the administrator would put the packages where this is the location where the application packages are copied from the network location. The *strRegpt* variable is the network location of the ThinReg.exe file. This file is important because that is the file that registers the application with the computer. The *strRegPfi* variable is how the script can

distinguish Active Directory Thinstall Groups from other security groups used for other functions. The SQL Server configuration settings allow an administrator to specify which server to connect to as well as variables to connect to the database such as a sql username and password.

8.2.2 Listing Installed User Software

The first step In the VBScript is to get a list of all the user installed software on the client computer. The user-installed software is different from the software that is normally installed on the computer system. The location of where this information is stored is in the system's registry, *"HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall"* to be more specific. The script looks at the location in the registry and gathers the *"DisplayName"* and the "UninstallString" from each key at the registry location (See Figure 14). The information gathered in this portion of the script is used later in the script.

```
Const HKCU = &H80000001 'HKEY_CURRENT_USER
strKey = "SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\"
strEntry1a = "DisplayName"
strEntry2 = "UninstallString"
aCounter = 0
Set objReg = GetObject("winmgmts://" & strComputer & _
 "/root/default:StdRegProv")
objReg.EnumKey HKCU, strKey, arrSubkeys
```

Figure 14: Installed Software Query

8.2.3 Find Distinguished Name

The next step in the script is to get the distinguished name of the user logging into the computer. To get the distinguished name is a two step process in itself. The first step is to the get the SAM account of the user logging in. This information can be obtained from the computer the user is logging into. To get the SAM account from the computer a sql-like query

needs to be made to the object *"winmgmts"* which contains several computer specific information such as computer name, operating system information and user information (6). When the SAM account is determined it contains several pieces of extra information that is not needed for the next step.

The script is programmed to filter out the extra information by using a function called ReplaceTest. The ReplaceTest function is built into the script and its function is to strip out text contained in a string using regular expressions. The extra information that is in the SAM account is the Domain information which is in the formation *<domain>**<username>*. Using regular expressions the script will call the ReplaceTest function, give the string to be filtered, and the regular expression and it would strip out everything before the username (See Figure 15).

```
struser = objComputer.UserName
struser= ReplaceTest(struser,".*\\","")
```

Figure 15 Script: Formatting Username

Once the SAM account is found the script uses that information to find the distinguished name of the user in Active Directory. To find the distinguished name of the user logging in a LDAP query needs to be set up and executed. Using the LDAP variable, strDomain, from the configuration section of the script it runs a select statement finding the distinguished name (See Figure 16).

```
'We need to get both the name and t
objCommand.CommandText = _
   "SELECT distinguishedName FROM 'LDAP://"  & strDomain & "' WHERE objectCategory='user'
" & _
      "AND sAMAccountName='" & struser & "'"
Set objRecordSet = objCommand.Execute
```

Figure 16 Script: Distinguished Name

The result from the query displays the distinguished name along with other information that is not needed. To strip out the information that is not needed the results are broken down into an array using the commas in the string. The needed information is pulled from the array because the strings are the same formation.

8.2.4 Getting the Groups

To get the groups that the user belongs to another LDAP query is used by using the distinguished name of the user. The query puts all the groups into a single array called arrMemberOf (See Figure 17). Once the groups are determined the script then filters through the groups removing the ones that are not a part of the application deployment. This is done by using the group prefix in the configuration section of the script and comparing that string variable with the name. If the string is found in the name then it keeps the entry, if it does not it will strip out that particular group from the array.

```
Const E_ADS_PROPERTY_NOT_FOUND  = &h8000500D
Set objUser = GetObject _
   ("LDAP://cn=" & strcn & ",OU=" & strOU & ","& strDomain)

arrMemberOf = objUser.GetEx("memberOf")
```

Figure 17 Script: LDAP Query

8.2.5 Copying, Upgrading, and Registering Packages

Once the membership groups are found, the script then begins to determine if the package needs to be downloaded from the server or just register the package. There are two cases that can cause the script to download the package from the computer.

One case is that the package does not exist so therefore it needs to copy it and the other case is that the package exists but needs to be updated with a new version. This part of

the script works by first determining if the base directory exists and, if it does not then the script creates the folder. The base directory is specified in the configuration section of the script. Once the script determines whether or not the base directory exists, it checks to see if the package folder exists. The package folders are named the same as the group name from the Active Directory. The script uses the array of groups found earlier and then determines if the folder exists based on the name in the array. If the folder does not exist then the script creates a new folder using the name of the group.

After the script determines whether the base and the package folders exist, the script then starts to check if all the files are the same as the files on the server. The first step is the script gets a list of all the files in the package folder that is located on the server and put them into an array. In the second step, the script then checks to see if the each file in the array exists on the local computer. If the file does not exist, then it copies that file from the server location to the local computer. If the file does exist, the script moves onto the next step to check for any updates to the folder.

```
If objFSO.FileExists(strLocpt & x & "\" & xy) Then
                        Set objLocal = objFSO.GetFile(strLocpt & x & "\" & xy)
                        Set objRemote = objFSO.GetFile(strSvrpt & x & "\" & xy)
                        dateFileDiff =
DateDiff("n",objLocal.DateLastModified,objRemote.DateLastModified)
                        if dateFileDiff <> 0 then
                                'Wscript.Echo "not equal"
                                objFSO.CopyFile strSvrpt & x & "\" & xy, strLocpt & x & "\",
OverwriteExisting
                        End If
                Else
                        Set objFSO = CreateObject("Scripting.FileSystemObject")
                        Const OverwriteExisting = TRUE
                        objFSO.CopyFile strSvrpt & x & "\" & xy, strLocpt & x & "\",
OverwriteExisting
                End If
```

Figure 18 Script: Coping and Installing Packages

To check if the any updates exist, the script compares the modified dates with the file located on both the server and the local computer. If the dates are different, then it copies and replaces the file on the local computer using an if/then statement (see Figure 18). The reason that the compare of the two files was done by the modified date was that in the situation where an update was applied to a package but was faulty. The package could be rolled back quickly by just replacing the updated package with a backup of the original on the server so that it may propagate to each of the assigned computers the next time the user logs into the computer system.

Once the files copied from the server to the client computer the script then installs the package on the client computer. The VBScript passes on a command to the command line of the remote computer to install the application. The command that runs the application calls the thinreg.exe program which is the Thinstall utility that registers the application package with

the server and gives the thinreg.exe a parameter that specifies the folder location of the application package. Figure 19 shows that it uses the configuration variable from the beginning of the script to find the folder location of the application package and uses a wildcard character to register the application. The wild card character is used to pick up any support applications that may be located in that folder location.

```
Set WSHShell = WScript.CreateObject("WScript.Shell")
runScript= strRegpt & "thinreg.exe " & strLocpt & "/""" & x & "/""\*.exe"
```

Figure 19 Script: Install Command

8.2.6 Uninstalling Thinstall Packages

The second to last step in the script is to check to see if any Thinstalled packages are installed that are not supposed to be installed. For instance, if a user were assigned to receive a certain package but no longer needs it, the software would need to be uninstalled. The way the script does this is by comparing the list of installed software found earlier in the script with the list of Thinstalled groups to which they belong. If a particular piece of software does not match up with the groups then the software is uninstalled using the uninstall string, also found earlier in the script.

8.2.7 Software Asset Tracking

The last step in the script is to document the current list of software installed on the system that is available to all users. This is done by using the same type of code as getting the list of software installed by the user. Instead of looking in the *'HKEY_Current_User' the script* looks into the '*HKEY_Local_Machine'* registry hive (6). The software displayed at this location is the software that is available to all users. Once the script gets the '*DisplayName',* '*UninstallString',* and *'QuietUninstallString',* it makes a connection to a database specified in the

configuration section of the script. The script then opens a connection to the database specified in the configuration section of the script. Before the script inserts the data into the database, it deletes all the previous entries in the database so that only the current information is available (See Figure 20).

```
objConnection.Open _
   "Provider=SQLOLEDB;Data Source=" & strSQLsvr & ";" & _
      "Trusted_Connection=no;Initial Catalog="  & strSQLdb &";" & _
         "User ID=" & strSQLusr &";Password=" & strSQLpwd & ";"

'Code to grab the computer name

Set objComputer = CreateObject("Shell.LocalMachine")
strCompn = objComputer.MachineName

'Opens a recordset from the Database
objRecordSet.Open "DELETE FROM " & strTblnm & " WHERE " & _
   "computer_nm = '" & strCompn & "'", _
      objConnection, adOpenStatic, adLockOptimistic
```

Figure 20 Script: Asset Tracking

9. Conclusion

In conclusion, the Administration and Finance IT department of UC manages a large variety of computer systems, but is having difficulty managing them. The solution is deploying a virtualized software deployment solution **using** Thinstall to package applications into a virtual package and deploying the packages using a login script.

References

1. "Application Streaming." Thinstall. 4 May 2008 <http://www.thinstall.com/solutions/application_streaming.php>.

2. Gaskin, James E. "When Apps are Virtualized." Network World. 21 Feb. 2005. 4 May 2008 <http://www.networkworld.com/supp/2005/ndc1/022105virtual.html>.

3. "Maintain a Stable, Secure Locked Down Desktop." Thinstall. 6 May 2008 <http://www.thinstall.com/solutions/secure_apps.php>.

4. McDade, Tim. Personal interview. 19 Oct. 2007.

5. "Script Center: Getting Started." Microsoft. 4 May 2008 <http://www.microsoft.com/technet/scriptcenter/default.mspx>.

6. "The Script Center Script Repository." Microsoft. 7 May 2008 <http://www.microsoft.com/technet/scriptcenter/scripts/default.mspx?mfr=true>.

7. "Thinstall 3.0 Virtualization Suite." Thinstall. 5 May 2008 <http://www.thinstall.com/products/virtualization_suite.php>.