DEPLOYMENT GUIDE

# VMware ThinApp Deployment Guide

*A Guide to Virtualizing Applications with VMware ThinApp*

vmware®

# Contents

# About this Guide

This document provides guidance for customers seeking to package and deploy applications efficiently with VMware ThinApp. It addresses most relevant deployment considerations but does not provide comprehensive detail. Please see the *VMware ThinApp User's Manual* for the specifics of implementation and further explanation.

## Integration with Software Delivery Vendors

All the functions discussed in this document utilize native features and functions of the generally available VMware ThinApp technology. VMware ThinApp can natively plug into software delivery tools from Microsoft, BMC, HP, CA, Novell Zenworks, Symantec, LANDesk, Matrix42 and more to integrate with existing organizational processes to minimize operational and administrative costs. VMware also has partnered with specific vendors to deliver customized and integrated functionality to third-party products for the purposes of deployment, discovery, inventory, and usage tracking. Contact your software delivery vendor for specifics regarding their customized offering.

# Getting Started

Customers should obtain the following components to construct an environment with which to package and deploy applications with VMware ThinApp. The use of VMware ThinApp in no way supersedes the licensing requirements for an operating system or applications. For this reason, all components should be properly licensed or obtained for evaluation purposes only.

- A clean operating system image either physical or virtual with minimal updates and no applications pre-installed. VMware recommends the use of a VM for this function because administrators can take advantage of the snapshot functionality to roll back to the pristine state of the operating system before the application is installed. Purchase or evaluation of VMware ThinApp includes a licensed copy of VMware Workstation for this purpose.
- VMware ThinApp software installed locally or accessible via a network location. For your convenience, VMware makes ThinApp available to customers for a 60-day evaluation period at the following location. http://www.vmware.com/go/trythinapp
- Application installation source files and install procedures.

The following posting, How to Make a ThinApp Application Package, provides additional detail of what is necessary for setup capture:
**http://blogs.vmware.com/thinapp/2008/10/how-to-make-a-t.html**

## Supported Operating Systems and Application Platforms

Applications that have been virtualized with VMware ThinApp are supported on the following operating system:

- 32-bit Operating System platforms: Windows NT, Windows 2000, Windows XP, Windows XP Embedded, Windows Vista, Windows Server 2000, Windows Server 2003, Windows Server 2008
- 64-bit Operating System platforms: Windows XP 64-bit, Windows Vista 64-bit, Windows Server 2003 64-bit, Windows Server 2008 64-bit
- 32-bit and 64-bit Terminal Server, Terminal Services, and Citrix Xenapp (including Presentation Server)

- 16-bit applications running on 32-bit Windows operating systems
- 32-bit applications running on 32-bit and 64-bit Windows operating systems

### Non-supported Applications and Operating Systems

Some application types are not suitable for virtualization with VMware ThinApp. Some applications that have kernel mode components can be virtualized but the kernel mode components must remain in the native operating system.

- 64-bit applications
- 16-bit or non-Intel OS platforms such as Windows CE
- 16-bit applications running on 64-bit Windows operating systems
- Applications requiring installation of kernel-mode device drivers such as scanner and print drivers (ODBC drivers work because they are user-mode)
- Applications making use of kernel mode components such as anti-virus, personal firewalls, and VPN clients
- Applications that utilize remote network based DCOM services

## VMware ThinApp Overview

Agentless application virtualization is the enabling technology for efficient delivery of applications to end users. This efficiency allows IT to respond more quickly to the application needs of the business and end users without sacrificing management and administrative control. Application virtualization augments both traditional and virtual desktop solutions by abstracting applications from the underlying operating systems. Just as VMware virtual machine technology decouples the operating system from hardware, VMware ThinApp decouples the application from the operating system to deliver the same benefits of flexibility, portability, and isolation. VMware ThinApp plugs directly into existing IT tools and processes, enabling corporate IT organizations and ISVs to deliver encapsulated application containers across a variety of operating systems without complex configuration and installation requirements. Administrators can package applications once and deliver to users across multiple environments consisting of physical desktop, virtual desktop and shared Terminal Services. VMware ThinApp pioneered agentless application virtualization, has invested in its product development for more than seven years, and has over a million deployed seats. The VMware ThinApp technology is a critical component in the desktop family of VMware products because it brings increased flexibility and efficiency to desktop virtualization.

### Technical Description

VMware ThinApp creates application containers by using a build process to package application files and registry into a single compressed executable file which can be run on a variety of operating systems without installation. The application container utilizes block-based streaming with transparent decompression into memory to execute all application functions. Applications can be executed from a user's desktop, a network path, or removable storage like a USB flash drive or CD ROM. Applications run entirely in user mode and are visible to the operating system as standard windows processes to maintain the appropriate security context. VMware ThinApp presents operating system resources and functions to the virtualized application, allowing full functionality and a seamless user experience while still encapsulating the application's files, registry entries, COM/ActiveX controls, and services in a portable container for use across multiple operating systems.

## Benefits

VMware ThinApp's technology provides many benefits to end users and the IT organization. The following list summarizes primary benefits realized by our customers:

- Eliminate Application Conflicts
  Enable incompatible applications to run without conflicts on the same system. The entire application installation is contained in a single ThinApp *package* that is isolated from the operating system and other applications.
- Reduce Cost of Packaging and Deploying Applications
  Organizations can greatly reduce the complexity and cost of the packaging and regression test cycle because once an application has been packaged using ThinApp, the application is encapsulated into a container capable of running on a variety of Windows operating systems. Applications can be executed and updated from a central location allowing for turnkey application update and delivery.
- Agentless Application Delivery
  Applications can be run on end user systems without the need for application install, administrative rights, pre-installed agents, or component dependencies (.Net Framework, Java JRE). The application package contains all the necessary application components to be fully functional allowing immediate application functionality without any pre-requisites.
- A Secure and Supportable Delivery Model for Applications
  VMware ThinApp application packages are compressed and encrypted to prevent corruption or tampering and yet run as a visible native processes to the operating system providing transparency and visibility to security tools. Applications are run in user mode thus allowing full application functionality even in locked down desktop environments where users do not have administrative rights.
- Low Cost of Implementation and Ownership
  VMware ThinApp requires no additional infrastructure investment. ThinApp can be integrated with Microsoft Active Directory via Group Policy or work with existing software deployment solutions such from Microsoft, BMC, HP, CA, Novell Zenworks, Symantec, LANDesk, Matrix42 and more.

# Conceptual Discussion

VMware ThinApp uses packaging and deployment processes and tools similar to those used for native applications. However, unlike native applications, virtualized applications are never installed for end users, so a process called *setup capture* is used to encapsulate the process of the installation along with the necessary files and registry required for the application to execute. The *setup capture* process takes all the necessary files and administrator supplied configuration information for access control and user setting storage options to build a read-only redistributable *package* containing everything needed to execute the application. The application executes within the virtualized container, making specific requests of the underlying operating system which resides outside of the container as needed.

These virtualized application packages are either stored centrally for multiple users to access or deployed to specific end points as in the case of physical desktops or Terminal Services based environments. *Streamed Execution* mode refers to the centrally stored model and *Deployed Execution* mode describes the delivery of the virtualized application packages to the end point devices. Both modes make use of VMware ThinApp's streaming capability to pull application components into memory as normal windows processes for full application functionality.

As virtualized applications are executed by end users, the dynamic application and user setting information is redirected into a storage location defined by the administrator, called the *sandbox*. Administrators can subsequently incorporate application updates and changes to configuration by a number of methods. These updates are rolled out to end users centrally as they launch the applications, deployed to end point systems via Active Directory group policy or alternative software deployment solutions, or provided via a centralized web service that will transfer a differential update. Applications run within VMware ThinApp containers cannot be tampered with because they are read-only and embedded into a compressed *package*. However, dynamic application configuration and user settings can easily be rolled back to the initial configuration for ease of support.

## Terminology

### ThinApp Streaming

VMware ThinApp streaming has a unique meaning that describes the execution of an application rather than the delivery of the application. When a ThinApp packaged application is launched only the necessary blocks of data are streamed into memory for execution. This streamed execution occurs whether the *package* resides centrally on a file share, locally on the file system, or on a USB device. VMware ThinApp streaming does not require the local caching of files, instead it only streams into memory what is needed at that time to perform the application function. The amount streamed varies by application usage and depends on how many of the application functions are used and which dlls, registry, and files are needed for those functions. ThinApp streaming provides application files and registry settings to the end user based solely on the applications requests and regardless of the *package* size. For example, a very large application *package* that contains a suite of applications (but that is being used minimally) may stream the same amount of data as a small application that is subjected to heavy end user load.

## People and Process Considerations

As with the implementation of any technology, it is critical to do the necessary planning, identify the appropriate skill sets, and integrate with existing processes. Practically speaking, implementation involves not only the initial deployment but also the ongoing maintenance and support structure for the entire lifecycle of the applications. In terms of planning, it is helpful to identify the major user communities and their application needs, then design your implementation accordingly. Different characteristics need to be considered for physical desktop, virtual desktop, and Terminal Services/Citrix based environments.

One of the unique strengths of the VMware ThinApp solution is the portability of the technology and ease of integration into a wide array of environments. However, every organization is unique and has different objectives for implementing application virtualization. Equally important to process considerations is the commitment to leverage appropriate people to make a project successful. The following list identifies key skill sets recommended:

- Application and Packaging Skills
  When packaging applications it is best to identify dependencies, update cycles, application errors, and custom files which are necessary for full functionality. Incorporating individuals familiar with custom applications will streamline the packaging process and help prevent unnecessary troubleshooting. Individuals who currently package applications can easily incorporate the VMware ThinApp setup capture into their existing processes and are often already aware of particular application requirements.

- Desktop Administrator
  Familiarity with the specifics of the local operating system and its configuration will help determine the desktop integration process, identify user groups and application needs, and optimal methods and processes for deployment and update.
- Active Directory or Third-party Desktop Management Administrator
  Individuals responsible for Active Directory group policies for desktops or administrators of software deployment solutions for desktops are essential for integrating ThinApp packages into existing processes.

# Setup Capture and Build

The process of virtualizing an application with VMware ThinApp begins with the *setup capture* process and ends with the build of a read-only redistributable *package* that encapsulates all the necessary components of the application along with the administrator supplied configuration settings necessary for implementation. The *setup capture* process creates a *project* to store the application and configuration settings. The build process compresses and embeds the *project* directories and configuration settings into the *package*. The project directory is the source location where the administrator can return to make subsequent updates or changes to the package configuration. The result of making a configuration change and rebuilding would be two separate packages created from the same VMware ThinApp project but with different configuration settings. The process of using s*etup capture*, the *project directories*, and the *build* process is meant to be an iterative process, often referred to as 'Capture and Build.'

While there are distinct operations in these phases it is helpful for the administrator to be mindful of the future deployment model when using the *setup capture* process to package the application. Configuration settings that describe the update method, specific application characteristics, and integration with the local operating system are embedded into the *package* during the Capture and Build phases. Building the *package* is the logical transition point between creating and deploying the application. The subsequent phases of Deploy and Update utilize the *package* as a modular application container, which is then distributed and updated accordingly.

## Terminology

### Setup Capture

The *Setup Capture* wizard guides the process of capturing the application and appending administrator supplied configurations specific to the *package*. *Setup capture* operates by taking a pre-scan snapshot, then allowing the administrator to install and configure the application, and then taking a post-scan snapshot. The differential between the pre- and post-scan snapshots, which now represents the application, is placed into the project directory.

### Project

The term *project* refers to the output of the *setup capture* process. The *project* is the critical asset for managing the lifecycle of a virtualized application because it contains everything needed to build and modify the virtualized application. The following are specific components of the project directory:

1. Directories and files that have been changed by the application's installation procedure. These directories are listed by the common Windows folder macros, such as %AppData% and %SystemRoot%.

2. Directories that represent the common Windows folders contain a '##Attribute.ini' file which contains Directory Isolation Mode settings for that particular location.
3. Registry changes made by the application and the Isolation Mode settings are contained in .txt files named HKEY_Current_User, HKEY_Local_Machine, and HKEY_Users.
4. The 'bin' directory contains the *package* files in .exe or .msi format created from the build process.
5. The 'support' directory contains the Capture Machine Overview.txt file, which records the pre-packaging specifics of the system. This is a helpful file for troubleshooting.
6. The 'Package.ini' file (described below)
7. The 'Build.bat' file (described below)

### Package.ini

The 'Package.ini' file, contained in every Project folder, contains the settings recorded by *setup capture* and all other configuration settings. The 'Package.ini' file is a repository of all ThinApp configuration data for deployment, update, shortcuts, and entry points of that particular application *package*. The following is a helpful reference for specific detail on Packager.ini settings: http://www.vmware.com/info?id=765

### Build

The term *Build* describes the process by which the project directories and configuration settings are compressed and embedded into the *package*. The build operation is the last step in the S*etup Capture* wizard or can be manually run using the build.bat in the project directory. The Build process can be re-run at any time to incorporate different settings or application changes into a *package*. Any machine having access to the file share which houses the project directory and can find the VMware ThinApp executables vftool.exe, vfregtool.exe, and tlink.exe in the path can build the *package*. There is no need to return to the source machine that created the *package* or be on a similar operating system or use a management console to build the *package*.

### Build.bat

The 'Build.bat' file is a simple batch file that is utilized by the Setup Capture process or can be run manually to create the ThinApp packaged executable. The 'Build.bat' initiates the build process by making calls to three components in the ThinApp installation folder, which in turn compiles the virtual registry, virtual file system, and application logic into the executable file.

### Package

The *Package* is the result of the Setup Capture and Build process. As an application container it encapsulates the application, the ThinApp run-time engine, and all the required configuration settings. A *package* is created in a ready to run read-only .exe format. The packaged .exe is compressed and encrypted to protect the virtualized application from tampering or modification. Additionally, the *package* can be placed in an MSI wrapper for native integration with software deployment tools. The MSI wrapper includes the *package*d .exe, the Thinreg tool, and the .msi database condensed into one file for ease of deployment. The *package* is transportable and simply requires a supported operating system to run the application, unless specific access control has been configured into the *package* by the administrator. The *package* is most often named after the application, such as Mozilla Firefox.Exe.

### Isolation Mode

The term *Isolation Mode* describes the degree of isolation between the virtualized application and the files, folders, and registry of the local operating system. Administrators use Isolation Mode settings to define which resources of the local operating system are visible to the virtualized application. Isolation Mode settings by default are inherited down through a folder and registry structure. Administrators have granular control to define the appropriate isolation mode settings for each folder and registry sub-tree. The three values for Isolation Modes are Merged, Write-copy, and Full. The following is a link to the online help file which provides further details on Isolation Mode settings: http://www.vmware.com/support/pubs/thinapp_pubs.html

### Sandbox

The *sandbox* is a user specific folder created for each virtualized application that holds run-time changes to the virtualized registry, folders, and files. The virtualized application maintains the sandbox and utilizes isolation mode settings to determine when to write changes to the sandbox or to the local operating system. The sandbox is located by default in the user's %AppData% location. If folder redirection or roaming profiles are in use then the *sandbox* simply follows that path. The sandbox folder location is configurable such that it can be located centrally for users to access their application settings from multiple devices.

### Entry Points

Entry points are user accessible starting points for virtualized applications or natively installed applications for use by virtualized applications. Entry points by default correspond to the executables detected during the Setup Capture process and chosen by the administrator to make available to the end user as an entry point into the virtualized application.

## Procedural Discussion

The *setup capture* process requires the following components as part of a packaging environment that can be used repeatedly for the Capture and Build phase:

- VMware ThinApp software installed locally or accessible via a network location. For your convenience, VMware makes ThinApp available to customers for a 60-day evaluation period at http://www.vmware.com/go/trythinapp
- A clean operating system image either physical or virtual with minimal updates and no applications pre-installed. VMware recommends the use of a VM for this function because administrators can take advantage of the snapshot functionality to roll back to the pristine state of the operating system before the application is installed. Purchase or evaluation of VMware ThinApp includes a licensed copy of VMware Workstation for this purpose.
- Application installation source files and install procedures.

### Overview of the Setup Capture Process

The Setup Capture process begins with a pre-scan of the operating system to provide a baseline for comparison after the application installation. Once the pre-scan is complete, the administrator will install the application, configure the application, and then run the post-scan. The pre- and post- scan make use of operating system level snapshots that record the registry, file and folder structure, and other components affected by the application installation routine. After the post-scan the Setup Capture wizard asks the administrator to provide a series of configuration options for entry points, primary data container, inventory name, access control, sandbox location, isolation mode, and compression.

### *Step-by-Step Guidance for the Setup Capture Process*

Place the application install files on the local drive of the capture machine.

1. Go to Start > Programs > VMware > ThinApp Setup Capture or run the Setup Capture from a mapped network location that houses all of the VMware ThinApp files.

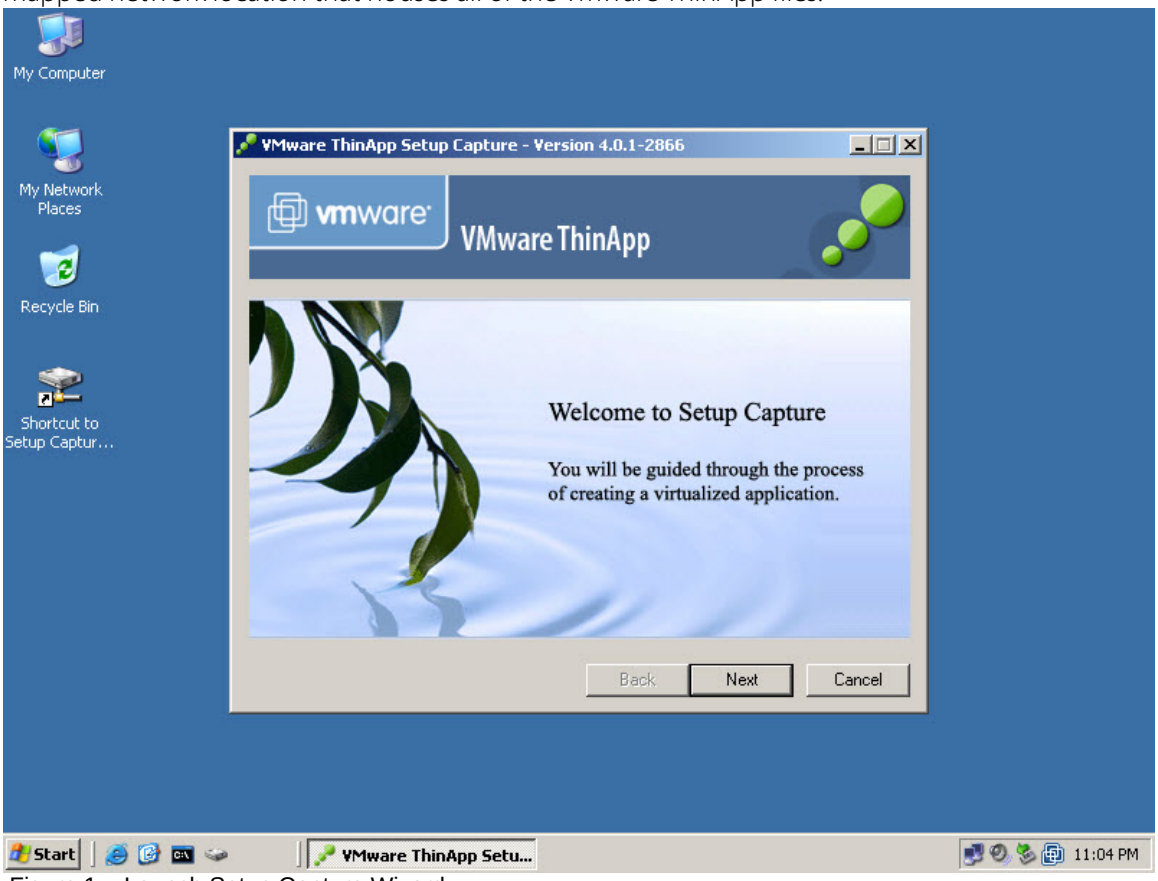


Figure 1 – Launch Setup Capture Wizard

2. (Optional) In the dialog box that clarifies the definition and use of a clean computer, click the Advanced Settings button to select the drives and registry hives to scan. You might want to scan a particular location other than the C:\ drive if you install applications to a different location. In rare cases, you might want to avoid scanning a registry hive if you know the application installer does not modify the registry.
3. Click Next to begin the first snapshot of the hard drive and registry files. The scanning process takes about 10 seconds for Windows XP.

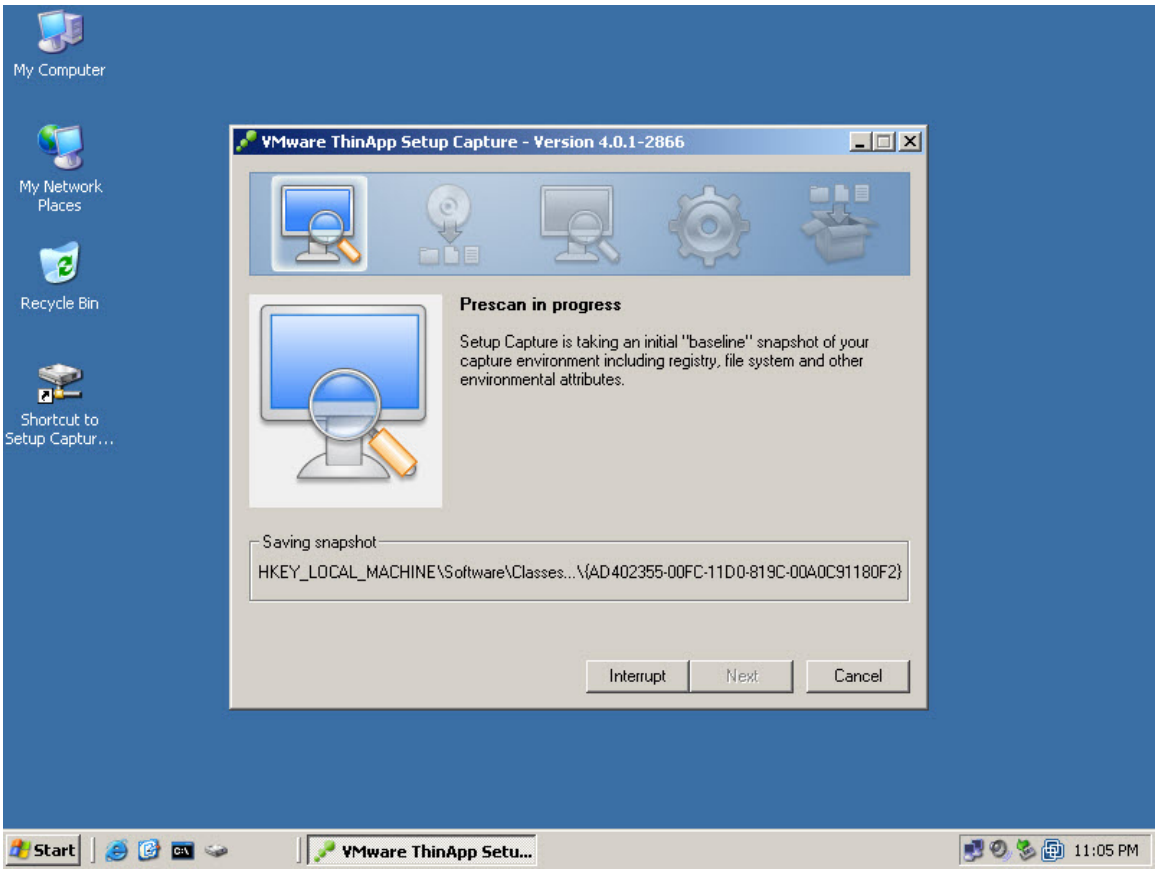


Figure 2 – Pre-Scan

4. Minimize the Setup Capture wizard and install the applications to capture. For example, double click Firefox Setup 3.0.5.exe to install Firefox. If the application needs to reboot after the installation, reboot the system. The reboot restarts the Setup Capture wizard or if multiple reboots are required re-running Setup Capture will allow the administrator to resume the process from the pre-scan snapshot at a later time.

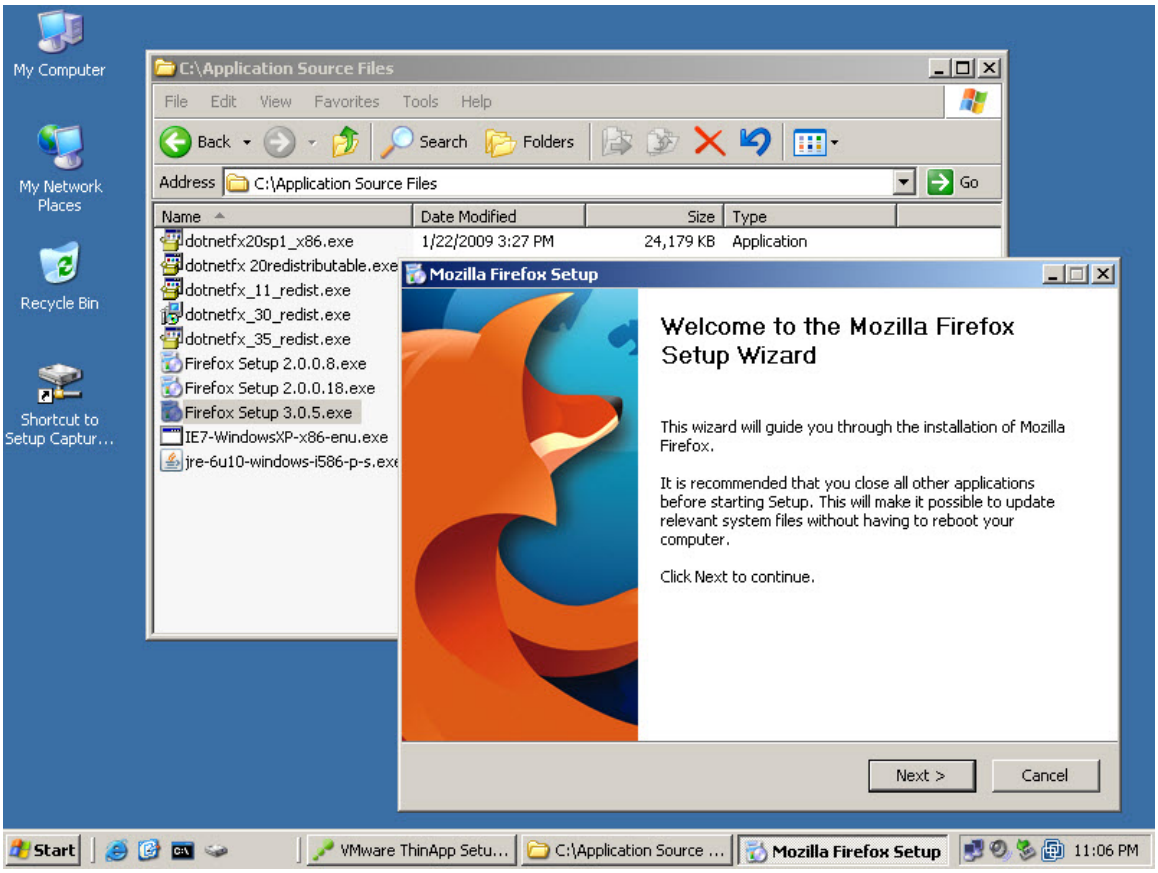


Figure 3 – Install Application

5. Make any necessary configuration changes to comply with your organization's policies, such as using specific security settings or a particular home page. If you do not make configuration changes at this time, each user must make changes at run-time which may lead to inconsistent setup and unnecessary user interaction.

6. Run the application to verify its functionality and allow any one-time setup operations to occur. Address any prompts for information before you continue with the Setup Capture wizard so that users do not encounter any unintended setup options when they run the application.

7. Close the application.

8. Maximize the Setup Capture wizard and click **Next** to proceed with another snapshot of the machine. ThinApp stores the differences between the first snapshot and this snapshot in a virtual file system and virtual registry.

9. Specify the entry points, primary data container, and inventory name:

   a. Select the check boxes for user accessible entry points.

      - Entry points are the executable files that start the virtual application. The entry points you can choose from depend on the executables that your captured application creates during installation.

      - If you install Firefox, you might select Mozilla Firefox.exe and Mozilla Firefox (Safe Mode).exe if users require safe mode access. If you install Microsoft Office, you can select entry points for Microsoft Word, Microsoft Excel, and other applications that are installed during a Microsoft Office installation.

- If you create an MSI wrapper for the *package*, the installation will create desktop shortcuts for the entry points selected on the target machine.

b. Select the primary data container, the file that will store the virtual files and registry information, from the list based on the selected entry points.
   - If the size of the container is smaller than 200MB, ThinApp creates a .exe file as the primary container. For a small application such as Firefox, any .exe file can serve as the main data container.
   - If the size of the container is larger than 200MB, ThinApp creates a separate .dat file as the primary container because Microsoft Windows does not show shortcut icons stored in large .exe files. ThinApp generates small .exe files along with the .dat file to store the icons for Windows to display.

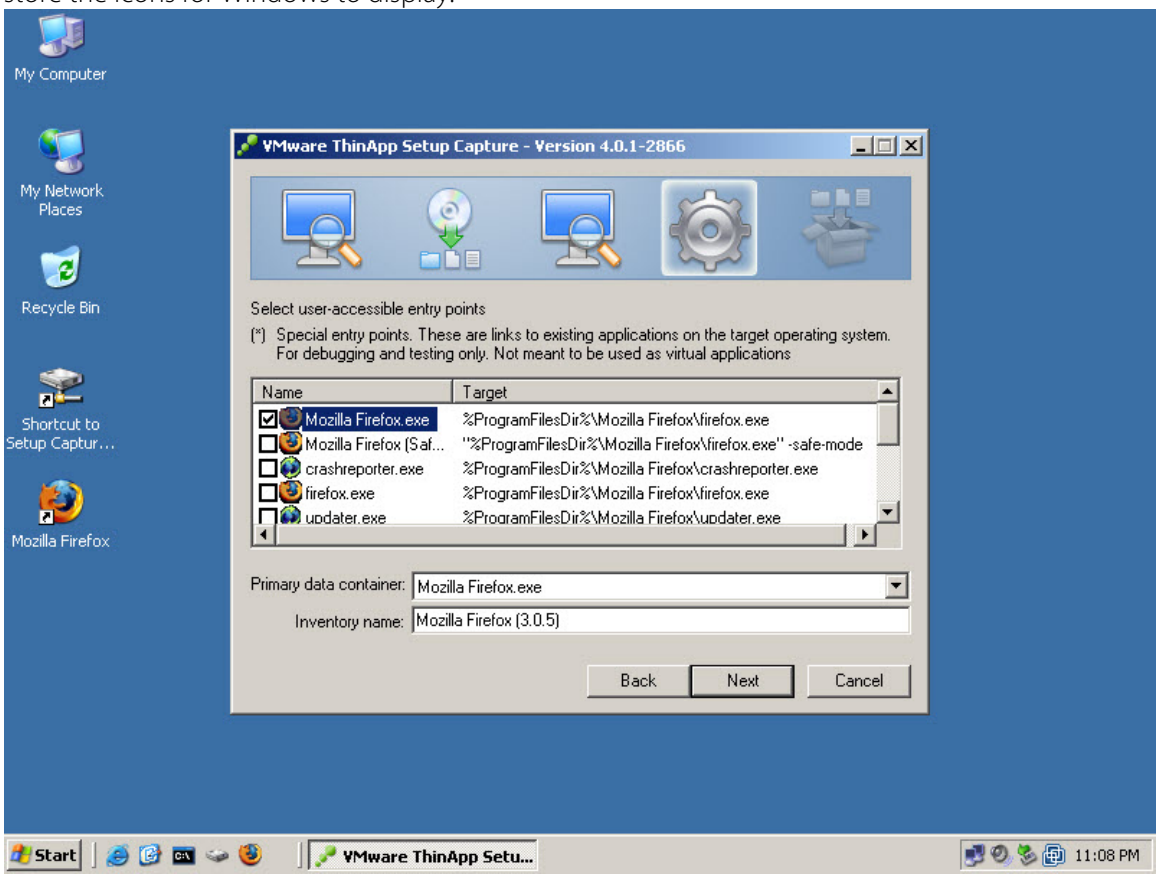


Figure 4 – Select Entry Point, Primary Data Container, and Inventory Name

10. Specify the inventory name that ThinApp uses for internal tracking of the application in the Package.ini file. ThinApp uses the inventory name during the update process with the Application Sync utility and also for identification in the Add/Remove Programs control panel applet which is often queried by software deployment solutions for inventory purposes.
11. (Optional) Select the Active Directory groups that you want to authorize for access to the application:
12. Click **Add**
    - To specify objects, click **Object Types.**
    - To specify a location in the forest, click **Locations.**

- To search object names, enter the names according to the examples in the dialog box.
- To locate user names in the Active Directory forest, click **Advanced** and use the **Common Queries** tab to search for groups according to names, descriptions, disabled accounts, passwords, and days since the last login.
  For example, you might restrict access to an application to ensure users do not pass it to unauthorized users.

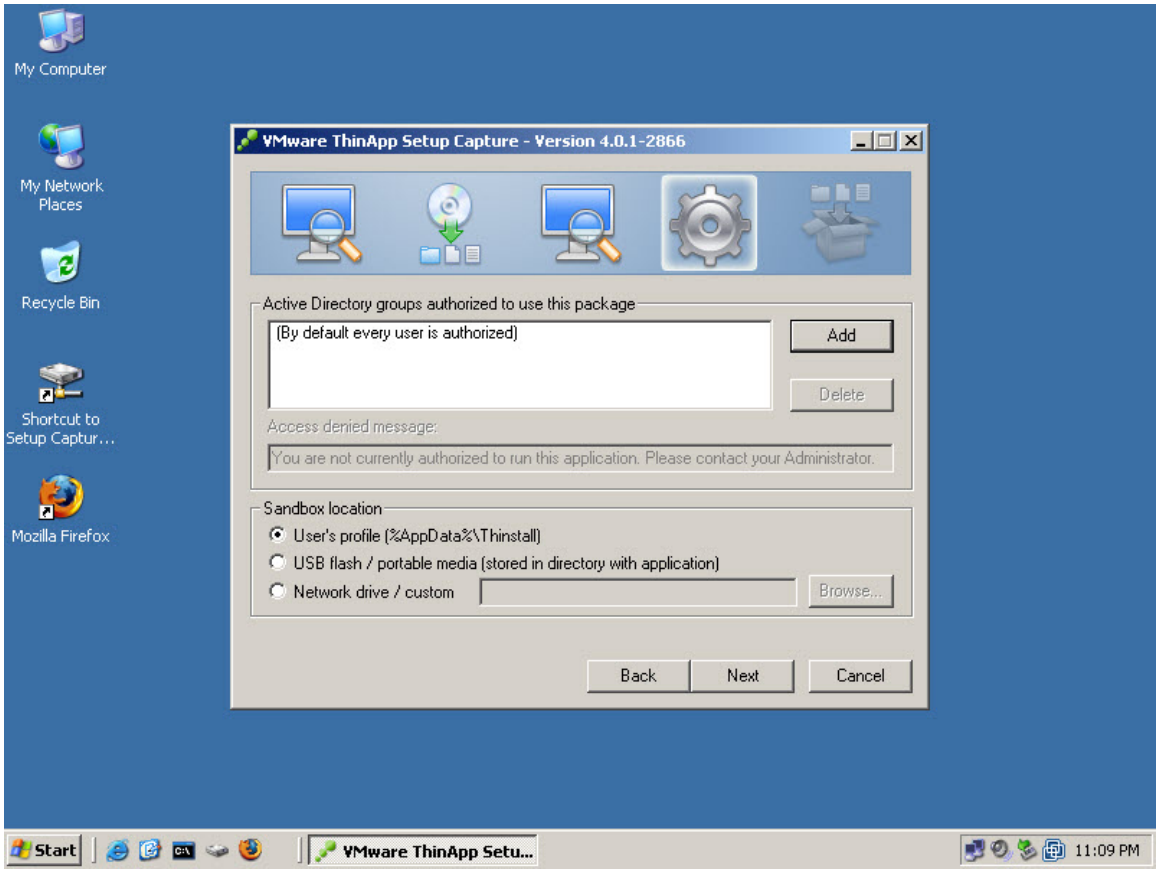


Figure 5 – Specify Active Directory for Access Control

13. (Optional) Select the location for the sandbox to be stored. By default, the user sandbox is stored locally in the Windows User Profile in the %AppData% location. Administrators can also use a network location such as a Home Drive path or a mapped network location in this field if desired. If you store the sandbox in a network drive, enter the relative path to the location where you want the sandbox created. A sample path is \\thinapp\sandbox\firefox. You can select a network location even if an application is installed on a local machine.
14. Select the isolation mode to determine which files and registry keys are visible to the virtual application you create:

    To allow the application to read resources on and write to the local machine, keep the default **Merged** isolation mode. This means that the application can modify elements outside of the virtual application *package*. To allow the application to read resources on the local machine and restrict most modifications to the operating system, select the **WriteCopy** isolation mode. VMware recommends this mode for legacy or untrusted applications. This mode is useful for locked down desktops where you want to prevent users from affecting the operating file system and registry files.

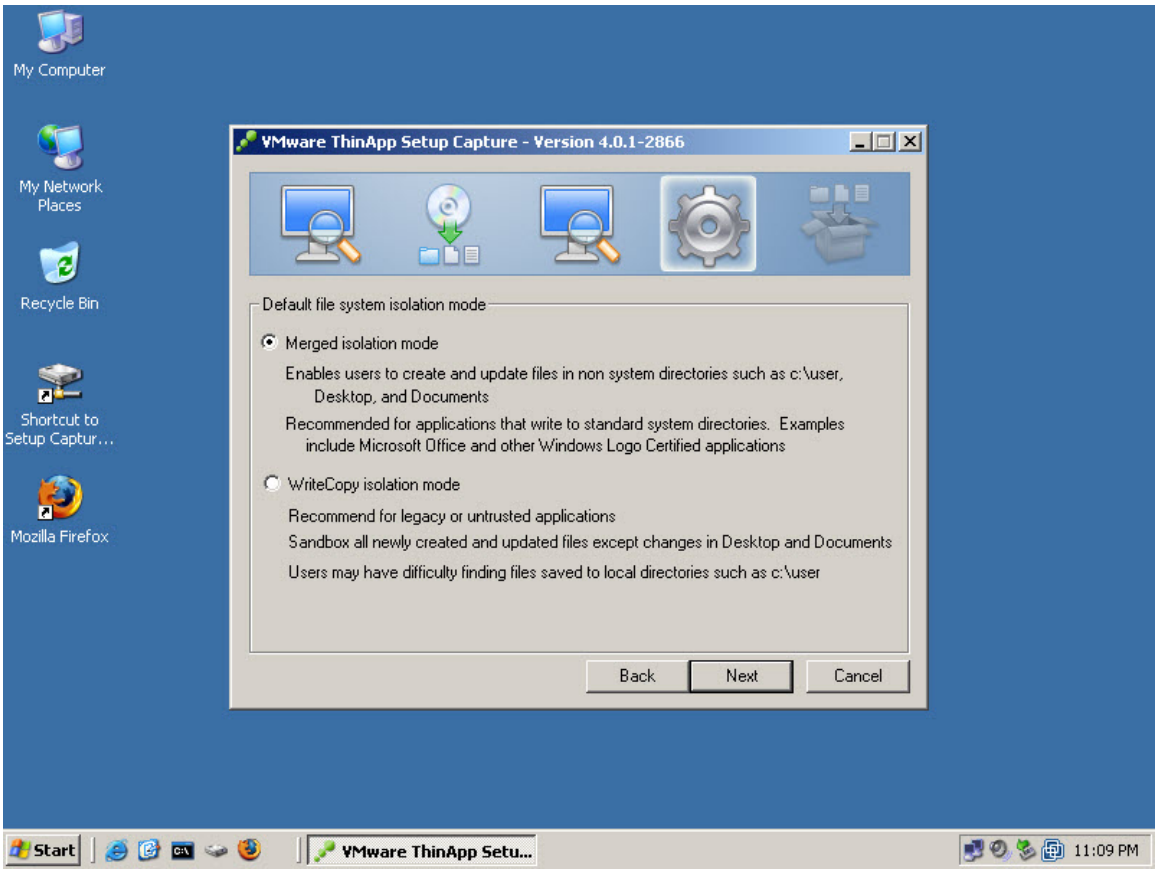


Figure 6 – Select Default Isolation mode

15. (Optional) Change the directory where you want to save the application *package*. The *package* stores the captured software applications. If you keep the default directory and capture Mozilla Firefox, the path might appear as C:\Program Files\VMware\VMware ThinApp\Captures\Mozilla Firefox.

16. (Optional) Select the Build MSI *package* check box and change the MSI filename. Select the option to Build MSI if you wish to integrate with application delivery tools that distribute that format. MSI generation requires you to install the MSI on the target device before you can use the application *package*. The MSI wrapper will use the thinreg.exe utility during install to register file-type associations, register desktops shortcuts, display control panel extensions, and facilitate file launching.

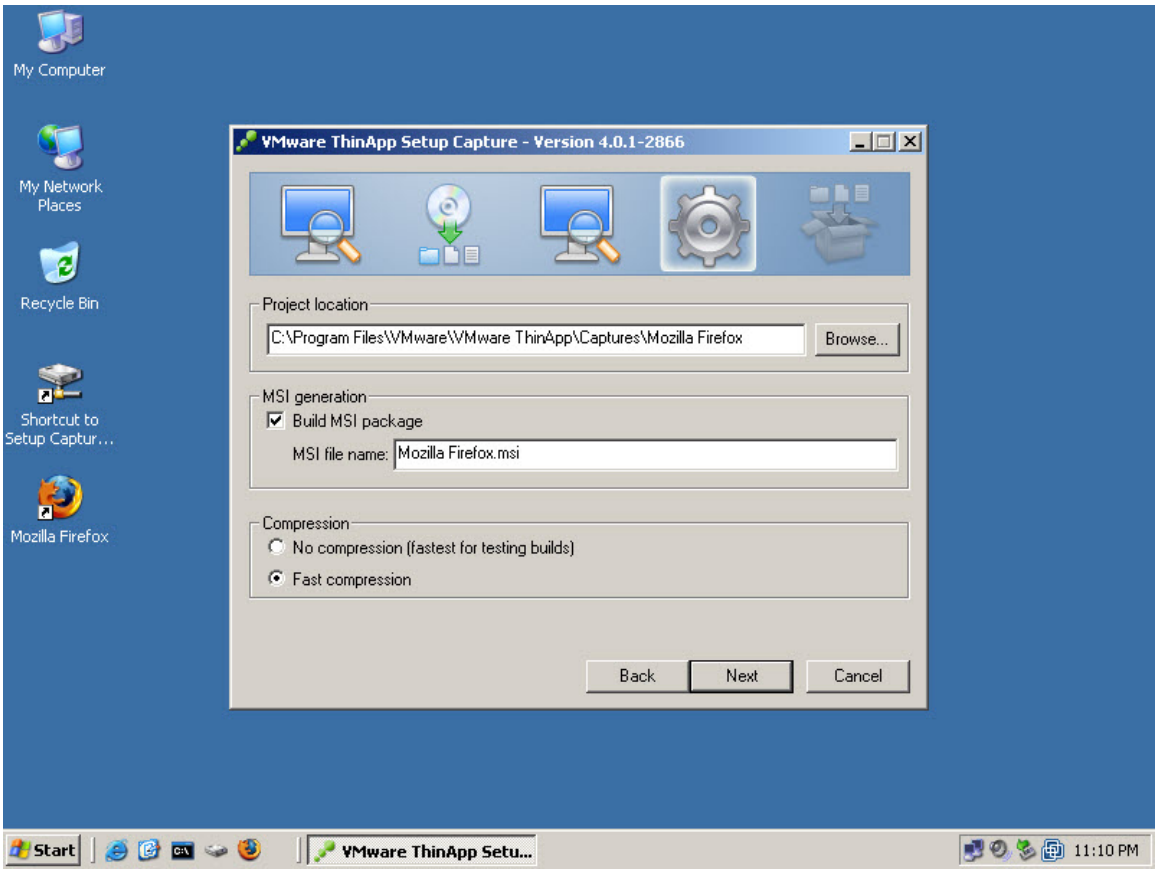


Figure 7 – Select Project Location, MSI generation, and Compression

17. (Optional) To reduce the storage necessary for the application *package*, select the Fast compression radio button. In typical circumstances, compression reduces the disk storage requirement by 50 percent and will benefit performance in streamed execution mode.
18. Click **Next** to create the ThinApp project.
19. In the final dialog box, launch one of these options and click **Finish**.

    - Click **Browse Project** to look at the ThinApp project files in Windows Explorer. For example, if you captured Mozilla Firefox, the location of the project files might be **C:\Program Files\VMware\VMware ThinApp\Captures\Mozilla Firefox**. You might browse the project prior to building the application executable or MSI file to update a setting, such as the sandbox location in the **Package.ini** file which contains the administrator configured settings entered during the setup capture process.

      The project includes folders, such as **%AppData%**, that represent file system paths that might change locations when running on different operating systems or computers. Most folders have **Attributes.ini** files that specify the isolation mode at the folder level. The isolation mode setting at the granular folder level overrides the overall isolation mode setting of the Package.ini file.

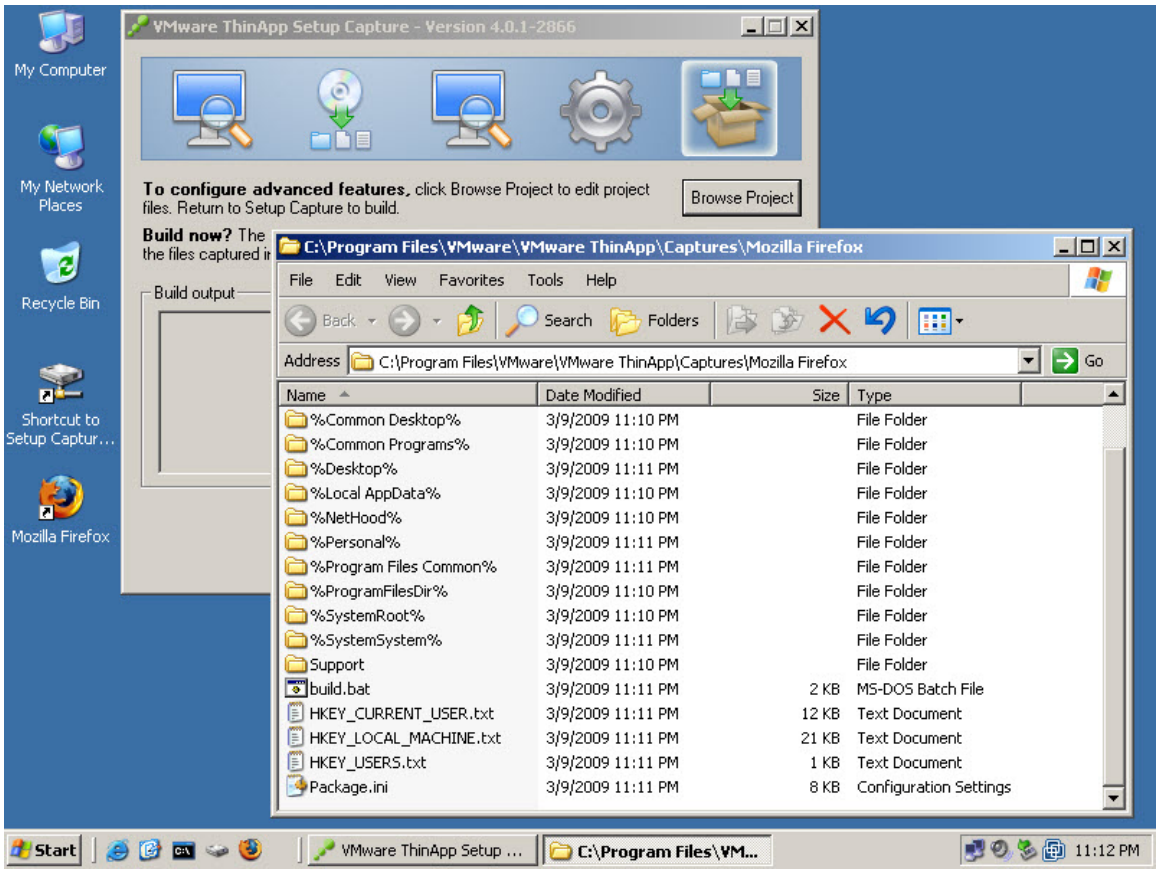

Figure 8 – Project Directory

- Click the **Build Now** button to build an executable *package*, and optionally the MSI wrapper, containing the application installed during the Setup Capture process. The build output appears in the display box.

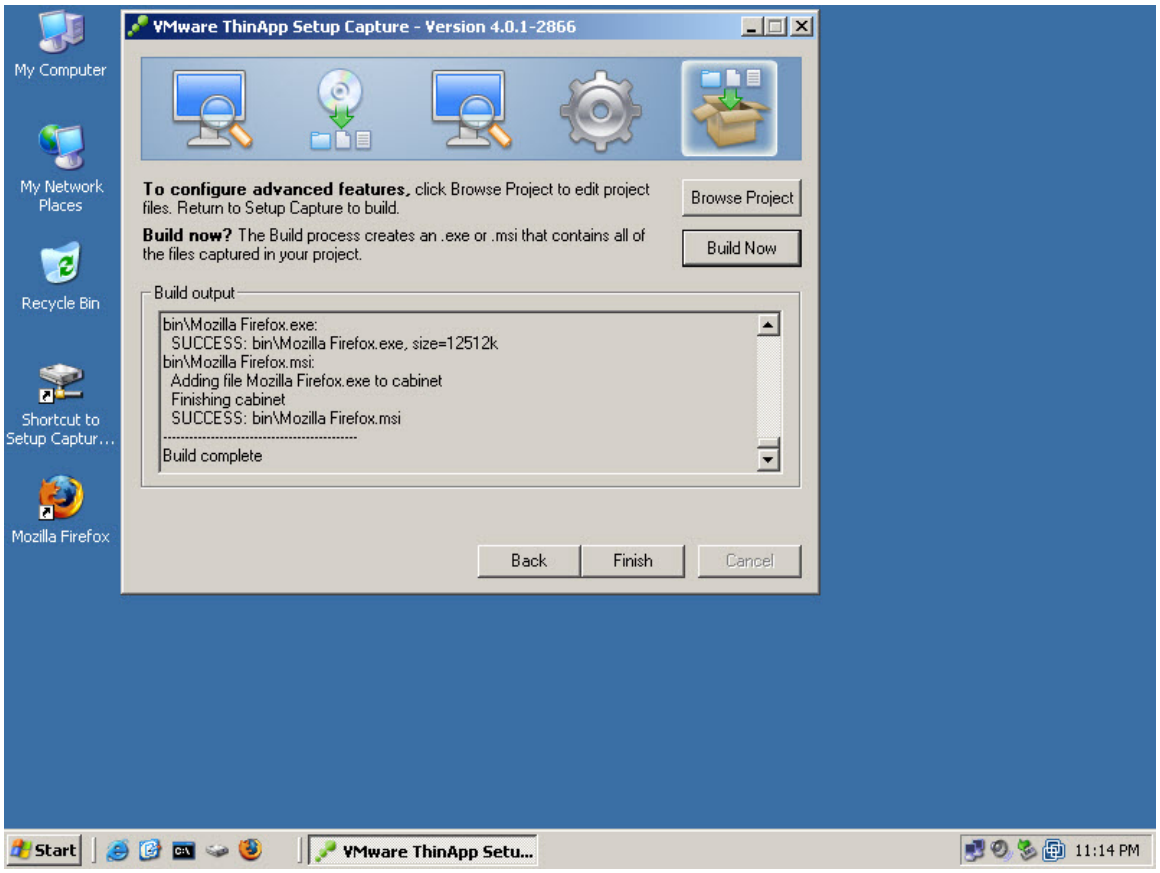

Figure 9 – Build Now screen

You can rebuild the project if you need to make changes at any time after clicking **Finish** by using the build.bat file in the project directory if you need to make changes.

## Modifying Isolation Modes

VMware ThinApp provides the Merged and WriteCopy isolation mode choices in the Setup Capture wizard. You can use a third isolation mode, Full, outside the wizard in the ThinApp project text files. The Full isolation mode secures the virtualized application by blocking visibility to system elements outside the virtual application *package*. This mode restricts generated data to the sandbox and ensures that no interaction exists with the environment outside the virtual application *package*.

ThinApp caches the isolation modes for the registry and the file system at runtime in the sandbox. If you change the isolation mode for the project and rebuild the executable file, you might need to delete the sandbox for the change to take effect.

## Modifying Settings in the Package.ini File

The Package.ini file contains configuration settings and resides in the captured application folder. For example, a Mozilla Firefox path might be C:\Program Files\VMware\VMware ThinApp\Captures\Mozilla Firefox\Package.ini.

The following parameters are examples of settings that you might modify:

- DirectoryIsolationMode – Sets the isolation mode to Merged, WriteCopy, or Full.

ThinApp caches the isolation modes for the registry and the file system at runtime in the sandbox. If you change the isolation mode for the project and rebuild the executable file, you might need to delete the sandbox for the change to take effect.

- PermittedGroups – Restricts use of an application package to a specific set of Active Directory users.
- SandboxName – Names the ThinApp sandbox.

  You might keep the name for incremental application updates and change the name for major updates.
- SandboxPath – Sets the sandbox location.

  You can set the sandbox in a USB location if the application executable file resides in that location.
- SandboxNetworkDrives – Specifies whether to direct write operations on the network share to the sandbox.
- RequiredAppLinks – Specifies a list of external ThinApp packages to import to the current package at runtime.

  If ThinApp cannot import a package, ThinApp stops the base application.
- OptionalAppLinks – Specifies a list of external ThinApp packages to import to the current package at runtime.

  If ThinApp cannot import a *package*, ThinApp allows the base application to start.

### Edit the Package.ini File

To edit the Package.ini parameters use Notepad or another text editor:

1. Open the Package.ini file located in the captured application folder.
2. For example, a Mozilla Firefox  path might be C:\Program Files\VMware\VMware ThinApp\Captures\Mozilla Firefox\Package.ini.
3. Activate the parameter to edit by removing the semicolon at the beginning of the line.
4. For example, activate the RemoveSandboxOnExit parameter for Mozilla Firefox:
5. RemoveSandboxOnExit=1
6. Another example might involve commenting out the Protocols parameter if you do not want Firefox to take over the protocols.
7. Delete or change the value of the parameter and save the file.
8. Double click the build.bat file in the captured application folder to rebuild the application *package*.

For example, a Firefox path to the build.bat file might be C:\Program Files\VMware\VMware ThinApp\Captures\Mozilla Firefox\build.bat.

### Modifying Settings in the ##Attributes.ini File

The ##Attributes.ini file applies settings at the directory level. The Package.ini file applies settings at the overall application level.

For example, you can set the isolation mode at the directory or application level to determine which files and registry keys are visible to the virtualized application. The detailed setting in the

##Attributes.ini file overrides the overall Package.ini setting. The Package.ini setting determines the isolation mode only when ThinApp does not have ##Attributes.ini information.

To compress only certain folders with large files rather than an entire application, compress files at the folder level with the CompressionType parameter in the ##Attributes.ini file.

The ##Attributes.ini file appears in most folders for the captured application. For example, the Attributes.ini file might be located in C:\Program Files\VMware\VMware ThinApp\Captures\Mozilla Firefox\%AppData%\##Attributes.ini.

### Edit the ##Attributes.ini File

Use Notepad or another text editor to update the ##Attributes.ini file.

9. In the ##Attributes.ini file, uncomment, update, or delete the parameter.
10. Double-click the build.bat file in the captured application folder to rebuild the application *package*.

## Recommended Practices

### Creation of the Package Digest

VMware recommends that customers create a standardized document that details the components of a *package* for IT organizational use. The packaging process is iterative by nature and so it is necessary to have a controlled document which communicates the primary contents and deployment configuration. A package digest serves this function and is highly beneficial over the lifecycle of the application. Critical elements of a package digest are outlined below. Customers should include information specific to their environment to make the digest as useful as possible.

- Source Application Files & Versioning Information
- Project Directory Location
- Deployment Objectives
- Package.Ini settings
- Isolation Mode settings and explanation for non-default settings
- VBScripts Scripts used, the originating authors, script documentation, and their dependencies
- Platform Testing results
- Build Notation and Naming Conventions

### Control Access to Project Directories

VMware recommends that administrators control access to the project directories and have regular backup procedures in place. The project directories become the source files for generating updated application packages and allow the administrator to incorporate changes without re-running the Setup Capture process. The project directories are also helpful for troubleshooting efforts because the specifics of the capture machine's configuration and application modules are contained in the project directory. Access should be restricted to administrators because some of the settings in the Package.Ini file provide sensitive information for access control and update locations. Once the application is packaged, this information cannot be changed by end users.

### Prune Project before Build to Minimize Package Size

A number of directories can be deleted from the project directories before the build process. Common directories to delete include files used for installation (.cab files or application install directories), uninstall directories, and miscellaneous other directories (%Cookies%, %History%, % Internet Cache%, %SystemRoot%\Debug, %SystemRoot%\Installer, and %SystemRoot%\pchealth). Deleting unnecessary files and folders from the project directory will reduce the overall size of the *package*.

### Use a Clean Operating System for Setup Capture

VMware recommends that customers install only the basic components of the operating system for the machines that is used for Setup Capture. However, an updated Windows Installer 3.1 will likely be required for the application install to commence. It is also recommended that you use the oldest version of the operating system to ensure that the application install puts down all required files. The reason to capture from a clean operating system is to ensure that all the files and components necessary for the application are detected by the Setup Capture process. If there is application install logic that looks for a certain version of a .dll and the capture machine finds it in the local operating system, then that .dll will not be included in the virtualized application *package*, and that may not allow the application to function when deployed to an operating system instance that has not been updated.

# Application Deployment

The process for deploying ThinApp application containers is very simple as there is no actual installation of the application and interoperation with the local operating system is minimal. Deployment involves making a decision for a centralized or de-centralized model for application delivery and then integrating the application into the desktop for end user accessibility.

## Terminology

### Execution Mode

*Execution Mode* describes the two options for providing virtualized applications to end users.

- *Streamed execution* mode allows the application to be centrally stored and accessed by multiple users. *Streamed execution* mode is a one-to-many model which provides centralized deployment and update of an application *package* to multiple end users for execution via a Windows desktop shortcut.
- *Deployed execution* mode application packages are first deployed to the end users system, and accessed from the local device. Users execute the application from an application *package* that is now local which allows for offline application use.

### Thinreg

The file Thinreg.exe is a utility provided to automate the registration of application shortcuts on the desktop, file-type associations, and entries in the Add/Remove programs applet of the Control Panel. Thinreg can be run from a script, command line, or a login script to accomplish these functions. Thinreg is also incorporated into MSI packages when this feature is utilized within the ThinApp build process.

# Procedural Discussion

## Choosing an Execution Mode

One of the decision points for virtualizing applications with VMware ThinApp is to choose which delivery model is appropriate for user groups and applications. There are two primary options of application delivery: one is the streamed mode of execution, and two is the deployed mode. Both of these options have requirements and benefits that are listed below. A hybrid approach is acceptable as well. Determine the appropriate execution mode for each application and user group.

### *Streamed Execution Mode*

The streamed mode of execution will often be the best option for environments that are centralized and desktops are always online. In streamed mode the application is launched from a shortcut on the start menu or desktop on the local workstation and then streamed into memory as the application requests files and registry. The details of ThinApp Streaming were discussed earlier in this guide.

### *Requirements*

The user must always have access to the central network location where the applications reside.

### *Recommendations*

The storage location that hosts the applications should be made highly available such that the physical uptime of either the host or storage device does not impact the environment. The use of any number of SAN, DFS, or file-replication technologies is sufficient to accomplish the objective of making the file share highly available and redundant.

The path through the network between the client device and the central network location should be robust. A virtualized application utilizes standard SMB protocol. The amount of network traffic will vary based on the application and the functions used by the end user.

### *Benefits*

Centralized administration is the primary benefit of the streamed mode of execution. The 'one-to-many' model of providing an application to one location for many users provides an efficient and effective model for application delivery. Providing access to the application merely involves placing a shortcut to the application on the desktop and can be automated through the use of the Thinreg utility in a login script.

The application packages, which can be large in size, do not have to be delivered to the end user devices so there is no need to transfer large files across a network or integrate with a deployment mechanism to distribute them. Additionally, there is no local disk footprint on the end user device because the applications are streamed into memory.

For users that access applications from multiple devices the streamed model of execution provides a single point of administration and a consistent user experience across multiple devices.

### *Deployed Execution Mode*

Deployed Execution Mode involves distributing the virtualized application packages to the end user's operating system. The actual location of the package can be on the local file system or a USB device. In this distributed model, each client device receives the *package* locally and therefore can

run the application regardless of network connectivity. End user devices that are occasionally or always offline will require this deployed execution mode.

### *Requirements*

Distribution of the packages to the local operating system is required in this model. A number of options exist to fulfill this requirement: Active Directory based publishing via Group Policy, 3rd party software deployment solutions, and/or custom scripted mechanisms. Users who are occasionally offline must have all applications and components deployed before working offline. Subsequent application deployment and updates are subject to network availability or a delayed update tactic such as providing CD's or USB devices with updates.

### *Recommendations*

Integrate the delivery of packages, which can be large .exe or .msi files, with your existing organizational process. An existing process, such as Active Directory publishing via Group Policy, will have an already established support structure and administration workflow. You can use Group Policy to deploy software to groups, OU's, or individuals. See the following KB article for details: http://support.microsoft.com/kb/324750

### *Benefits*

Once the application *package* is delivered, application performance and availability is not subject to network or storage dependencies.

## Choosing a Sandbox Location

When users execute virtualized application, ThinApp stores changes that the application makes in the user's sandbox. Most often administrators choose to keep the sandbox in the user's profile. The user's sandbox can also be in a central location for either deployed or streamed execution mode to allow users to retain their individual application settings across multiple devices.

VMware recommends that you configure your packages to keep user settings for applications in the sandbox but keep user data such as documents and spreadsheets stored in their local file system or on a central home drive location, not in the user's sandbox. This will allow the user to access their data regardless of the location of their sandbox or their application.

## Desktop Integration Mechanisms

### *Registering Applications to the Desktop*

Virtualized Applications integrate with the desktop in the following ways:

- Windows Shortcuts on the Start Menu and Desktop
- File Type Associations
- Entries in Add/Remove Programs

The process of registering applications to the desktop makes use of the ThinReg utility whether the *package* is deployed as an .Exe based *package* or in an MSI wrapper.

### *MSI-based Packages via Group Policy*

An organization with an established mechanism for deploying MSI files, such as Active Directory, can simply deploy these packages in the same manner that they would deploy native applications.

When applications are deployed and installed as MSI's the desktop integration occurs automatically as ThinReg is run during the install.

### *Exe-based Packages via ThinReg*

Packages that are created in the .EXE format will benefit from registration via the Thinreg.exe tool to provide seamless integration into the desktop. Applications will launch and run without registration, however, file type associations are often necessary for users who start with a data file and launch the application by association.

The Thinreg.exe tool is a simple utility that can be run from a login script, from a local directory, or a file share. It provides the ability for administrators to register all of the packages at once using an asterisk (*) as a wildcard character. Administrators can use pre-made scripts that run based on group membership to only register packages that are valid for a certain group or for individuals. Also, ThinReg is aware of the Active Directory 'permitted groups' listing provided during the Setup Capture process. So if the user is not a member of the permitted groups ThinReg will not register the *package*. If you choose to deploy packages in .EXE format but wish to use an alternate method of creating shortcuts, you can use standard Microsoft Folder Redirection to place shortcuts on the desktop and in the Start menu. http://support.microsoft.com/kb/232692.

## Controlling Access

The process of deploying virtualized applications offers administrators control and flexibility over which machines and users receive either the application packages or access to the packages. Utilizing Active Directory or an alternative software deployment solution for distribution allows an organization to use the existing processes and controls. In addition to these organizational controls, VMware ThinApp allows an administrator to embed access control into the *package*. This access control mechanism is obfuscated from the end user when the *package* is built so it is impossible to identify or remove before the application is launched. In this way, the access control travels with the *package* if it is moved between devices after deployment. This mechanism can also be used when packages are hosted centrally on a file share as a secondary control in addition to file share permissions. There are three mechanisms described in the following sections, which are available to use for access control functions.

### *Active Directory Permitted Groups*

You can control access to applications using Active Directory groups. When the administrator specifies PermittedGroups in the setup capture process or manually places the SIDS in the package.ini file they are embedded into the *package* during the build process. The PermittedGroups entry in the Package.ini restricts usage of a *package* to a specific set of Active Directory users and provides the administrator a way to customize the error message to the user if they are not allowed to launch the application. For a desktop that is offline, the PermittedGroups function will utilize cached credentials to determine if the user has permission to launch the application.

### *Custom Scripting Options*

VMware ThinApp allows for the execution of custom scripts before starting an application packaged with ThinApp, during the application's use, or after an application exits. Callback functions allow you to specify when blocks of code execute. Using the OnFirstParentStart function allows an administrator to check for a condition when the application is launched. To add scripts to your application, you can create an ANSI text file with the .vbs file extension in the root project directory for an application (the same directory in which Package.ini is located). During the build process, ThinApp adds all of the script files to the application package and then at runtime it

executes each of these script files. This is a very extensible mechanism that can incorporate local and/or network variables for determination of whether the user will be allowed to launch the application.

Reference the VMware ThinApp User's Manual section 'Using Scripts' for more detail. Example scripts are provided. Link provided below:

http://www.vmware.com/support/pubs/thinapp_pubs.html

The VMware ThinApp blog also provides further explanation and a number of scripts which can validate conditions prior to application launch. Link provided below:
http://blogs.vmware.com/thinapp/2009/01/adding-non-acti.html

#### *AppSync Parameters for Package Expiration*

VMware ThinApp provides tethered control mechanisms for those devices that are typically described as unmanaged, that is, devices that are only occasionally on-line with a corporate network or not members of the organization's Active Directory. For these types of devices, you can utilize Application Sync for access control as well as an update mechanism. The use of the AppSyncExpirePeriod entry, which is embedded in the *package*, will set an interval for how many days the *package* will run without accessing a particular URL. Warning messages and their frequency can also be configured.

## Recommended Practices

### Discovery and Inventory

When packaging applications for delivery, VMware ThinApp provides an Inventory Name to the operating system. The inventory name can be customized and placed in the parameters of the MSI wrapper or executable *package*. The inventory name will also appear in the Add/Remove Programs applet of the Control Panel. Tools that provide discovery and inventory of applications will be able to retrieve this information alongside natively installed applications. For tools that do not reference these locations, administrators can place a tracer registry key or file in the local operating system in a pre-defined location for identification by software delivery tools.

### Application Monitoring and Host Security Software

The implementation model of the virtualized application container into the local operating system provides transparency and security in that processes which are run in the application container are fully visible to the local operating system. Thus, application monitoring and metering tools and security mechanisms in the local operating system provide their functions for applications virtualized by VMware ThinApp in the same way as applications that are natively installed.

### Environment Specific Configuration

The user experience of applications virtualized with VMware ThinApp inside of a virtual desktop or a shared desktop (vis-à-vis Terminal Services) is identical to a physical device. One of the primary advantages of using VMware ThinApp is the economy of scale created when application containers can be reused across multiple platforms. The location of the application sandbox is the primary consideration when deploying applications to multiple platforms.

### *VMware View and Virtual Desktops*

As organizations begin to realize the benefits of virtual desktops, they tend to separate the OS layer from the Application layer in order to achieve greater management and storage efficiency. VMware ThinApp enables organizations to accomplish this objective and approach the most scalable and cost-effective model for providing desktops.

### *With VMware View Composer*

When used in conjunction with View Composer, administrators have the option of utilizing User Data Disks to logically separate application and user storage from the operating system. By default, VMware ThinApp places the application sandbox in whatever location the %AppData% variable resolves to in the operating system. This will redirect the dynamic changes of the applications and user-specific settings stored in the sandbox and maintain these through a 'refresh' of the operating system. In this way, user applications and configuration settings are maintained.

### *Without VMware View Composer*

As noted previously, VMware ThinApp will place the application sandbox in the %AppData% location. This location is often managed through the use of roaming profiles, folder redirection, or home directories. Centralized storage of this data enables a degree of separation between user personalized settings and the local operating system. Administrators have the option of manually configuring the location of the sandbox, per application, and this can be set previously or at run-time. Please refer to the VMware ThinApp User Manual for details on the use of multiple locations for the sandbox.

### *Terminal Services Configuration*

For users that access a shared desktop, meaning a Terminal Services based configuration, the default configuration will redirect the sandbox into the local profile. For multiple reasons, Microsoft and Citrix recommend the use of Terminal Server Profiles and Home Drives for storage of user specific data. The use of local profiles often results in profile corruption and the users settings are limited to that one server. VMware recommends utilizing Group Policy, login scripts, or folder redirection of the *%APPDATA%* variable to a pre-defined location such as the Home Drive. See the example below:

%HOMEDRIVE%%HOMEPATH%\WINDOWS\APPDATA

# Application Update

Providing application updates can be necessary to either provide additional application functionality for end users or to comply with administratively prescribed updates to software. When packaging applications it is necessary to decide if the responsibility to *package* the application rests with the user or the administrator. Users who self-update virtualized applications will incorporate the application changes directly into the applications sandbox, which may increase the size significantly. If a user self-updates an application then those settings also may interfere with future updates provided by the administrator.

## Terminology

### AppLink

Application Link is a feature that connects dependent application packages at runtime. This allows the administrator to build relationships between packages in order to package, deploy, and

update component pieces in separate ThinApp packages rather than using Setup Capture to package all needed components into a single executable.

### Application Sync

The Application Sync feature is a setting that initiates the pull of a differential update package from a central http web service or UNC location. The interval for polling for updates and the location of the http service or fileshare is configurable along with other settings in the Package.ini file.

### SBMerge

The SBMerge.exe utility allows the administrator to merge run-time changes from an existing sandbox into any project directory of a captured application. The virtualized application *package* can then be rebuilt and distributed to end users incorporating the changes from the SBMerge process.

## Procedural Discussion

### Packaging Updates and Modifications

There are three methods for capturing and deploying updates. Choose the method most appropriate for the update you wish to deploy.

- Recapture
  Recapture simply means going through the setup capture process again for the purpose of incorporating the updates in between the Setup Capture pre-scan and post-scan snapshots. The result of this process is a new *package* that has the new changes in configuration or updates embedded. For example, your original *package* was Microsoft OneNote. To create the updated *package*, simply install Microsoft OneNote and apply the most recent Service Pack, then build a new application *package*.

- Sandbox Merge
  This method consolidates updates from a sandbox into an existing project directory. To use the Sandbox Merge method, first launch the virtualized application onto a clean workstation. Then run the update, which will place the new files, registry, and configuration changes into the sandbox of that workstation. Then use the sbmerge utility provided by VMware to merge the changes from the sandbox of that capture machine into the existing project directory, and then rebuild the *package* to incorporate the changes.

- Post-Capture
  The post-capture method of incorporating updates involves manually placing folders in the appropriate directories of the capture, manually editing the registry files to include changes, and editing the package.ini file to change configuration settings. Use this method when you definitively know the files or registry changes that you want to make. This method does not require the use of the Setup Capture process but you must re-build the *package* using the build.bat file to incorporate the changes.

### Deploying Updates

Once the applications updates have been incorporated into a *package* there are three methods for deployment. When choosing which method is best it is important to consider the following questions:

1. For deployed mode of execution only: Is the update of significant size? For deployed mode of execution it will be necessary to distribute the *package* to the end workstations. Should this be scheduled after hours due to network impact?
2. Once the update is deployed should users have a new sandbox created or re-use the existing sandbox configuration? User specific registry settings and preferences may be stored in the sandbox so in most cases re-using the existing sandbox is preferable.
3. For packages delivered via streamed execution, is there a change window where the administrator can make changes without user impact? If so, the package replacement method is an option. If there is no defined maintenance window then a side by side update will be required. (see next section for further explanation)
4. What is the capability for rollback if the application package is not functioning?

### *Package Replacement*

The package replacement method for updating application packages can be used for either streamed or deployed execution mode. If you have created an updated package and have an administrative window where no users will launch the applications then you can simply replace the original .exe based *package* with the updated .exe. Make sure that the filename stays exactly the same: users depend on the shortcuts previously created to launch the applications.

### *Side by Side Update*

The side by side method for updating application packages can be used for either streamed or deployed execution mode. There is no requirement for application downtime. This method works by placing the new application *package* in the same directory as the original application *package* and incrementing the filename extension to an integer number. Subsequent updates can be placed in the directories with extensions .2, .3, etc.

The implementation of this update strategy follows a simple process. When a user launches an application from a shortcut that references the original .exe, logic built into the *package* automatically checks for identical *package* filenames with a numeric extension in the same directory. If an updated *package*, such as Mozilla Firefox.2, is found, the application launches using the file with the highest numeric extension. Always keep the original .exe that is referenced by the shortcut in place as it is a necessary pointer for the application to launch with or without updated packages. There is no downtime for the users with this method of update and no change window required for the administrator. Users will execute the updated *package* as they launch the applications and the original application packaged .exe directs them to the updated *package*.

### *AppSync*

Application Sync provides updates to unmanaged machines that connect over networks with some degree of latency. AppSync provides a mechanism for a differential transfer over http to the endpoint, thus it is only used for application packages in deployed execution mode. When an application starts, Application Sync can query a Web server or fileshare to see if an updated version of the *package* is available. If an update is available, the differences between the existing *package* and the new *package* are downloaded and used to construct an updated version of the *package*. The end user must have the rights to modify the packages. If not, then the AppSync utility can be run as a scheduled service as a user with sufficient rights to perform the update. The updated *package* is then used for future launches of the application. Settings that configure the location for Application Sync and detailed configuration are contained in the package.ini file.

### *AppLink*

Application Link is a feature provided with VMware ThinApp that allows the administrator to build relationships between packages. Administrators should understand this capability and use this function to create modular packages that link together rather than creating fewer larger packages that will be more difficult to distribute and update. You can use Application Link to create relationships to local or remote application packages which contain components or dependencies for the originating application *package*.

Create links between packages for the following scenarios:

- Link runtime components, such as .NET, JRE, or ODBC drivers, with dependent applications. For example, you can link .NET to an application even if the local machine for the application does not allow for the installation of .NET or already has a different version of .NET.
- Package and deploy application-specific components and plug-ins separately from the base application. For example, you might separate Adobe Flash Player or Adobe Reader from a base Firefox application and link the components.

### *To Create an Application Link between Packages*

Follow the process below to set up the link. You can also use nested links between multiple packages or create a link to a directory using a '*' wildcard to establish links to all components in that directory.

1. Create the *package* with the component that you want to link, build the *package* as an .exe and then rename the file to a something other than an .exe extension to prevent users from running that *package* directly. A .Dat extension will be used for this example, AdobeFlashPlayer.Dat
2. Create the capture of the originating *package* with the component already installed, for example, Mozilla Firefox.
3. In the originating *package*, Mozilla Firefox, open the package.ini file and edit the following line in the [Build Options] section:
4. RequiredAppLinks=AdobeFlashPlayer.Dat
5. Place both packages in the same directory, locally or in the central fileshare.

**Note:** Application Links can be specified as Required or Optional in the Package.ini. If specified as Required the primary application will NOT launch if it cannot connect to the Linked application.

## Recommended Practices

### Use Application Link to Compartmentalize Updates

The use of the Application Link feature provides an expedient mechanism to address frequent updates of component *package*s such as Microsoft .Net and the Java JRE. In the case where an application is dependent on a component which is updated more frequently than the application, Application Link allows the administrator to avoid recreating the entire *package* every time a component is updated.

### Use Application Self-Updating Sparingly

Allowing users the auto-update their applications is very much a one time decision that cannot be easily brought under administrative control at a later time without losing the user specific changes

in their sandbox. Removing the components of an application that allow for auto-update in the original *package* will prevent users from managing their own updates and allow for proper administrative control over when and how application updates occur.

# Additional Resources

Some helpful tools include the following:

- Sysinternals Tools such as Procmon:
  http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx
- The AppDeploy.com website provides an online reference for specific application installation and configuration information. There is also an active community for support of packaging nuances for various software products. http://www.appdeploy.com
- Use an advanced editing tool to make efficient changes to Ini files and compare directories, files, etc. Some examples include: Beyond Compare and AdminScript Editor
- ThinApp Post Build GUI provided by the community:
  Thinstall Helper http://thinstallhelper.cis-group.nl/
- Miscellaneous ThinApp Utilities and links
  http://t3chnot3s.blogspot.com/2008/11/thinapp-links-and-3rd-party-utilities.html
- VMware ThinApp Blog
  VMware ThinApp employees regularly post and participate on the VMware ThinApp blog site
  http://blogs.vmware.com/thinapp/
- Product Documentation http://www.vmware.com/support/pubs/thinapp_pubs.html

## About The Author

Aaron Black is a Senior Technical Marketing Manager at VMware. In this role, his primary focus is to develop technical content to aid in evaluation and implementation of VMware ThinApp technology. Aaron's background includes roles as a systems engineer and solutions consultant in the Technical Services organization. His previous positions include systems engineer with Citrix Systems, leading a technical corporate IT team at Sprint, and solutions design for customers of Choice Solutions, a platinum reseller of VMware products.

## Acknowledgements

Significant contributions were made by Desktop Specialists Travis Sales, Dean Flaming, and Peter Bjork.

**VMware, Inc. 3401 Hillview Ave Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com**