



realtimepublishers.com[™]

The Definitive Guide[™] To

Windows Installer
Technology
for System Administrators



Darwin Sanoy and Jeremy Moskowitz

Chapter 5: Windows Installer with or without Active Directory.....	122
Beware the Tide of Windows Installer	122
Services Provided by Win2K Technologies	122
IntelliMirror Repository Technologies Overview	123
IntelliMirror Deployment Technologies Overview	125
Source Lists—the Good and the Bad.....	126
Trickle Services, CD-ROM Distribution, and Source Lists	128
Fixing Existing Unmanaged Sources.....	128
Designing the Package Distribution Repository	129
When to Choose Something Other than DFS	130
DFS Functionality Alternatives	131
Managed Drive Letters and Managed Environment Variables.....	132
Package Source List Management.....	137
When to Choose Something Other than FRS	138
FRS Alternatives.....	138
Directory Structure Issues.....	139
Directory Structure Considerations.....	140
Administrative Installs.....	141
Repository Availability Service Level Agreement.....	143
Package Deployment Technology Planning.....	143
IntelliMirror Fine Print	144
When to Consider Alternatives to IntelliMirror Deployment.....	145
Summary.....	146

Copyright Statement

© 2003 Realtimedpublishers.com, Inc. All rights reserved. This site contains materials that have been created, developed, or commissioned by, and published with the permission of, Realtimedpublishers.com, Inc. (the “Materials”) and this site and any such Materials are protected by international copyright and trademark laws.

THE MATERIALS ARE PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. The Materials are subject to change without notice and do not represent a commitment on the part of Realtimedpublishers.com, Inc or its web site sponsors. In no event shall Realtimedpublishers.com, Inc. or its web site sponsors be held liable for technical or editorial errors or omissions contained in the Materials, including without limitation, for any direct, indirect, incidental, special, exemplary or consequential damages whatsoever resulting from the use of any information contained in the Materials.

The Materials (including but not limited to the text, images, audio, and/or video) may not be copied, reproduced, republished, uploaded, posted, transmitted, or distributed in any way, in whole or in part, except that one copy may be downloaded for your personal, non-commercial use on a single computer. In connection with such use, you may not modify or obscure any copyright or other proprietary notice.

The Materials may contain trademarks, services marks and logos that are the property of third parties. You are not permitted to use these trademarks, services marks or logos without prior written consent of such third parties.

If you have any questions about these terms, or if you would like information about licensing materials from Realtimedpublishers.com, please contact us via e-mail at info@realtimedpublishers.com.

Chapter 5: Windows Installer with or without Active Directory

by Darwin Sanoy

Although Windows Installer saw the light of day as the technology that installed Office 2000, its grand entrance was with the release of Win2K. Many new management technologies were also released in Win2K. To the casual observer, it might appear that the management technologies in Win2K are only available if a company is willing to deploy Win2K to all of their servers and desktops and to use AD for authentication and directory services. In fact, Windows Installer can actually be used very effectively in the absence of Win2K servers, AD, and even Win2K or later desktops. This chapter will examine several topics with two major themes in mind:

1. If a company does not have AD, what plans must be made to ensure that Windows Installer has the necessary supporting technologies to be effective. This perspective is important for companies who will never have AD as their primary directory service and companies who will have a delay or extended AD deployment.
2. If a company has or plans to deploy all the Win2K technologies (server, directory, desktop), there are still some significant challenges to building software distribution by Microsoft's IntelliMirror playbook. This perspective will point out these areas and highlight alternative approaches.


Beware the Tide of Windows Installer

Packaging is an expensive and time-consuming activity that software vendors are not willing to do twice. As software vendors change to Windows Installer packages for their latest software releases, vendors will cease to provide setup.exe packages. Many vendors are delaying the switch to Windows Installer until Windows Installer has a much greater install base. However, once the tide turns, it might catch many organizations off guard.

This chapter is somewhat presumptive that your company is taking a planned approach to the introduction of Windows Installer technologies in your environment. However, even if you do not have the luxury of deploying Windows Installer as part of a formal IT project, it will be important to manage the items highlighted in this chapter to ensure that Windows Installer does not create more problems than it solves for your company. You can do so by gradually phasing in some of the new support technologies with the first Windows Installer packages that need to be deployed.

Services Provided by Win2K Technologies

Management technologies deployed in Win2K provide some key services to the IntelliMirror desktop management model. Most notably are technologies that support the package distribution repository by allowing it to be universally accessible, locally sourced, load balanced, and fault tolerant. These technologies also help to replicate the files to multiple locations throughout the network. The primary technologies at work here are Distributed File System (DFS) and File Replication Service (FRS).

 In Chapter 4, we talked about the fact that Windows Installer relies on Group Policy for managing application settings that must be changed after package deployment. This chapter does not discuss replacements for this functionality in detail. Any third-party policy management system can replicate the basic registry tweaking abilities that are used in Group Policy for managing application settings.

Win2K technologies also allow packages to be installed on client workstations. The primary technology here is Group Policy Objects (GPOs), which require Win2K Server, AD, and Win2K or later desktops, as Figure 5.1 illustrates.

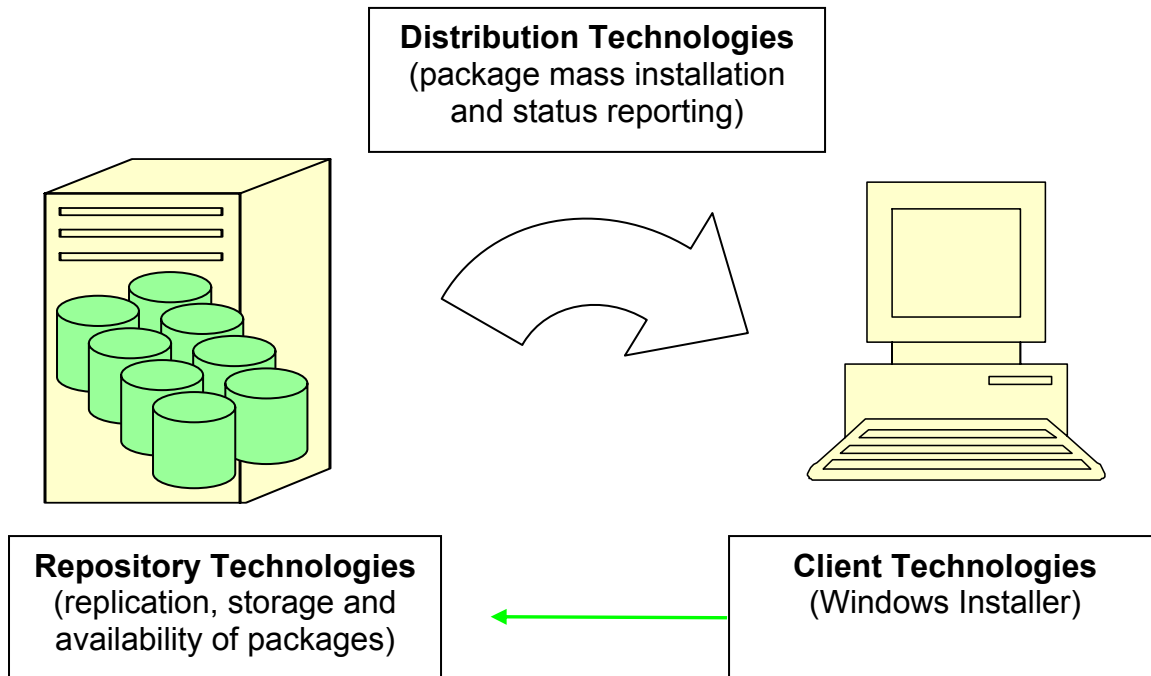


Figure 5.1: Win2K technologies needed for Windows Installer package distribution.

IntelliMirror Repository Technologies Overview

The out-of-box IntelliMirror desktop management solution uses DFS for repository services in large-scale enterprise implementations. DFS allows Windows Installer packages to be sourced from a single domain-level share name. Because DFS operates at the file-system level, its services are transparent to the client applications that use them. When designed and configured properly, DFS provides the following benefits (see Figure 5.2 for an illustration of these concepts):


- **Local sourcing**—Requests for files are automatically taken from a server that is in the same local site as the computer requesting the files. Figure 5.2 does not indicate the location of the client computer; however, DFS would favor a server in the same site as the client when finding a source location for the files.

- Universal availability (file location abstraction)—Requests for files are made to a share name that appears to be attached to the domain rather than a specific computer. Win2K DFS then determines from which server to actually retrieve the files. Figure 5.2 shows that the share name `\\domain\share` (where *domain* is any AD domain and *share* is a DFS share name) can be requested, and DFS finds a server location for this reference. This type of approach is also known as *abstracted* because the client computer is requesting a logical location and DFS finds a real location.
- Fault tolerance—If a request for a file encounters a server that is not operating, a new server is tried until a working server is found. The requesting application does not need to be aware of the multiple locations where these files reside. If any of the servers in Figure 5.2 were not available, DFS would find a different server for the client to use.
- Load balancing—Requests for files are automatically balanced between identical directory trees on multiple servers, which provide those files to the network. In Figure 5.2, this functionality would manifest itself by dynamically allocating half of the client requests at Site 1 to one of the servers and the other half to the other server.

Although these benefits are all desirable, the two that organizations of all sizes will be affected by are local sourcing and universal availability. Local sourcing ensures that packages are not pulled across WAN links, which is critical for the user experience due to potential impact to shared WAN bandwidth.

Universal availability lets Windows Installer packages be installed on computers with little or no concern for where the computer might be moved—either permanent moves during system provisioning or temporary moves constantly made by mobile computers. Universal availability is related to local sourcing for mobile computers whose “local site” changes depending on where they are located.

FRS provides replication services for DFS. FRS is the technology that replicates AD objects. FRS is a simple replication system for synchronizing the contents of DFS shares. It automatically configures a replication topology, tracks when files change, and automatically copies updates to all replicated copies. As Figure 5.2 shows, each server that participates in replicating a set of files will map to the other servers and push files to them as changes occur.

 FRS is also used to replication logon scripts, GPOs, and directory changes; however, FRS acts very differently when performing these replication activities than when it replicates DFS shares. I will discuss this behavior in more detail later in this chapter.

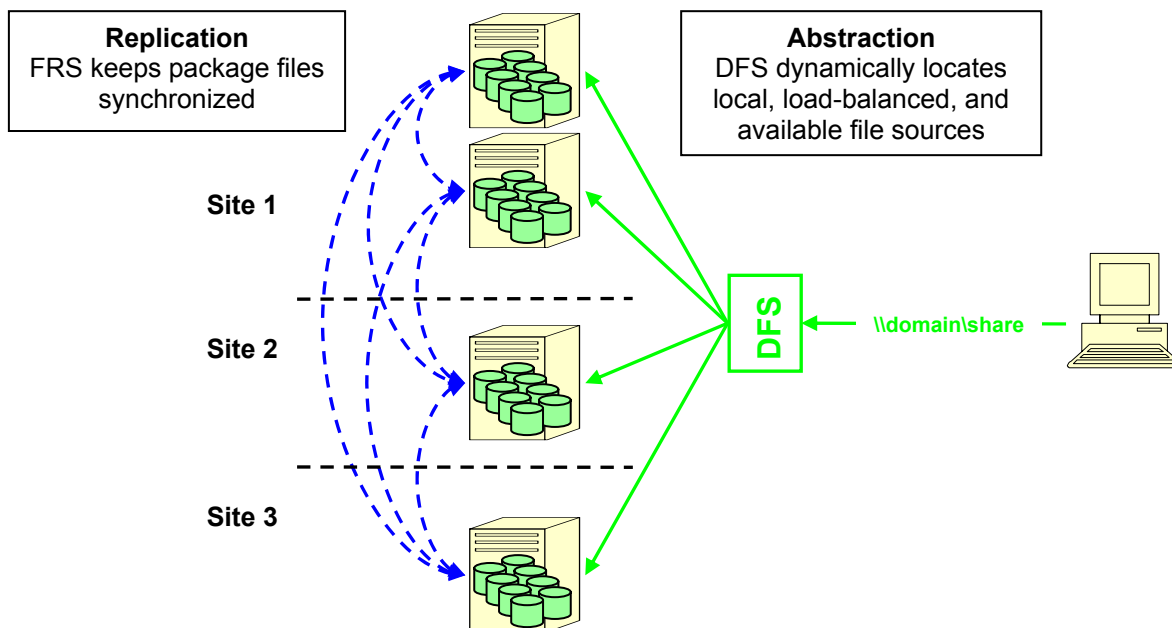


Figure 5.2: IntelliMirror repository technologies.

IntelliMirror Deployment Technologies Overview

GPOs are the main technology that allows Windows Installer packages to be distributed directly to clients without the use of an Electronic Software Distribution (ESD) system. As Figure 5.3 shows, Group Policy uses a client agent that reads AD objects to determine whether software packages need to be installed. This same agent reads the package installation information from the directory and triggers the actual installation of the Windows Installer package on the client. One of the biggest limiting factors in utilizing this technology is that clients must be running Win2K or later and AD must be used for directory services. However, even if AD is available, there are other limitations of Group Policy deployment that should be considered—we will discuss these limitations later in this chapter.

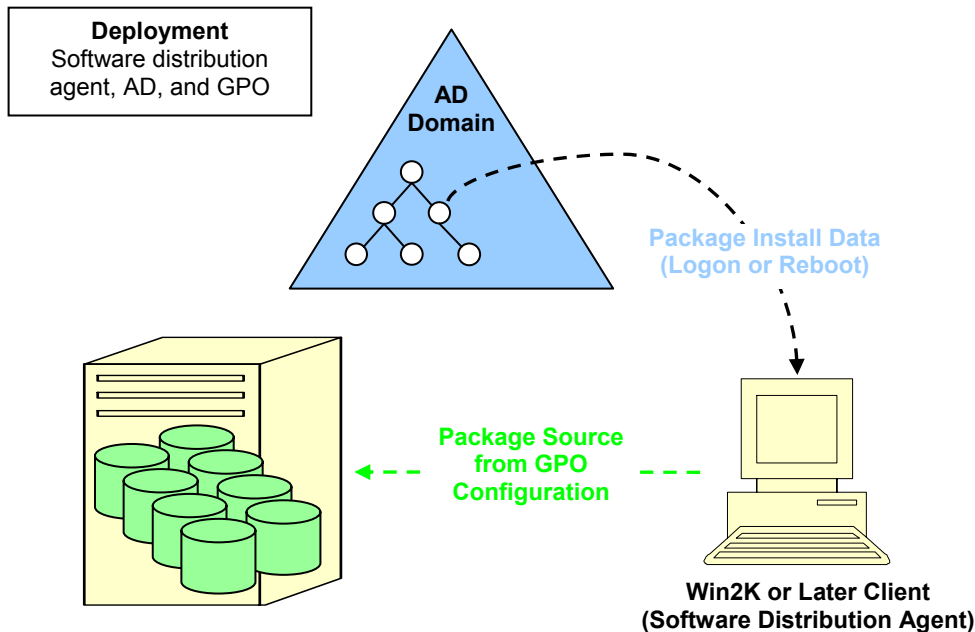



Figure 5.3: IntelliMirror deployment technologies.


Source Lists—the Good and the Bad

In the last chapter, source list management was mentioned as a necessary best practice. As mentioned in that chapter, each package installed on a computer keeps a list of places where the package source files can be found. Package source files include the package file itself (MSI), any needed transforms (MST), and the actual software application files. The software application files can be embedded in the MSI, external from the MSI in .cab files, or external from the MSI and uncompressed. Aside from the installation of the application itself, they are also used for self-healing, install-on-demand, patching, and for server-based client applications. Source locations are also used to re-cache missing transforms if they are unavailable in the local Windows Installer file cache.

 Microsoft has enhanced Windows Installer 2.0 so that it does not need to touch the source files as frequently as previous versions do. In addition, Windows Installer 2.0 supports building patches that do not require access to the original package source files.

Source lists are intended to provide fault tolerance in finding source files, so they are processed sequentially until a valid source location is found. Source lists do not inherently provide any type of load balancing. A poor man's load balancing can be accomplished by randomizing the source list when the package is installed.

Source lists can be managed more loosely in a by-the-book enterprise IntelliMirror implementation because packages are generally installed from a DFS share. The DFS share handles fault tolerance as well as several valuable features. A by-the-book implementation will need to manage source lists if the overall approach includes any mobile machines that will be installing from removable media.


 When no efforts are made to manage the source list for a package, the source defaults to the location of the package file when the initial installation begins. A casual approach to installation locations was the norm with setup.exe technologies, and most administrators do not discover this painful Windows Installer behavior until they have quite a mess on their hands.

There are three types of source lists—*removable media sources* for packages located on CD-ROMs, floppy drives, and so on; *network-based sources* for packages located on network drives or UNC; and *URL-based sources* for packages located on Web servers.

Drive imaging is a popular method of deploying ready-made desktop system configurations. If Windows Installer packages are deployed on the desktop image, special care must be taken to ensure that the embedded Windows Installer source lists will be relevant in all the physical locations where the image will be used. Because decisions to use drive imaging might occur in a different part of your IT organization, this nuance can be lost in the shuffle of desktop engineering.


The following list provides the basic actions that can be taken to manage the source list:

- Manage which type of source lists are scanned first—removable media-based, network-based, and URL-based (SearchOrder policy).
- Add source list entries one or more at a time in a specified order (SOURCELIST public property or scripting API call).
- Force the SourceDir private property value so that the *recorded* initial location is different from the *actual* initial location (custom action). Doing so can only be accomplished if the forced location is available to the client at the time of installation.
- Use a custom action to make MSI API calls to clear the source list and add new sources.

 WindowsInstallerTraining.com publishes a script called MSICAResetSources (MSI Custom Action Reset Sources) that inserts a custom action into existing MSI files that will implement the approach in the last bullet point. This script uses the default public property SOURCELIST to populate the source list—essentially overriding the normal behavior of making the install location the first source location. If SOURCELIST does not exist, the custom action does nothing, which ensures that your package defaults to the normal expected behavior. You can download this script from <http://windowsinstallertraining.com/msiebook>.

You can combine these methods to do things such as remove the original install location and add managed locations or add network sources and reorder scanning so that network sources are preferred over media installs (such as with laptops that load software from CD-ROMs).

Environment variables can also be used in source lists to make the lists dynamically point to site-level repository locations. If you are using an environment variable strategy rather than a managed UNC or drive location, the source list must also be cleared of the initial installation location. The reason is that any environment variables used in the initial installation location will be resolved to a literal location before Windows Installer processes the package. This literal location then becomes the first source list location—which does not include the environment variable to make it dynamic. This occurrence is in contrast to a managed UNC or drive location, which does not require source list management because the embedded literal location is made dynamic through DFS or dynamic drive mapping.

 Using environment variables causes the “dynamic” nature of your source lists to be distributed by being embedded in every package configuration on every computer. Using a managed UNC or drive letter leaves the “dynamic” nature of source lists in the hands of a centrally manageable infrastructure.


Trickle Services, CD-ROM Distribution, and Source Lists

Many distribution systems, including SMS 2003 (code-named Topaz), support the ability to trickle a package down to a client slowly over time, then execute it from the local drive. These types of services generally do not attempt to manage the source list for your package, so the first and only source list entry will point to the locally cached copy. Locally cached copies can be deleted by the distribution system on some clean up interval. In some companies, mobile users who are always remote use a CD-ROM for software distribution. In both of these cases, it is important to manage the source lists for installed packages. You can use several approaches to manage the source lists.

One approach is to include a couple of network-based source locations in the SOURCELIST property when launching the package. This approach causes the client to search for network locations first and then for the media it was originally installed from—if the mobile user keeps the media handy, they can use it. If they do not have it, they can connect to the network and attempt the Windows Installer operation again.

It might also be worthwhile to set the SearchOrder policy to *nm* to ensure that Windows Installer searches network sources (*n*) before media sources (*m*). *nmu* is the default behavior for this policy when it is not configured (*u* being URL-based sources), but setting the policy explicitly helps others to know that it is a managed policy value. It should be set high in your OU hierarchy to prevent overriding at lower levels.

Another approach is to clear the source list of all sources, then add only network sources. Doing so forces mobile users to connect to the network to gain access to package files. You might want to do so if you have multiple versions of packages and want to ensure that the mobile user does not attempt to use an old CD-ROM as the source for an installed package. You can implement this setup by using in-house maintenance scripts or the MSICAResetSources script mentioned earlier.


 If you've always wanted to use trickle-type distribution, you might already have it for free! Windows XP and Win2K SP3 include the Background Intelligent Transfer Service (BITS) for supporting trickle down of Windows Update files. BITS works very well—detecting when to resume, limiting bandwidth usage, and so on. By using the BITSADMIN.EXE utility on the Windows XP CD-ROM, you can schedule your own files to deploy from a Web server (using HTTP) to a client through BITS. For more information about how to use BITSADMIN, visit <http://desktopengineer.com/bitsadmin>.

Fixing Existing Unmanaged Sources

Many of you might be feeling a sense of doom in regard to package installation locations because you have not been managing them for the past 2 years that installations have been occurring. You might have thousands of computers that have installed packages from many different locations that might no longer be valid. Or perhaps you are going through a process to consolidate locations or migrate servers—in all of these instances, package source lists might be pointing to incorrect locations.




MSISources is a script written for just this purpose. It allows you to identify which packages you want to manage source locations for in a control file. The control file is then used by MSISources to find packages that should be managed and point them to new locations. MSISources allows drive letters or UNC's and allows absolute paths to be used (for example, [\\Server\Share\acrobat\version5](http://Server/Share/acrobat/version5)). In addition, MSISources allows re-rooting of existing source list paths. This feature facilitates easily consolidating or moving existing source locations without manually coding all the paths into the control file. When source lists are re-rooted, each existing source list is retrieved, and the drive letter or UNC is replaced with the new root location.

 MSISources allows flexible remapping of the source lists of multiple installed packages at one time. For example, by running it in the logon script, you could use it to ensure that every installed copy of Visio on every computer in your company has its source list pointed to a managed location. You can download MSISources from <http://windowsinstallertraining.com/msiebook>.

Designing the Package Distribution Repository

In this section, we will talk about the considerations in designing the package distribution repository where packages will reside so that clients can access them for installations, self-healing, patching, and upgrades. It is actually the scaling of the package distribution repository that creates the challenges and solution alternatives we will be discussing. If you work at a small, single-site company at which all packages can be put on one well-managed server, simply installing all of your packages from this server location would handle most of your repository challenges. However, as soon as the repository must take into account multiple sites, mobile computers, slow links, global organizations, and other similar factors, the repository must be designed and deployed correctly to effectively support enterprise-class software distribution.

In Chapter 3, we briefly touched on the importance of these design activities. The following initial discussion will identify some reasons why organizations that have the option to use DFS might elect to choose other technologies. The remainder of the section deals with alternative configurations to DFS—this information will apply equally to those who could have DFS but choose not to and those who cannot have DFS due to technology or business limitations.

 The guidelines given here should be used carefully. A mix of DFS and non-DFS file sources is a viable solution. In fact, larger sites (as judged by the number of seats) that generally require fault tolerance and load balancing might also have the IT resources to justify a site-level DFS solution. Smaller sites could go without DFS if it posed design, deployment, or scalability difficulties for enterprise-wide implementation.

When to Choose Something Other than DFS

DFS deployment can be challenging to deploy or inappropriate for your computing environment. You might want to consider alternatives to DFS if your situation is characterized by any of the following statements:

- You will not be deploying AD.
- You have few sites or very small sites—DFS deployment in these locations might be overkill simply to support software distribution.
- Global DFS can be difficult to design and deploy. DFS has some roots in the NT 4.0 version of this technology which generates some limiting design recommendations such as 16 DFS servers per DFS namespace and only one DFS root being hosted on any given server. In many global environments, these limitations can prove difficult when DFS is being leveraged for multiple purposes. If your organization is planning multiple AD domains, DFS planning can become even more complex.
- Not all sites in your organization are moving to Win2K and AD before Windows Installer packages need to be deployed. In companies with large overseas contingents or companies with regional budget responsibilities, technology spending tends to be need-based. If desktop management is the only requirement for a given site to move to AD and DFS, there might be push back on these costs.
- An extended desktop deployment to Win2K or Windows XP is planned. During this time, package repository services will still be needed for Windows Installer at all sites and/or for pre-Win2K desktops. For example, if you need to distribute the latest release of your MSI packaged virus software when your company is only halfway through a Win2K deployment, you will potentially need to deploy Windows Installer packages as sites without DFS and to clients that are not Win2K.
- Delayed plans to move to AD can cause Windows Installer software distribution to become a higher priority as software vendor packages and internal packaging initiatives continue to develop. In this situation, the repository must be built before AD is available and be compatible with AD once it is in place.
- Large environments might never be able to support AD for all desktop clients due to the sheer diversity of their networks or server environments.
- A mass project to move to AD and Win2K or later clients is planned, but the project planning is unnecessarily hampered by DFS dependency on AD. In this case, requiring that AD deploy (even in a limited fashion) before desktop computers are upgraded can create costly dependencies for infrastructure migration projects.

DFS Functionality Alternatives

Earlier we talked about four main functions provided by DFS. Table 5.1 shows possible alternatives for these functional areas. A complete discussion of these alternatives is not possible in this book, but specific solutions are discussed in more detail.

DFS Functional Area	Alternatives
Local sourcing (Getting files from the local site)	<ul style="list-style-type: none"> • Managed drive letter • Managed environment variable • Source list management
Universal availability/file location abstraction	<ul style="list-style-type: none"> • Managed drive letter • Managed environment variable
Fault tolerance	<ul style="list-style-type: none"> • Source list management
Load balancing	<ul style="list-style-type: none"> • Complex source list management

Table 5.1: DFS functionality alternatives.

As with DFS, these alternative solutions have some infrastructure requirements:

- A managed drive letter generally requires logon script changes so that the same drive letter is pointed to a local copy of the package repository at all locations. If this approach is extended to mobile machines, additional logon scripting is required to ensure that mobile machines map their drive letters to local resources when they are away from their home sites.
- A managed environment variable might require similar logon scripting changes to ensure that the environment variable used is set correctly for every computer, including mobile machines if traveling support for local sourcing is desired.
- Source list management might be as simple as including additional sources during package deployment, then using the policy to change the search order to network-based sources first. This solution is more static and generally cannot be easily adapted for mobile computer needs.
- Complex source list management can provide a method of mimicking load balancing using an install-time script that ensures that each client install occurs from a pool of possible locations. Instead of using a static source list property, a script must be devised that can dynamically randomize a site-specific list each time a given package is installed. To accommodate mobile users, your list should always include a standard corporate location or site-specific location (environment variable) to ensure that mobile users can get to package sources while traveling to other locations.

Figure 5.4 illustrates that you can use a combination of both source lists and file system approaches to effectively manage the package distribution repository. The illustration is showing all possible methods—any given company should be utilizing the same method for all packages. The Microsoft Office package in Figure 5.4 demonstrates the generally assumed model for IntelliMirror. DFS is set up to handle the main repository requirements—the source list on the client then only needs one entry. If the package is originally installed from the DFS location, the installation defaults for the package source lists work without need for changes or maintenance scripts.



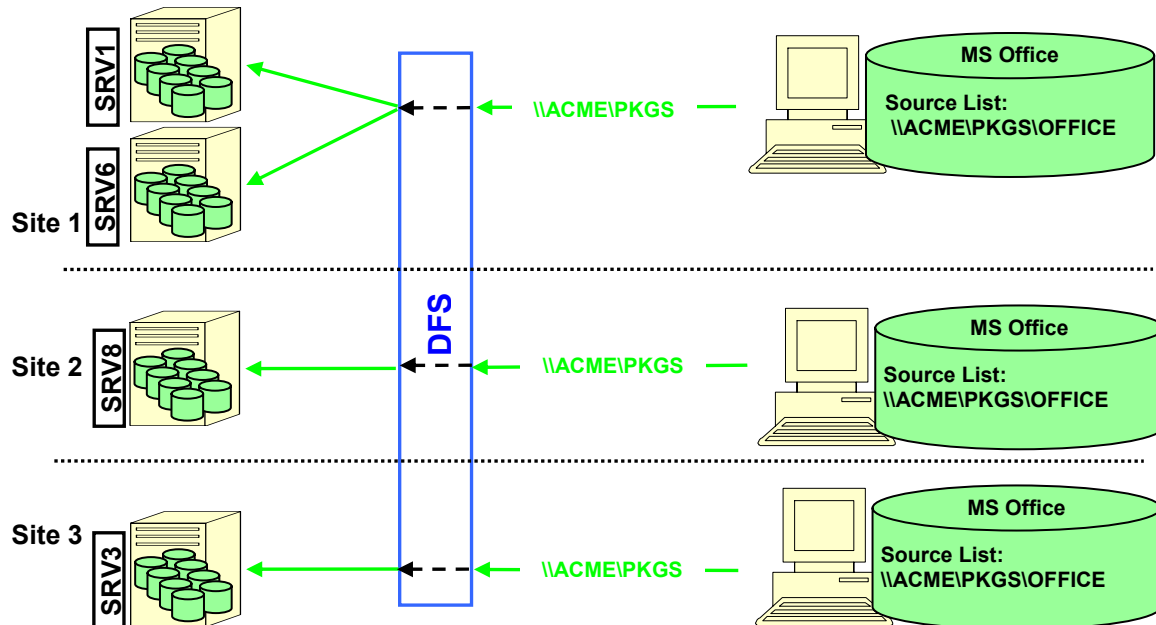


Figure 5.4: DFS for abstracting package source location.

Managed Drive Letters and Managed Environment Variables

Long before Windows 95 came on the scene, many companies accomplished local sourcing and file system abstraction by using a managed drive letter. Each physical location would run specific logon scripts that would set one or more drive letters to local locations. Users at every site would know that the Y drive was for data and that T contained software to install. Drive letters can still bring the benefits of local sourcing and abstraction to the current Microsoft world, and indeed, many companies have never abandoned their drive letter strategies for this reason. Figure 5.5 shows how drive letters can be used to abstract the repository to a local location. As long as packages are initially installed from Q, the source lists can remain unmanaged.

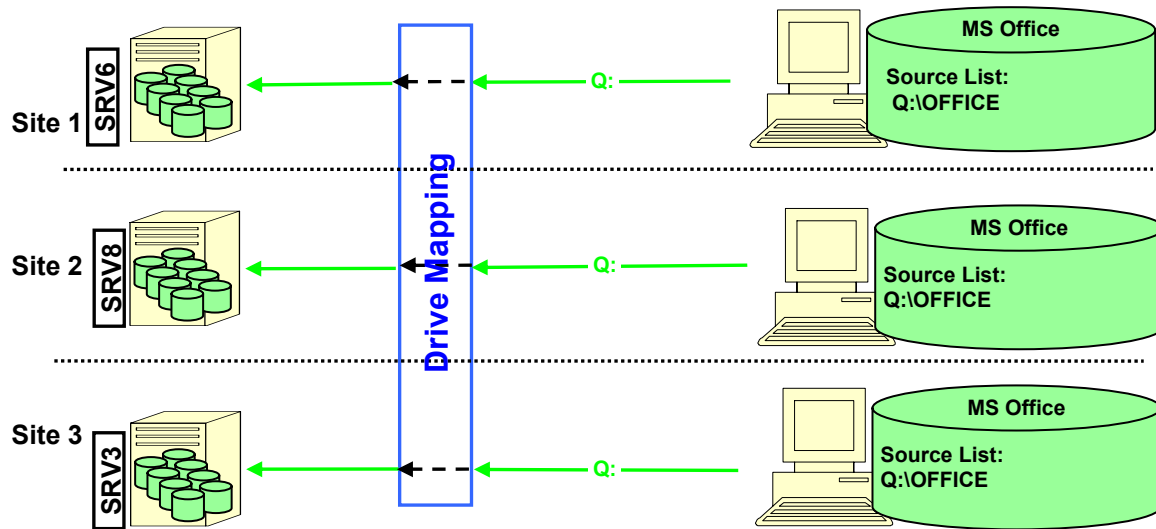



Figure 5.5: Drive letters for abstracting package source location.

Drive letters are a well-established method of mapping to network-based file systems in Windows. Many different file systems are supported. The following list provides some of the benefits of a managed drive letter:

- Eliminate DFS deployment concerns—Earlier we discussed several key concerns with the deployment of DFS. Using a drive letter solves many of these problems, including multiple domain AD designs (with possible multiple DFS namespaces), design constraints in scaling DFS, and small sites at which resources for supporting all infrastructure services might be scarce.
- Relieve DFS domain bounded nature—DFS namespaces are in the context of a domain. Organizations that will be using multiple domains in their AD design will be faced with either creating the same software distribution DFS share in each domain (and deploying packages within each domain to that specific share) or using the DFS namespace of one of the domains in all the other domains, which would necessitate a domain controller with a DFS server running in close network proximity to each client that must use the share.
- Eliminate deployment project dependencies—By using existing managed drive letters, extended Win2K deployments do not hamper the building of a distribution repository. This benefit addresses the problem in which desktop clients cannot deploy until enough DFS infrastructure is present to support package deployment. It also handles scenarios in which Windows Installer packages will need to be deployed to pre-Win2K clients while Win2K infrastructure is trickling through an enterprise organization.
- Enable offsite, offline, and OEM builds—Some workstation build processes require that packaged software be called during the build. When the build depends on resources that are on your company's private network (such as DFS), builds cannot be done while disconnected or by third-party system-building organizations.

- Eliminate source list management during system provisioning—Packages can easily be loaded from any file source connected to the managed drive letter. When the client is deployed to the final location, source lists will work fine without clearing or rebuilding the list because the same drive letter is mapped to the relevant local resources.
- Eliminate drive imaging concerns—Using the same drive letter for software eliminates concerns for deploying drive imaged systems because the drive letter can easily be made available in every location.
- Work with latest technologies—If necessary, drive letters can also work with technologies such as DFS by simply mapping a drive letter to the DFS share name. Drive letters can be used by Group Policy deployment as well.

 Group Policy will automatically resolve the UNC of a network mapped drive and embed it in the software distribution configuration. To override this behavior, use the SUBST command (built-in console command) to map the drive.

- Work with all client OS versions—Domain-based DFS share access is limited to Win2K and later desktop OSs. Additional DFS functionality can be achieved by previous desktop OSs by installing the AD client; however, an enterprise-wide deployment of the AD client can pose its own challenges.
- Work with any network client type—Whether you are using Microsoft or the Novell network client (or a third-party client), a drive letter mapping will work.
- Use any back-end server system/file system—Any file system that a drive letter can be mapped to could be used as a repository, including UNIX systems, mainframe systems, mid-range systems, and any other type of server that can have a drive mapped to it in Windows.
- Turn off drive letters—By unmapping a drive letter, the repository can be made unavailable. When using DFS, if a local repository server is not available, DFS will randomly select one from the list and map to it over whatever network links are between itself and the first successfully contacted server for the DFS namespace. In some environments, this over-the-WAN behavior has a much higher impact on system and network resources than the problem of individual desktops not being able to load software.
- Allow for a universal approach—Because drive letters can be mapped to any network resource as well as any local device (such as CD-ROM, zip drive, hard drive, LS 120 drive) the same drive letter can be utilized for local and remote package loads—keeping source lists consistent across online and offline storage.
- Enable site-level responsibility for repository—Some companies operate with independent IT resources at each site. Standards are used to ensure interoperability. Using drive letters for the repository allows the standard to be loose—letting the site determine how to store repository files. The site can implement local repository services by simply replicating the files from an agreed location and mapping the repository to an agreed drive letter.

- Integrate with many ESD systems—Using a drive letter allows the distribution repository to be utilized by many software distribution systems. The drive letter can also be leveraged via custom pull menu systems and by technicians who can browse the distribution repository directly. SMS 2.0 and 2003 (code-named Topaz) can be configured to use such a repository. Any ESD that can pull files from a drive letter can most likely work with this configuration. Some ESD systems use specially prepared file sets and cannot source files directly from the file system mapped by the end user.
- Use drive letters at times when DFS cannot be used—Suppose you wanted to use the repository for building the OS on workstations and you are using a DOS boot disk—the DOS boot disk can access a drive letter-based repository. You can use drive letters when DFS is not an option.

Figure 5.6 illustrates that the repository can be leveraged for many software distribution activities, including the initial machine build.

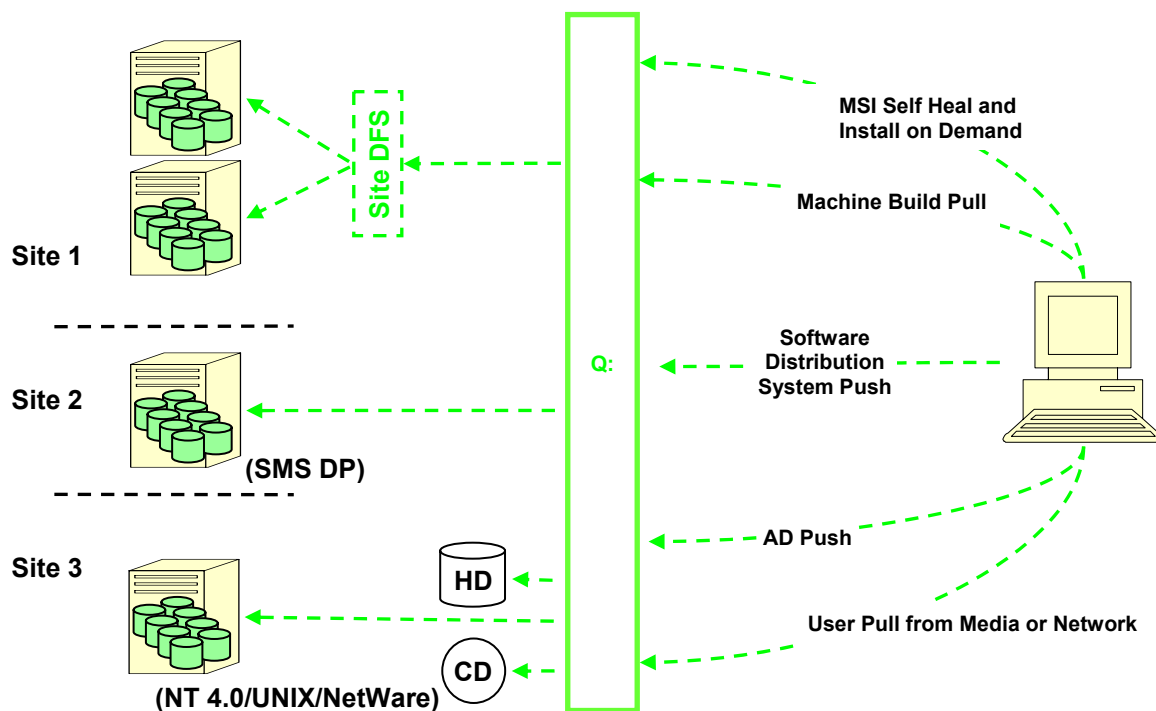


Figure 5.6: Multiple use repository.

Use an Existing Drive Letter

It might be difficult to clear a single drive letter across your organization just for distribution packages; however, you might be able to use an existing drive letter. A new subdirectory can be created on an existing shared drive. With Win2K server disk management, this subdirectory could be served by new, dedicated disks. Server-level DFS can also be leveraged to make the subdirectory point to another server.

Use an Environment Variable in Source Paths

If a dedicated managed drive letter or a directory on an existing managed drive letter does not work for your organization, it is possible that an environment variable could provide the flexibility needed to manage source lists. Source lists will allow an environment variable to be embedded in the source reference, and Windows Installer will properly retrieve the environment variable value before scanning source locations. An environment variable can provide a single reference in all packages that can be set individually for specific sites, regions, or even individual computers. Environment variables can also be set dynamically for special deployment jobs such as packages located in a non-standard location for security reasons.

The flexibility benefits of environment variables come with some challenges that are not present with a drive letter solution. Some ESD systems might not be able to invoke a package using environment variables in the command line. In many cases, you can work around this drawback by having the distribution system call a batch file. Figure 5.7 shows how an environment variable can abstract package source locations to point to any type of file system resource at local sites. Note that the source list must be cleared of the initial install location and the environment variable location added.

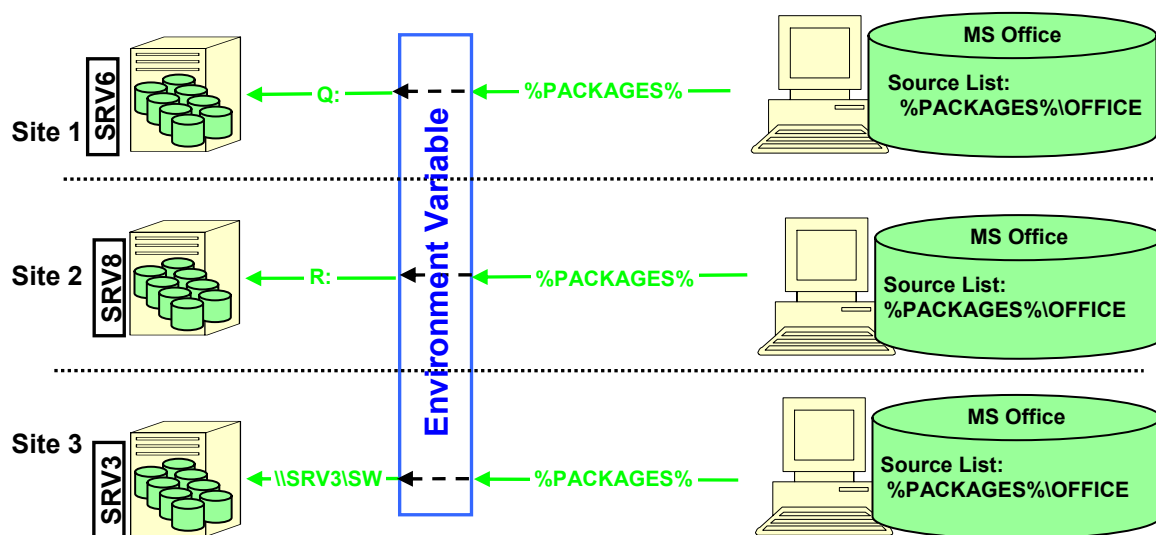



Figure 5.7: Environment variable for abstracting package source location.


The use of environment variables also requires some source list management through which a drive letter strategy can be devised that does not require source list management at all. Source list management is required because the first source list location is the location from which the package was originally installed. This original location must be resolved to a literal disk location by Windows Installer in order to get the package installed on the system; it will always contain a direct reference to the location of the package, not any environment variables used when calling the package location. The absolute location must be cleared from the source list in order to rely completely on the environment variable.

Drive Letter and Environment Variable Infrastructure Requirements

Universal drive letter and environment variable solutions have some infrastructure requirements that DFS handles automatically. DFS automatically provides information about where in the physical network a client is located. DFS does so dynamically, regardless of how the client connects or whether logon scripts run. DFS then maps the client to local servers for the given domain-based share. When creating a custom solution, some or all of these dynamic capabilities might need to be mimicked if your environment requires them.

Many organizations already have well-established logon script designs that can easily suit repository mapping needs as well. If you have deployed AD, you can use ADSI scripting to learn the physical location of the computer and map the local repository based on that information. Without AD running, a custom solution must be devised to assist with pinpointing the physical location of computers on your network. The IP-based cool tool solution that will be discussed shortly has an advantage over ADSI scripting in that it can match multiple hierarchical location contexts such as Europe AND France AND Paris and Mercer Building. The follow two cool tools can help you by providing some key elements to a scripted solution.

 SiteSense.vbs is a script for detecting a computer's exact physical location by determining which subnet the computer is on. Once the computer's physical location is pinpointed, it can be used to map a local version of the package repository. This functionality is similar to AD's built-in sites functionality. However, SiteSense only uses the computer's IP address, so you can use it with any version of Windows, any network OS, and any directory service. You can download SiteSense from <http://windowsinstallertraining.com/msiebook>.

 VPNRASLogonHook handles several cases in which logon scripts do not reliably execute. Logging on to the network from a VPN or RAS generally does not run the logon script. As of the release of Win2K, dynamically plugging in a network card also allows users to access network resources without running the logon script. VPNRASLogonHook uses WMI to detect when a network connection occurs and runs the logon script if it is available. You can download VPNRASLogonHook from <http://windowsinstallertraining.com/msiebook>.

Package Source List Management

Active source list management can help ensure that source lists correctly point to local package repository locations. This approach does not dynamically allocate local sources for mobile computers, but can provide an effective approach for desktop machines.

Some organizations use MSISources, which was introduced earlier, on the first boot of a newly built computer. A primer script detects the computer's physical location and remaps all managed packages to the local repository server. This script could be run during computer moves or scheduled periodically to correct source lists for desktop computers that change physical locations.

When to Choose Something Other than FRS

The Win2K version of FRS is capable of replicating reasonably sized logon script directories. Replication loads that are more intense in their size or replication frequency can be quite difficult to manage with FRS. The following list provides the primary attributes of FRS that make it unsuitable to the loads generated by large-scale replication:

- **Multi-master replication methodology**—A multi-master model means that each server copy of the replicated files can be independently changed and the changes will be automatically replicated to all other servers. This model is a big plus in the area of directory services but creates difficulties in software distribution where there is generally assumed to be a single master copy of software distribution packages that should be replicated to every other location.
- **Full mesh replication topology**—I have referred to this topology as “full mess” replication topology. When a set of servers are configured to keep a single, replicated copy of files, each server maps directly to every other server, regardless of physical network topology. Servers at end sites of the physical network will attempt to communicate with each other to exchange files that have changed.
- **Unrestrained automatic operation**—Once configured and turned on, replication happens automatically, immediately, and has no performance governors between physical sites. Thus, replication cannot be constrained in terms of bandwidth usage or scheduling times of day when distribution should occur.



Rudimentary editing of FRS replication topology and replication schedules is possible but a brief encounter with the editing utilities reveals these capabilities to be ill fitting in enterprise environments.

- **Lack of human-readable logging**—Although FRS has logging for its internal purposes, these logs use globally unique identifiers (GUIDs) to track multiple versions of the same named file. This system essentially makes them unusable for tracking replication problems. In addition, there is no audit tracking for finding out which user IDs might be responsible for accidental replications that can have a high impact on network and server resources.


Even when used in a LAN environment, the flurry of activity created by FRS replication might generate uncontrolled peak loads for networks and distribution servers.

FRS Alternatives


There are several alternatives to FRS that might be considered for replication of the package distribution repository. Specialized replication software manages replication by making sure that bandwidth usage can be controlled through scheduling and throttling. Replication topology can also be designed to coincide with physical network layout and individual link utilization.

Many enterprise-level ESD systems have robust built-in replication. Bandwidth can be scheduled and throttled and topology can be managed. It is unlikely that an ESD can be cost justified based on replication needs alone; however, if you already have one, you might be able to leverage it for your repository needs.



 Any replication system (or the underlying network) can be overloaded if your package release strategy does not account for long-term network bandwidth requirements and server disk space requirements. For instance, if your methodology allows software application owners to request package updates at any time and then build and replicate an entire upgrade package for every change, it can generate big strain on replication bandwidth and the replication management system. If you also keep older copies around for too long, disk space can be strained.

Many companies have used NT or later shell scripting (.CMD or .BAT) and the resource kit utility ROBOCOPY to implement their own replication scheme. Implementing replication with scripting rather than a product is a challenging exercise—especially in a large organization.

 Tim Hill's *Windows NT Shell Scripting* (New Riders Publishing) contains a complete and robust sample script called REPL.BAT that uses ROBOCOPY to emulate logon script replication. This script could serve as a starting point for creating a script-based replication system.

BITS was mentioned earlier as a cool tool to perform trickle replication to remote desktop clients. It might also be used a method of low-impact replication between servers on your network. You will need to be running Win2K SP3 on your servers to use BITS directly on the server. In addition, the file source must be an HTTP server. If Win2K SP3 is not an option, you could script a Windows XP desktop to transfer the files and then move them to a server. BITS might not be a good alternative if you are transferring a large number of files—for instance, if all your package sources are administrative installations.

Directory Structure Issues

Up to this point, we have been discussing the repository location in terms of the root location—the location where packages are stored. This root location would be a UNC or drive letter. We could simply put all our packages at the root of this location if they all had different names. However, it is important to choose a directory structure for the repository that will be flexible and be scalable.

For pre-Windows Installer package repositories, some organizations have used a convention of having the latest package version in the same directory so that it is easy to locate. The location of the latest version might need to be known by technicians browsing the repository and by automated build processes and pull menu applications. Windows Installer adds a new wrinkle for this approach. When a package is installed from a network location, Windows Installer will return to that location to find the MSI file and the associated software application files. If Windows Installer determines that the MSI file is not the same as the one that was installed from that location it might refuse the location as a valid source.

This occurrence might not present a problem for applications that have a limited distribution scope (number of seats, number of sites) because they can all be updated to the latest version in a fairly short time. If, however, you are updating software in an organization of 30,000 desktops, the sheer time it takes to update every one might result in problems with self-healing or new computer builds if the new version is pushed to all distribution servers as the first step in deploying the updated package. If local sites have responsibility for triggering desktop installations, the time lag between a centralized push replication to distribution points and local site installation on clients might be unpredictable. This creates the same problem with regard to self-healing and other Windows Installer activities not completing due to the wrong version of the source.

You might want to consider a scheme that allows a distributed package to reside in its destination directory indefinitely and use another approach to identify the latest version. For instance, a batch file could be created called `current.bat` that would contain the command line to use for the current production version of the package. This batch file could be read and parsed or simply executed by automated processes and administrators.

Directory Structure Considerations

This section will consider several key Windows Installer issues when designing a directory structure for the package distribution repository. The following list offers some of the high-level considerations:

- If you are using administrative install shares, multiple versions of the package cannot have the same directory root. The reason is that each version will have files of the same name and directory location—the install share would end up being a mix of files from multiple versions. Windows Installer does not check the actual version of the file on disk, so an installation could be performed from this share, but the software might be inoperable and self-healing behavior could be erratic.
- Schemes that are too shallow could take a long time to enumerate, which causes delays when browsing the directory or enumerating it with maintenance or pull menu applications.
- Schemes that are too deep might not fit on offline media (if replication to offline media is a part of your scheme).
- If you plan to use *path rules* with software restriction policies, make sure that your chosen directory structure does not result in overly complex rules.


The following example provides a good place to start with directory structure:

```
<root>\<pkg_id>\<version>\<filename>.msi
```

This structure allows each package version to exist simultaneously, which prevents problems with self-healing and install-on-demand during lengthy upgrade cycles. It also allows multiple administrative installs of the same package because each administrative install uses the `<version>` directory as its root. Some type of batch file or other file can be stored in the `<pkg_id>` directory level to ensure the current version. This scheme can also work well for storing informational data such as instructions, documentation, logs, and so on. The following example shows an alternative to the previous scheme:

```
<root>\PACKAGES\<pkg_id>\<version>\<filename>.msi
```

This scheme allows more portability and flexibility for where the software share is placed because it does not assume that it owns the root of the file location. For instance, if you need to place the distribution repository on an existing drive letter or DFS namespace, this scheme would be helpful. Even if you currently have a dedicated root location, consider what happens when you scale up your implementation. Also consider whether other IT departments might want to piggy back their needs on your repository strategy; using the `PACKAGES` directory allows multiple functional sublocations under the root locations.

 You might be tempted to store the command line for the “current version” in SMS or some other distribution system configuration. This information will be much more flexible and accessible in a batch file because it can be utilized by less capable automation activities such as initial computer builds and by technicians without the need to rifle through the distribution system to learn the appropriate command line.

The following schemes are based on traditional non-Windows Installer approaches, so they are bound to come up during repository design. These schemes might not work well:

```
<root>\<pkg_ident><version>\<filename>.msi
```


When the version is attached directly to the directory level for the package identifier, the number of directories under the root location might become difficult to manage. This approach also makes it more difficult to establish a method of identifying the current version of a package because the files that do so must reside in the package root directory.

```
<root>\<pkg_ident>\<version><filename>.msi
```

When the version is appended to the file name and stored in the same directory as other versions, administrative installations cannot be used.

Administrative Installs

Chapter 3 covered administrative installs; in this section, they will be explored specifically in regard to considerations of building the repository. Most likely, you will have some type of administrative installs in your repository. The consideration is whether to make it a standard practice to make all of your network deployed packages into administrative installs. There are several issues revolving around using administrative installs in your distribution repository.

 In Chapter 3, we covered some reasons why you might want to create administrative installs. The following list provides a quick review:

- To present properties (such as TRANSFORMS) for execution when the MSI is double clicked.

- To use served applications.

- To pre-activate Microsoft products (and to deploy patches to them).

- To reduce the required server-to-server replication load by patching admin shares with updates.

- To extract only the needed files from your installation media.


Using administrative installs in your distribution repository can be a good long-term implementation decision if you have site-to-site bandwidth constraints because you can patch administrative installs to reduce the server-to-server replication load. Only patches need to be replicated and executed. There are some qualifiers to this approach:

- The initial replication load is much higher than compressed source files.
- The extra effort to build reliable patches is significant.
- An automated or manual activity must execute the patch on all distribution points—something that does not need to be done with straight replication of complete packages.
- If you want to keep previous versions intact for self-healing to work properly, you must locally copy the administrative directory to a new location before applying the patch.
- Offline media replication (sending a tape or CD-ROM) might be less work than a full-scale patching approach.

The following list provides additional considerations (aside from a patching strategy) that should be a part of your discussion when designing a distribution repository that includes administrative shares:

- You will most likely have to deploy some administrative installs, even if your strategy is to favor not using them. For instance, Microsoft Office products assume that network distributions are always done from administrative installations.
- If you use administrative installs for all of your packages, disk space requirements will be much higher—potentially double that of compressed .cab files or .cab files stored inside the MSI.
- The number of files being replicated will be much higher with administrative installs. It will be important to ensure that your replication technology is up to the task.
- If you are using code signing of MSI packages for your packages, you will need to replicate the signed version. Patched administrative installs will need to be re-signed.
- Directory structures for a repository might become too deep to fit on offline media.

This list might make it sound as though administrative installs are not a good idea. Such is not the case. Rather, because most traditional package repository schemes use some type of compressed source (setup.exe files), it is easy to overlook the many additional considerations of using Windows Installer administrative installs.

 AD cannot deploy patches directly to clients, which seems to have led Microsoft's software deployment guides to emphasize administrative installs for network-based deployments. If patching administrative shares is not a part of your deployment strategy, you can effectively use compressed source files (external or internal .cab files) to manage your distribution repository.

Repository Availability Service Level Agreement


The magic of Windows Installer self-healing implies to many end users and business departments that their applications will never break, no matter how far mobile computers travel from their home base. As you build a repository strategy, you will learn that it would take an immense amount of human and technological resources to keep that promise in every single usage scenario in a global company. It is, therefore, important to establish a service level agreement (SLA) that sets reasonable bounds around when users and business departments can expect magic and when they will receive a prompt for a package location.

For example, perhaps self-healing can be expected to work when mobile users are on their home continent. For some organizations, self-healing might only be expected to work when mobile users are at their specific physical site. Whatever the case is at your company, be sure to go through the design activity to balance your implied or formal SLAs with the IT resources that are required to get the job done.


Package Deployment Technology Planning

AD and Group Policy provide the capability for an out-of-box Win2K implementation to perform software distribution to desktop computers. IntelliMirror technologies can provide a workable software distribution solution for workgroup environments or companies that have fewer sites that operate fairly independently. There are two high-level indicators that you might need more than IntelliMirror can provide:

- If AD and/or Win2K or later clients are not in your future or are in a slow-burn deployment plan, AD deployment capabilities might not help you in the immediate term.
- If you are accustomed to a full-featured software deployment system, AD might not be able to meet your current SLAs.

 Many companies assume that they can eliminate the cost of their current software distribution system when moving to a full deployment of AD and Win2K or later desktops. Once this idea gains momentum, it can be difficult to turn back. I advise that you insist on a thorough feasibility study of whether IntelliMirror is up to the challenge of your current distribution system requirements before this momentum starts to build.


IntelliMirror has limitations, some of which stem from the fact that it is piggybacking on a directory service, others result from the simple model of deployment scenarios that IntelliMirror targets. It can be difficult to get a comprehensive view of how these limitations stack up against existing and future SLAs.

 SLAs are a critical part of technology design activities. SLAs can be formal or informal. For instance, you might have a signed document stating that a user will receive a software distribution within 24 hours of requesting it. Informal SLAs can carry as much or more clout than formal ones. It might just be a foregone conclusion in your company that software distributions always occur at night to prevent user interruption—even if there is no signed document to back this assumption, everyone intuitively understands what would happen if distributions were to be done during working hours. Assessing whether a technology can meet your formal and informal SLAs is a critical step and can be an early indicator of whether your implementation will be ultimately accepted by all stakeholders.


IntelliMirror Fine Print

IntelliMirror has limitations with regard to how it can classify objects because it uses the directory service to do so. It also has very limited functionality for non-MSI packages. In the areas of reporting, logging, and scheduling distributions, IntelliMirror is lacking. The following bullet points provide a fairly complete list of these limitations and their potential effects on your software distribution design:

- The ability to target users and computers for software installation is limited to AD's OUs. This hierarchical container classification only allows a single container membership for each user and computer. Many ESD systems allow for objects to be classified in multiple groupings or containers. In AD, container membership is arbitrary, while many ESD systems allow targeting by inventoried hardware information. IntelliMirror allows filtering of distribution targets by security groups, and in Windows XP, IntelliMirror allows filtering by WMI queries executed on clients; however, this flexibility comes at the cost of multiple layers of target filtering that are evaluated at different stages of distribution.
- Most of AD's inheritance is within the bounds of a domain. Distribution targets, security groups, GPOs, and delegated authority are all relative to the domain they are configured within. If your company will have multiple domains, many distribution configuration activities will need to be repeated for each domain. In addition to being a lot of work, this requirement can lead to quality issues when detailed work such as this must be repeated many times.
- AD's classification scheme (sites, domains, OUs) is leveraged by a company for many IT management and business unit purposes. With so many requirements that stem from outside of software distribution, the directory structure naturally becomes inflexible to all but the most pressing needs. Because ESD systems focus on software distribution, many times their underlying classifications schemes can be adjusted or reworked with no implications for other parts of the IT or business organization. In addition, excessive fragmentation of directory nodes (OUs) for the purpose of distribution targeting can lead to an unmanageable directory configuration.
- Non-Windows Installer package distribution is difficult and severely restricted in IntelliMirror. Non-Windows Installer packages cannot use elevated privileges and they can only be published to the user. Thus, a user must visit the Add/Remove Programs applet in the Control Panel to kick off the distribution. Although MSI wrapper scripts have been developed by Microsoft and others to distribute items such as service packs, they are definitely a workaround use of Windows Installer technologies and can be difficult to learn and maintain.
- IntelliMirror uses a simple methodology for triggering distributions. During the boot up and logon process, users are not using applications and cannot have locks on key system resources or applications that need to be updated. For this reason, IntelliMirror only distributes software at that time. This activity can, however, result in peak loading for distributions because many users logon or boot up at common times.

 Software distribution policies differ from other Group Policies in that they DO NOT refresh every 90 minutes—they only occur at boot up (for computer-targeted distributions) and logon (for user-targeted distributions).

- The inability to schedule distributions to desktop clients is a result of the simple methodology mentioned in the previous point. The implications of this limitation vary by company. Some companies use scheduling to prevent interruption of business activities (for example, overnight distributions), other companies use it to offset the load placed on the network and distribution servers. To help with this, IntelliMirror only advertises software when it is assigned to users. However, as discussed in Chapter 3, there are several key reasons why software should be installed per computer rather than per user.

 Windows Server 2003 will allow users to have software completely installed when assigned rather than only being advertised.

When to Consider Alternatives to IntelliMirror Deployment

The previous considerations might be difficult to evaluate for any given organization. The following list enumerates attributes of your technology environment and key distribution system features that might indicate that you need to choose another technology for software deployment:

- Large-scale implementations with large numbers of clients and/or many sites might find that the lack of logging restricts their ability to deliver distribution services in the same fashion traditionally expected by business units and end users.
- Desktop management and software distribution activities have formal SLAs associated with them that require a distribution success rate or exception reporting. Environments with formalized service levels might find IntelliMirror unable to provide needed detailed and summary reporting to prove the service level is being met.
- Homogenous environments that will always contain many different directory services, desktop OS, and server OSs might find the limitation to AD to be too restrictive.
- The inability to schedule distributions can have a surprising number of cascading effects on other distribution activities. Ensure that this factor is evaluated thoroughly.
- Organizations that want to manage servers as well as desktops with the same system will find that IntelliMirror's focus on interactive use (reboots and logons) and Windows Installer technology make it a difficult fit for server distributions.
- Companies that are coming from distribution systems that have a mature set of distribution capabilities (including SMS, Novell's management products, or any of the integrated desktop management suites available) might be surprised by what they would be giving up to move to an IntelliMirror-only system.

This last section paints a restrictive picture for organizations that choose IntelliMirror for their package deployments. IntelliMirror is a very respectable solution for the class of businesses that have traditionally been targeted for Microsoft's Small Business Server offering. For these companies SMS is definitely overkill.

However, this ideal implementation target for IntelliMirror distribution has been stretched by unrealistic expectations of upward scalability and cost effectiveness for enterprise environments. If you feel that IntelliMirror might be suitable for software distribution in your environment, be sure to do your homework.

Summary

Platform 2000 (Win2K servers + AD + Win2K desktops) was presented to the market with intricate interdependencies that can make it difficult to understand how to extract value from specific subassemblies such as Windows Installer. Hopefully, this chapter has unraveled a few mysteries and given you some alternatives to think about if you are implementing Win2K management technologies. This chapter is the last to be written by me, as Jeremy will be wrapping up the book with a final chapter about distribution systems. I hope to cross paths with you at conferences, in my training class, and on the Internet!