realtimepublishers.com™

The Definitive Guide™ To

Windows Software Deployment

LANOVATION®

Making Complex Technology SIMPLE *Since 1985*

Chris Long

Chapter 6: Deploying in the Enterprise

I've talked so far about creating a package, how to get the package out, and how to handle deployment to remote users and machines. The focus so far has been from the application point of view. It's all been about the mechanics of the process and how to make the deployment do what you want. I've talked about testing methodologies and about some of the different technologies for delivery and packaging. Now along comes this chapter about deploying in the enterprise, and you're probably wondering "Hasn't this topic been discussed?" Well, sort of.

The discussion up to this point has been centered on the technical aspects of getting the first package out. The goal of the early stages of the book was to provide enough data to get your packages up and out that first time. Although there have been casual mentions of doing deployments in repetitive forms or multiple deployments, the main theme has been on just one package at a time. Software deployment is not a one time function nor does it entail only the technical details. Windows software deployment is an entire process that must be repeatable in both the technical and procedural arenas, and the process must show a reduction in effort (and basic expense) as the numbers of deployments continue.

In this chapter, I'm going to delve into the area that all technologists dream about: Process, procedures, and documentation. To make it sound a little more technical, I'll refer to this topic as PP&D.

PP&D is just as critical to software deployments as the actual creation of the package. Remember that the deployment system you use within your enterprise must be repeatable by others, not just by you. The steps to create a package, the standards to use, the SLAs, and the methods to use must all be known to the technical staff responsible for the work. Management must be aware of the process as well as informed about the work you've already done. Because you most likely enjoy taking a day off here and there and dream of an actual vacation that is not interrupted by work calls, documenting everything to the level that others can follow is critical. It's all about making your life just a little easier.

Before I get too deep into the chapter, let's define a few terms that I'll use frequently:

- Enterprise—This term has come back into vogue in recent years. For this chapter, it will be interchangeable with business unit, company, division, department section, or any other term you like that describes a group focused on making the business run. It's a good generic term, as it can cover the large company as well as the mom-and-pop shop.
- Shops—The IT department. It can be one person that does everything or several thousand specialists. Adjust the term to fit your situation.

Does Size Really Matter?

Does size really matter? Sure it does. When you are talking about motorcycles, bench pressing, monitor viewing sizes, CPU speed, and RAM. When you are dealing with software distributions, the logic and methods applied to a distribution is the same whether you apply it to 20,000 machines or 200. True, the time factor, distribution points, and network issues are increased as the number of end points increase, but the basic structure and process remain the same.

As a recap, remember that in smaller shops in which distributions are to a few machines (say a couple dozen), getting things done is generally faster. Notice I didn't say better—just faster. You have one person that does the packaging and testing and distribution. If something happens to the distribution, everyone in the enterprise knows where to go for resolution. With only a few machines, the packaging may be skipped in favor of the old faithful shrink-wrapped Sneakernet distribution with manual adjustments at each point.

In larger shops, the same distribution that takes only a day or two in a smaller shop may take longer. The length of time depends on the processes, procedures, number of people involved, and the company's political environment. If something happens to the distribution, the solution may take a little longer as different groups will be involved in finding a solution. However, the entire process becomes cost and labor effective when there are multiple remote client end points.

As technicians, chances are you have a stronger interest in the bits and bytes of the distribution than the political side. Left to your own devices, the distribution would roll out without issues and be done rather quickly. The larger the enterprise, the more others need to be involved in the distribution—it is in these situations that things start getting sticky. Change management, version control, distribution notifications, corporate communication, business unit negotiations, Help desk staff, and support technicians all want a piece of the distribution knowledge. Rightly so, because they will all be impacted in some form. Of course, each of those departments will want documentation in a specific form about what is going to happen and how to support it.

Documentation

No matter the size of the enterprise, your distributions should involve some form of documentation. I can hear the common statements being muttered already: "My shop supports 15 machines and I'm the only one in IT, so documentation is a waste of my time. Besides, real techs don't need documents or manuals."

Sorry to burst your bubble, but RTFM (Read the Flipping Manual) does come in handy at times. Small shops share the same problem as large shops—turnover. You may opt at a later date to move on to other opportunities with other companies, but someone has to support what you are doing today. As large as the computer field is, the community itself is small. You will run into the same people time and again at some point. You may not have met the tech that was in your position before you, but you know who it was. Thus, after you leave your current position, the next person will know about you. All those complaints and problems you have because the distributions and packages were not documented can be corrected by doing some simple documentation as you perform the rollout.

Even if you don't leave your company, a wise old techie once told me that if no one else can do your job, you will stagnate and never move from that job. I had a great time doing DOS/VSE (mainframe stuff) in the early 80s, and at the time, could not imagine doing anything else. It's a little hard finding those kinds of jobs now. Make sure someone can cover you so that you can move on to other technologies as they come out. PP&D can help. OK, enough of Career 101.

Documentation is the backbone of computers. However, not many people like to write it—the developers don't have the time, nobody likes the way someone else wrote it, and the documents never have the data you need to solve your problem. Look at how many books and manuals and magazines and Internet articles are available for any major OS. Everyone has a better method of creating the documentation and solving a problem. So how do you create a document about a rollout so it appeases everyone? You don't. Don't even try.

Not exactly the response you were expecting, is it? Documentation is important. It has to be done. The larger the enterprise and the more groups that are involved, the more important the document becomes. In the first part of this book, some samples were discussed that showed the importance of documenting everything about a rollout. If you follow those guidelines, chances are that the necessary answers will be available when someone needs them. However, if you are a small shop, you may not have the time to devote to a detailed documentation endeavor. So where do you draw the line? How much documentation is needed and how much is too much?

Look at documentation as way of preventing interruptions in your non-work life. Vacations are only vacations if you don't really need to check in everyday to solve problems. If your company has standards already, you only need to fill out the necessary forms. If you are creating the standards, create documentation based on what you would like to see about a deployment you had nothing to do with. If you satisfy your thirst for deployment details in the document, the document is correct. Just don't assume that all facts are known to everyone. There are two general types of documentation you will need to provide: package and distribution.

Package Documentation

There is a point in documenting a package at which more time is spent than what is needed. You can judge a rollout by the size of the supporting documentation—the thicker it is, the hairier the rollout. Of course, you need to decide what you want in your application documentation. I suggest a cover sheet, conflict data, and content details.

The application cover sheet should cover these basic categories:

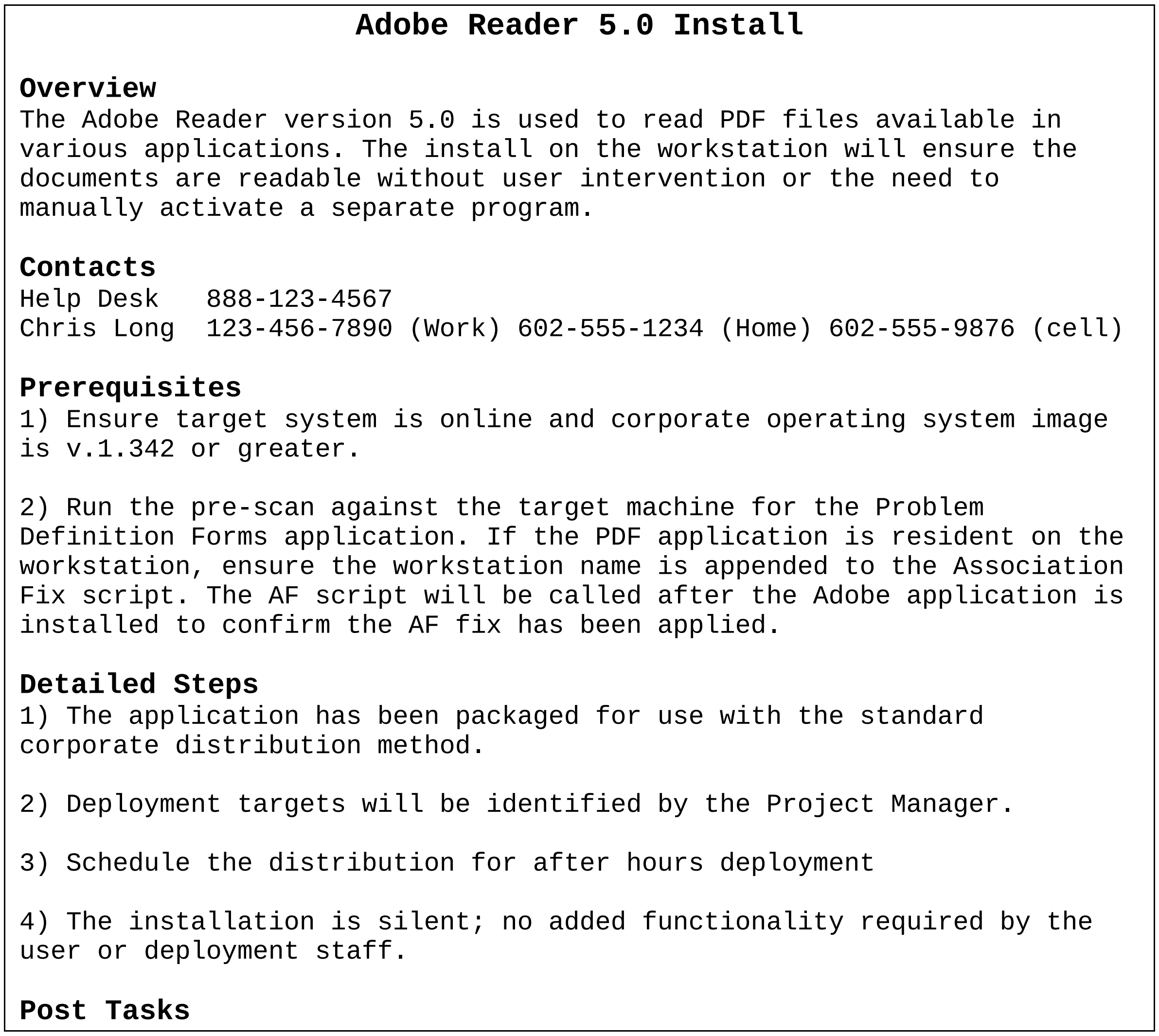
- Overview—The overview is a short one or two paragraph description of the application, where it goes, and what it does. Aim this description at the non-technical level so that everyone understands what the application does.
- Contact—This information explains who to call for problems or questions about the application and package. If the application is a custom piece supplied by a vendor, you should provide the vendor contact data, including any contract items, costs, or phone numbers. Be sure to specify if an internal escalation path should be used first; otherwise, you might find your allocated incidents being used to answer questions on how to change the desktop wallpaper.
- Prerequisites—What is needed on the target systems to get the application? Are there any customizations to be done to the application or package prior to distribution?
- Detailed Steps—Detail in short and concise sentences what needs to be done and when. No fluff, no explanation, simple step by step instructions. If the application is set to follow standards and is silent, say so. Screen shots are a good idea, within reason. Don't provide shots of the Welcome screen, only the pertinent shots of decision points or where data needs to be included. Remember that the document will be used in the heat of battle, so clear and concise information is ideal.
- Post Tasks—Identify anything that needs to be done after the application is installed. Use the step-by-step method again.
- Troubleshooting—In this section, list where to go to find solutions. If your package creates logs, list where they are located and what each log will do for the troubleshooter. If you know of any other places to look for help, write them down. Anything to help the

troubleshooter. You notes during testing should help write this section. It's okay if this section is blank because you don't have any information.

- Process Overview—If you created a home-grown process, detail it here. What does each step of the script or .exe file do?
- Known Issues and Errors—Sometimes an application is rolled out with known bugs. List them here.
- Optional Items—Detail a checklist of steps to follow and provide script listings.

Figure 6.1 shows a sample cover sheet for a deployment of Adobe Reader 5.0. As you can see, the document is short and simple and provides basic information. It assumes that the reader knows the standard distribution process. This document could be expanded by listing the various scripts in detail or providing any additional requirements needed for the rollout. Because Adobe Reader 5.0 is free from Adobe, the vendor data was not listed. The point is that the overview document is a good spot to start with and provides the cover data for the application rollout.

Corporate use of Adobe Reader may require licensing or other legal considerations. Be sure to contact Adobe (http://www.adobe.com) for details. This application was used as an example only.



Adobe Reader 5.0 Install

Overview
The Adobe Reader version 5.0 is used to read PDF files available in various applications. The install on the workstation will ensure the documents are readable without user intervention or the need to manually activate a separate program.

Contacts

Help Desk	888-123-4567
Chris Long	123-456-7890 (Work) 602-555-1234 (Home) 602-555-9876 (cell)

Prerequisites
1) Ensure target system is online and corporate operating system image is v.1.342 or greater.

2) Run the pre-scan against the target machine for the Problem Definition Forms application. If the PDF application is resident on the workstation, ensure the workstation name is appended to the Association Fix script. The AF script will be called after the Adobe application is installed to confirm the AF fix has been applied.

Detailed Steps
1) The application has been packaged for use with the standard corporate distribution method.

2) Deployment targets will be identified by the Project Manager.

3) Schedule the distribution for after hours deployment

4) The installation is silent; no added functionality required by the user or deployment staff.

Post Tasks

```
1) Confirm the pre-scan collection of workstation with the Problem
Definition Forms application have had the Association Fix applied.

2) Check the Central Reporting Server under \\CRS\Reports\Adobe\AF to
confirm.

Troubleshooting
1) Installation logs are located at C:\Temp\Install\Adobe50
Shows installation steps and time of execution. Return codes are
included.

2) Application logs have been modified to reside in
C:\Temp\Apps\Adobe50
Any application created messages not available in the Event Viewer
should reside here. Application specific.

3) http://www.adobe.com/support/readguide.html
Knowledge base for Adobe Reader

4) www.microsoft.com
Review the knowledge base for operating system generated errors.

Process Overview
The installation will create the Adobe Reader application and insert it
into the Default User hive. Installation is silent

Known Issues
1) If the association to the *.PDF file extension is disabled, the
application will not be called up. Recreate the association for the
impacted user.

2) Conflicts between standard Adobe PDF extension and the custom
created Problem Definition Forms application may cause conflicts to
some users with both applications on their system. Manual startup of
the Adobe files is recommended.
```

Figure 6.1: A sample deployment documentation cover sheet.

Automated methods of creating the supporting application documents are another good reason for a commercial application distribution system. Such a system can provide some of the data for you. Save the conflict-resolution details in a DOC folder that goes with the application data. Any logs that are generated during the package creation should be saved and stored with the application source files. This information gives you a reference to visually review when troubleshooting problems months later—or to hand off to someone else.

Listing 6.1 shows a copy of the file generated by Conflict Checker Professional (included in Lanovations' Prism Deploy product). Any conflicts listed would be investigated during testing to ensure no problems occur on the workstation images. Including this file in the application package provides added data for troubleshooting. The generation of the conflict file took less than a minute to create on the test system. Add another couple of minutes to save the file off to the application directory, and inside of 5 minutes (for this example) you have a record that will be with the package for the life of the application in your organization. Five minutes invested during the package creation that would take much longer to recreate months down the road.

```
System File Conflicts

---------------------
C:\deploy\AdobeFiveO_1.PWC
 AceLite.dll
  397312,04/16/2001 04:39:02 PM,1.2.0.1,"C:\Program Files\Adobe\Acrobat 
5.0\Reader","C:\deploy\AdobeFiveO_1.PWC"
  397312,04/16/2001 04:39:02 PM,1.2.0.1,"System\Adobe\SVG 
Viewer","C:\deploy\AdobeFiveO_1.PWC"
 Agm.dll
  1138688,09/05/2001 02:10:34 PM,4.4.26.1,"C:\Program 
Files\Adobe\Acrobat 5.0\Reader","C:\deploy\AdobeFiveO_1.PWC"
  1138688,09/05/2001 02:10:34 PM,4.4.26.1,"System\Adobe\SVG 
Viewer","C:\deploy\AdobeFiveO_1.PWC"
 Bib.dll
  147456,04/16/2001 04:39:02 PM,1.0.20.1,"C:\Program 
Files\Adobe\Acrobat 5.0\Reader","C:\deploy\AdobeFiveO_1.PWC"
  147456,04/16/2001 04:39:02 PM,1.0.20.1,"System\Adobe\SVG 
Viewer","C:\deploy\AdobeFiveO_1.PWC"
 CoolType.dll
  1441792,07/24/2001 08:02:54 AM,4.4.26.1,"C:\Program 
Files\Adobe\Acrobat 5.0\Reader","C:\deploy\AdobeFiveO_1.PWC"
  1441792,07/24/2001 08:02:54 AM,4.4.26.1,"System\Adobe\SVG 
Viewer","C:\deploy\AdobeFiveO_1.PWC"
 nppdf32.dll
  103344,09/10/2001 03:47:38 AM,5.0.5.452,"C:\Program 
Files\Adobe\Acrobat 5.0\Reader\Browser","C:\deploy\AdobeFiveO_1.PWC"
  103344,09/10/2001 03:47:38 AM,5.0.5.452,"C:\Program Files\Internet 
Explorer\PLUGINS","C:\deploy\AdobeFiveO_1.PWC"


Program File Conflicts
----------------------
C:\deploy\AdobeFiveO_1.PWC
 No conflicts found


Registry Setting Conflicts
--------------------------
C:\deploy\AdobeFiveO_1.PWC
 No conflicts found
```

Listing 6.1: An example conflict report file that you should include in documentation for later troubleshooting.

In addition to the conflict report file, having a detailed listing of what the application does to the workstation is invaluable further down the road. Again, a commercial application distribution system can provide this data for you. Listing 6.2 shows an edited file from Prism Deploy. (I edited the sample to save space, but it shows the different sections of information from the original file.)

You can see the type of file and registry modifications captured in the package. By including the data now when the package is being created, you have a viewable file to reference if you are caught in a troubleshooting session. If you had to recreate the file later, it would take time from your troubleshooting and delay the resolution of the issue.

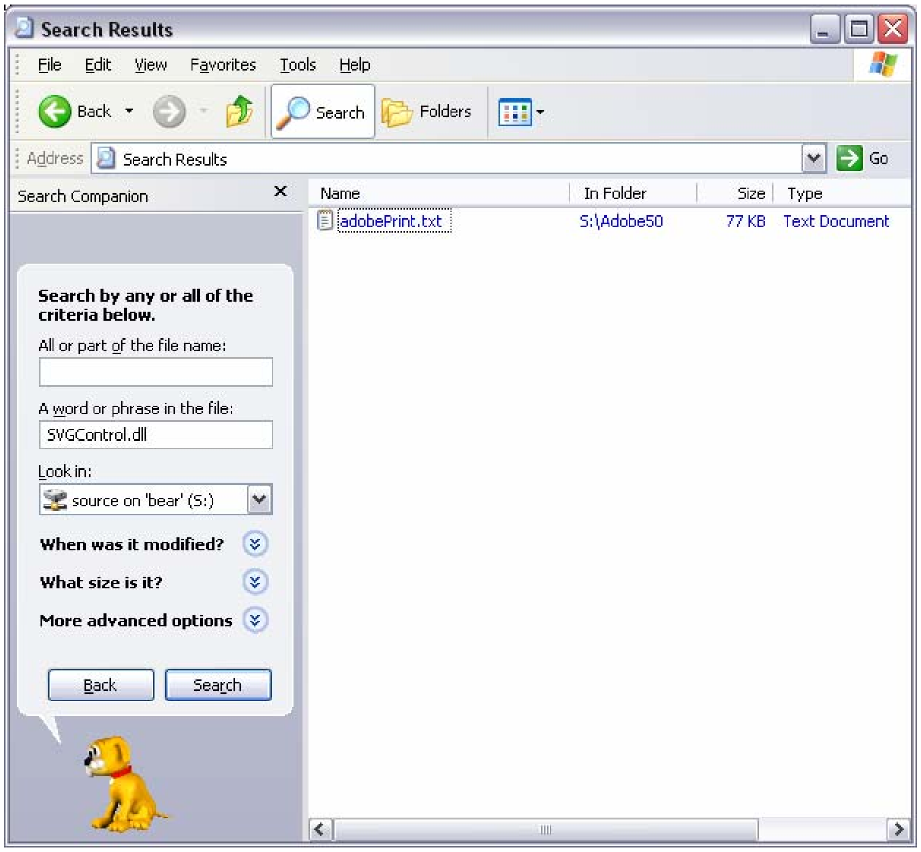
```
                                   CONTENTS

                              C:\deploy\AdobeFiveO_1.PWC

  C:
   Documents and Settings
    Admin
     Application Data
      Adobe
       Acrobat
        Whapi
         CreatePDFWinColor.ico           2KB  Icon
04/16/2001
         CreatePDFWinGray.ico            2KB  Icon
04/16/2001
         SearchPDFWinColor.ico           2KB  Icon
04/16/2001
         SearchPDFWinGray.ico            2KB  Icon
04/16/2001
         WHAppList.xml                  16KB  XML Document
08/17/2001
    All Users
     Desktop
      Acrobat Reader 5.0.lnk             1KB  Shortcut
03/23/2002
      |
      |
      |
      |


   Program Files
    Adobe
     Acrobat 5.0
      Help
       ENU
        ACROBAT.PDF                     28KB  Adobe Acroba...
09/10/2001
        DocBox.pdf                      82KB  Adobe Acroba...
08/01/2001
        MiniReader.pdf                  76KB  Adobe Acroba...
09/10/2001
      Reader
       AceLite.dll                     388KB  Application ...
04/16/2001
       ACROFX32.DLL                     52KB  Application ...
04/16/2001



      |
      |
      |
      |
  HKEY_LOCAL_MACHINE
   SOFTWARE
    Adobe
     Acrobat Reader
      5.0
       AdobeViewer
```

```
         EULA = 01 00 00 00
         TrustedMode = 00 00 00 00
        InstallPath
         (Default) = "C:\Program Files\Adobe\Acrobat 5.0\Reader"
        Language
         current
          (Default) = "acrord32.exe"
         next
          (Default) = "acrord32.exe"
      Adobe SVG Viewer
       2.0
        dir = "%WINSYSDIR%\Adobe\SVG Viewer\"
        path = "%WINSYSDIR%\Adobe\SVG Viewer\SVGControl.dll"
```

Listing 6.2: A sample of a detailed listing of what the application does to the workstation.

I keep stressing to have a copy of the data in a text-readable format in a central location for a reason. Speed to resolution in problem solving. For instance, several months and many applications after deploying Adobe Reader 5.0, you discover a custom application is having a problem with a DLL called SVGControl.dll. All you see in the pop-up screen is the DLL name, but no path or reference to where the DLL is being created. As Figure 6.2 shows, on your Windows XP system, you pull up the Search wizard, tell it to search the application source location and look for any entries that contain the string SVGControl.dll. The search pulls up the detailed file listing for Adobe Reader 5.0. You are now well along the path to resolving your problem.

A strong suggestion: If possible, keep the document readable via Notepad or WordPad. Complex documents needing Word look nice but are difficult to read on systems that don't have Word on them. You would need to move to another machine to read the document, a machine that may not be resident to your work location. You need to balance the level of detail with the ease of reading.



Name	In Folder	Size	Type
adobePrint.txt	S:\Adobe50	77 KB	Text Document

Search by any or all of the criteria below.

All or part of the file name:

A word or phrase in the file: SVGControl.dll

Look in: source on 'bear' (S:)

When was it modified?

What size is it?

More advanced options

Figure 6.2: The results of your search for troubleshooting information.

Distribution Documentation

After the package documentation is complete, you need to detail how the distribution is going to be done. This step is the form filling that helps get the package sent through the proper channels. You've already done the work, so you simply need to compile it into clear and short reports.

As Figure 6.3 shows, I've included a sample distribution document form for you to review. As you can see, the form takes up only one page, includes common answers that only have to be checked if needed, and provides basic data to get started. The document shows the form completed for a deployment of Adobe Reader 5.0. All this data is saved off to the application source server in the application directory. Of course, the source server is backed up so that all the data can be recovered when the server fails. (We all know that hardware fails, it is never a matter of if, only when.)

Filling out the simple form takes only a few minutes, and it provides the basic points from which to start a deployment. As new versions of the application are rolled out, updates are done to the document. You can create entirely new sheets, create a copy and update the new data, or scratch over the old data. Personally, I like creating a copy of the previous version, updating the changed information, versioning the document, and keeping all the versions in the same place. This way you can track changes by version.

Distribution Sheet

Item	Value	Notes
Name of Deployment	Adobe Reader	
Version	5.0	
Initial Deployment Date	January 19, 2002	
Base Package:	Commercial Vendor	Web download for source; free Reader version
Source location:	\\bear\source\adobe50	
Programmer (if custom):	N/A	
Manuals / documents	\\bear\source\adobe50\docs	
Dependencies		
Applications	N/A	
%PATH%	N/A	
Specific Server	No special server needed	
Variables	%USER%, %DPRTMNT%	
OS	Win2K, Windows XP	
Corp. Base OS Image	Version 1.342 or later	
Reboot Needed?	No	
Deployment Method:	Standard	Check document for Problem Definition Form application issues.
Target Group(s)	All	Check with Project Manager for target list.
Deployment restrictions	Must be done after normal business hours.	Normal considerations; no special issues
Rebuild Instructions	Trigger standard repair first. If not successful, remove/install	

Figure 6.3: A sample distribution document.

So far we've created the following documentation for the application deployment:

- The overview document
- Conflict listing
- File / registry data
- Distribution sheet

Will these four documents solve any and all issues encountered with the application? Obviously, they won't. Will the documents enable a deployment-knowledgeable person to come along at a later time and deploy the application properly? They should—assuming the person is familiar with the deployment methods in use within the company. The documentation is not a primer on how to perform the basics of the job. If problems arise in the future, can the documents provide help in solving them? They should be able to provide a starting point for file or registry references along with listing any known issues.

If you are part of an organization that has been doing deployments for some time, your documentation requirements might be more extensive to fit the unique situations within your enterprise. If you are starting off, use the provided samples as a starting point and adjust as your needs grow. Now, let's move into a big part of the overall enterprise strategy—coordinating the multitude of changes.

Change Management

Any enterprise, no matter what the size, needs a form of change management. It is critical for the integrity of the applications and the environment. If possible, the change-management people should not be the same people that do the distributions. Otherwise it becomes a case of the fox guarding the chicken coop, and the entire process becomes a joke. Change management has a simple directive: Review all production changes that impact the production environment and ensure the changes don't cause outages.

That large umbrella directive is a daunting task. The larger the enterprise, the harder the task becomes. Let's start with a mom-and-pop shop. One person does it all. Because only one person is doing all the work, that person generally doesn't try to deploy applications while converting the entire shop from Token Ring to Ethernet. That one person understands that the Web server is not really down, it's just being upgraded with new hard drives and memory. Anything that happens in the small shop is generally known throughout the organization.

Now look at the large enterprise that employs 500 staff members in the IT shop. You sent out a deployment last night and are getting reports of problems this morning. Not from everyone, just from some end points. You start digging and after an hour (or more), you find only the clients that were homed to a specific server failed to get the dark window deployment. A little more digging, calling around, and casually talking to the other members of the department, and you find the server was down for maintenance last night during the dark window. Now you have to reschedule the deployment and perform the normal reviews again in the morning. Let's see: an hour for basic troubleshooting, 30 minutes to confirm the server is functional for your needs, 10 minutes for the reschedule job, another hour to recover the end clients that got only part of the deployment before the server went down, and 30 minutes to review the rescheduled deployment. Three hours and 10 minutes wasted. And don't forget the reporting time to management on the reason for the failure. Four hours gone from your life. Change management is a process (remember, PP&D) that works to prevent those wasted 4 hours. By coordinating with all areas that impact the production world, needless outages are eliminated. Ideally, I'll admit.

Because change management is a process it does slow things down. Having to go through another procedure and another person prevents the casual production changes that get tossed in on a whim. Because of the process, it also tends to stop some installs. A simple tweak may not get installed because someone decides the benefit of the tweak is not worth the effort of going through the process. This result could be a good thing.

One benefit of change management that is overlooked is that it helps prevent duplication of effort. If a change goes through the process, someone may realize it is identical to another project and tie them together. Otherwise, both projects could proceed and end up stepping on each other.

Effective change-management enforcement will pay for itself quickly in terms of preventive costs. If you figure your current environment experiences three outages a week totaling 3 hours a week, you need to figure all the costs associated with finding resolutions:

- The number of full-time employees required for troubleshooting an issue.
- The number of full-time employees required to provide the fix (or fixes).
- On-call phone support of your staff (plus any off-hour call-ins).
- Lost income (real or potential) of the impacted application or system.
- Costs of vendor support, including vendor support calls; vendor call outs; and software creation, testing, and deployment.
- If the application or system is customer interfacing, figure in the cost of the lost goodwill from the customer. If the application directly impacts the customers, they will start becoming wary of using your systems if they can't be sure your systems will remain functional.

Do a spreadsheet on your own for the last outage you went through. Add up everything involved and try to associate a cost with it. Be sure to include the pizza and soda for the late night emergency work you did. If you haven't totaled up an outage before, you might be surprised how much it actually costs. By controlling the change process, enforcing effective peer reviews, ensuring all the back out or rollback plans are in place and workable, change management can reduce or eliminate preventable outages. That is the goal.

One comment about the peer review process. Make it valid. If your shop has only one or two IT people, the review is simple. "Hey Tim, check out the package and tell me if I missed something, would you?" is all it would take to get a package reviewed. If your shop is large, the review process may involve a separate department that could require an entire process itself (which might add days or weeks to the scheduling date). Remember I said that size does not matter? In this case, the practical application of the peer review is impacted by size, but the concept is not. The idea is that someone with acknowledged skills will review your entire application. This review is not a certification step in which the application is tested against all images and applications, but a review of the package. The peer review checks for

- Syntax errors
- Logic flow
- Completeness of the documentation
- Effectiveness of the package (is the delivery method going to work?)

- Compliance with company (and/or legal) policies and standards

Even having a buddy check your work is a good thing to run through. We so often get swallowed up in one logic flow that we miss alternatives and overlook the simple mistakes. The situation is analogous to spell check. When spell check runs against a document, does it go *threw* it *two* check for proper word use? The previous sentence made it through the spell check, but is obviously using the wrong spelling. Having another set of eyes will catch a lot of those simple mistakes.

☞ If you really want to make sure the entire package is correct, send it to someone that you don't really get along with. There is nothing like having a known antagonist review your work to make sure it is correct!

A danger of the change-management process is having too much interference. Keep on your guard. When the process becomes a hindrance, a review of the process is required. Make sure all the steps that have been added bring a value to the process and aren't just included to make someone to feel important.

How does a change-management process work? In keeping with the simple directive, anything that will impact the production environment needs to be reviewed prior to implementation. That means changes and updates are scheduled long before they need to be installed. This time gives everyone a chance to review the change request. It also forces people to plan ahead. OK, OK, OK, I know. Forcing changes to be planned in advance is Nirvana. A strong change-management process will balance the needs of a true emergency with the need of a Project Manager to meet a looming deadline. As the change-management process gets better, the need for true emergency updates will decline.

To staff a change-management process, you need to have at least two technical (underline that part) people in the shop. One peer reviews the other. Thus, the person creating the install does not do the review and schedule the change. In a two person shop, trading off the change-management review hat would be expected. Larger shops would have a dedicated person in the role. The change-management process reviews all documentation, thus forcing it to be completed on time. And the process ensures the documentation is useable, not just slapped together.

Change-Management Request Forms

Let's go over the change-management process in a general manner. Prior to applying any changes to the production environment, the change is submitted to the process in the form of a request. The change request consists of

- Any documentation (as discussed earlier).
- A description of the change.
- The scope of the change (how many clients, servers, users, and so on).
- Who exactly will be impacted? Which business units?
- Who is doing the change? In a small shop, that will normally be the same person each time.
- What steps are needed to implement the change?

- How is a rollback or back out performed? Is there any backup that needs to be done first?
- What are the dark windows? Is the change going to be within the accepted dark window or is it being done at some other time?
- How is a rollback decided on? Does the installer make the decision? Is there a drop-dead time needed to ensure there is enough time to perform the rollback?
- What steps are needed to verify the change and who does the change?
- Are any signatures needed for the change?

Figure 6.4 shows an example form that you can use as a starting point, if you don't already have a change request form template.

```
                      Change Request Form

1) Application name
      _____________________________________________________
2) Name of submitter
      _____________________________________________________
3) Short Description of the change:
_________________________________________________________________________
_________________________________________________________________________
4) Areas of impact:
_________________________________________________________________________

5) Date of install            ___/___/___ 6) Time of install ___:____
AM / PM
7) Duration of install  ___:____          8) Back out deadline ___:___
AM / PM
9) Installer: (name or department)
____________________________________________
10) Within dark windows? Yes ___   No ___
      10a) If No, has the impacted area been notified? Yes ___   No ____

11) Critical category:
Emergency ___     Important ___     Routine ___ Informational ___

12) Peer review:
      Scheduled: Yes / No      To Be Done By:
________________________________
      Date  ___/___/___

13) Documentation Completed? Yes ___  No ____
      Application ___   Network  ___           Back out ___
      Support ___

14) Install actions to be performed. Attach a separate sheet or the
application documentation.
15) Detail the back out actions. Attach a separate sheet or the
application documentation.
16) Detail the verification actions. Attach a separate sheet or the
application documentation.

Approvals:
Change Management
______________________________________________________
```

```
I.T.
     ____________________________________________________________
```

Figure 6.4: An example change request form template.

The Change Management Coordinator

To properly implement a change-management policy, there must be a Change Management Coordinator. Don't rotate the duties; keep one person as the Change Management Coordinator. This provides the consistency needed to make the policy work. If you must rotate the duties, keep the tour of duty 6 months or so.

The main function of the Change Management Coordinator is to do a cursory review of all the requests and ensure everything has been addressed. The Change Management Coordinator is a process coordinator with a good understanding of technology. The job's main duties include:

- Reviewing all the required documentation for completeness.
- Keeping the change calendar accurate and available for everyone.
- Ensuring that the proper people review a request for conflicts or problems.
- Distributing the requests prior to the formal scheduling meetings.
- Calling emergency reviews as needed.

A Change Management Coordinator must understand the technology enough to know whether the data they collect is accurate and to get the right people involved as needed. Could the Change Management Coordinator be an administrative assistant? Sure. Many of the duties involved compliment each other nicely. I've seen this double duty work in many shops; it all depends on the person doing the functions.

Flexibility is important to a good Change Management Coordinator. Not all change requests warrant a full process. Other changes may require added information. That is where the technical knowledge of the Change Management Coordinator comes into play.

If you are rolling out a registry change that adds text to the Winlogon\LegalNoticeText, the process is pretty simple. The change is not a major change to the production environment. A peer review would ensure the script was tested and does not accidentally apply the change to a different registry key. The back out is easy, so the overall issue is pretty simple. The Change Management Coordinator would confirm the basic documentation was in place, the peer review was done, and that the method of delivery is a proven method. Time of distribution is not a problem.

Now take the same change request and apply it to a service pack upgrade. In addition to the service pack install, you are going to apply added registry tweaks and modifications to workstations and servers. Now you have a complicated process that will take more care in reviewing and ensuring all the testing is done for the install and roll back. The Change Management Coordinator is going to carefully review the request and ensure the business part of the enterprise is aware of the request and expected install date. A little more work will be applied to the request.

Obviously, the two examples are on either side of the extreme scale. It is important to realize the change-management process needs that type of flexibility. If a rigid system was in place that followed a detailed process from step 1 to step 700, making simple changes would never happen. The balancing act is being able to adjust as needed without going too far to either side.

A good Change Management Coordinator will review all the steps, follow the procedures, and know when to call a halt to a process or just delay it. If a critical deployment is needed and the only thing amiss in the procedures is a missed field on a document, the deployment would not be held up. However, if the review process shows that problems are evident, the Change Management Coordinator should put a stop to the change until all the issues are addressed.

> The Change Management Coordinator does not need to be the sole decision maker. A committee can be used and usually is. The members of the committee can rotate depending on the technical knowledge needed of the request. In these cases, the Change Management Coordinator is acting as the chairperson of the committee, not the sole decision maker.

An effective change-management process becomes the de facto enforcement arm of the IT shop. Everyone will know what kind of fallout will happen by not following the process and that will prevent many last-minute adjustments or rollouts. And by forcing everyone to schedule the changes, those implementations that are not planned out will be reduced or eliminated. The key to success is having the proper management buy-in of the process. When someone circumvents the process, some form of retribution needs to be uniformly applied. And if the process circumvention causes an outage, the enforcement should be swift and mildly public—a town square flogging is probably a bad idea, but making it public knowledge that "someone on the project team" is in trouble might a good idea. Otherwise the change-management process is a waste of time for everyone. Either everyone follows the process or the process should be removed.

If a change is a true emergency, the change-management process should have a clearly defined method of streamlining the process to facilitate a time-critical change. Review that emergency process often to keep it accurate and not too burdensome or easy.

An Example Change-Management Process

It's time for an example of the process. Let's take the Microsoft security rollup package for Win2K Pro. You need to get the application out to a static group of users via your commercial application distribution system. The package has been adjusted and tested, and you have done all the due diligence required to ensure the security rollup package will deploy properly without issue. Your defined change-management process requires that you

1. Submit a request to the change-management group that includes an overview of the application. The overview is high level only and will be used for general reference during the change-management process.
2. Schedule the request one week in advance.
3. Attend a scheduling meeting to discuss the changes. The frequency of the scheduling meeting is dependent on the number of changes the enterprise experiences. Weekly is common for most shops. Larger shops in a dynamic environment often have daily meetings.

4. The Change Management Coordinator will review the required documentation before the scheduling meeting and distribute the information accordingly. Doing so gives the attendees time to review the documents and process before the meeting.
5. Get signatures from specific groups or people before the meeting. If another department will be doing the distribution, you need to have their buy-in that the proposed date and process is OK for their schedule. Because the security rollup package will require a reboot, you need to confirm that the timeframe is within an agreed upon dark window and that there are no changes to the window during your proposed time. The scheduling meeting also prevents conflicts between groups; your goal for your application is to ensure that the targeted machines are available. Scheduling a reboot on March 14th at 4:30 for the accounting department most likely will not win many friends for you (tax crunch time for the number crunchers).

You've submitted your proposal, have checked with major parties involved, and have your documentation in line. You've identified the rollback options and have had a peer review done. You've done the change proposal request and have all your ducks in a row. The date for the scheduling meeting arrives. Now it is time to sweat a little.

A good scheduling meeting should make you nervous. You know you will be facing a tough audience that will be checking whether you're prepared and you've done everything. You will also be placed in to a negotiation stance if there are conflicts with other changes.

Another strong payback to the change-management process is that you can schedule changes to prevent conflicts. If you are sending out changes and the network suddenly goes down, wouldn't it have been nice to know the routers were going to be undergoing firmware updates before you got part of your deployment rolling? The scheduling meeting will not only review the applications for completeness, but will also be looking for conflicts and overlaps in other changes. For that reason, it is critical to have a single scheduling meeting instead of offshoots for each discipline. One meeting for NT, one for UNIX, one for network, or one for mainframe most likely will not work as overlaps will occur.

The scheduling meeting begins and you are in the hot seat. The Change Management Coordinator brings up your request (gives a little evil laugh) and begins the review process. The group reviews the basic request, confirms that all the i's are dotted and t's are crossed, and confirms the request conforms to the standards and policies. Not technically, just in the procedural arena first. Because you are an efficient professional, nothing is amiss. Now the Change Management Coordinator begins the technical review. Because everyone in the meeting has reviewed the request before the meeting (we can only wish for Nirvana), you give only an overview of the request. You identify the groups that will receive the security rollup package, explain that it will be sent out Friday after business hours, note that it requires a reboot which might impact anyone working late, and tell the group that you will be in on Sunday afternoon to review the deployment for any issues that might cause problems Monday morning. The Change Management Coordinator opens the floor for general comments and the questions come in. After fielding a barrage of questions, the group admits that you have thought of all the possible problems and how to resolve them. Your request is approved for scheduling.

The Change Management Coordinator then checks the schedule and finds that the building you are going to be sending the application to will be experiencing a power outage on Sunday for normal maintenance. Your request is adjusted a little in that you will have to confirm the package deployment on Saturday afternoon or evening instead of Sunday afternoon. Although this change shortens the time of the deployment by a day, it also keeps you from wasting a trip into the office on Sunday. And your part of the meeting only lasted 10 minutes. In summary, the change-management process

- Acts as the enforcement arm for change control.
- Prevents conflicts in scheduling.
- Tracks changes.
- Forces compliance with company and legal standards and policies.
- Ensures documentation is complete.
- Ensures back out processes are defined.
- Keeps the flaky installs to a minimum.

Where is the Source?

Here is a difference between deployments in large and small enterprises: Where do you keep the source files for an application and how do you get it distributed? When I ask systems administrators this question, they quickly reply "Oh, it is stored on server X. Everyone knows that". Excellent. The next question: Is that source SECURED? The usual response is "Huh?"

Here is where the size difference comes in to play. Depending on the commercial application distribution software you use, you will deal with three general types of source locations for software deployment:

1. Application source—This source is a CD-ROM or floppy for shrink-wrap packages or a source location for in-house created software. This source is generally the unedited and unpackaged source data that is has not been modified for your distributions.
2. Deployment source—This source contains the edited and deployment-ready packages. It is sometimes called the Golden server.
3. Distribution endpoints—Your commercial application distribution software generally likes to have a place to find the software-modified source or your custom deployment solution has locations from which the files can be installed.

Dealing with the application source is easy. The files are generally read-only on the CD-ROM (or floppy or downloaded file). Smaller shops generally have a desk drawer in a systems administrator's desk designated as the place to put all the CD-ROMs. The larger the shop, the more sophisticated the storage location becomes. It moves from a desk drawer to a file cabinet. Not much difference, just a larger storage location. As the shop gets larger, the source files move to a server location for easier access between buildings. The source files are still resident on the CD-ROM, so if the server location gets wiped out or the files are modified, the original source files can be replaced. No problem so far.

Now we deal with source location number two—deployment-ready source. You have worked hard to create the package and have it ready to go. No matter which distribution method you use, that deployment-ready package needs to reside someplace prior to being inserted into the commercial application distribution software or your custom method. And the prepared package is usually the only copy around that includes all the final tweaks and modifications.

If anything were to happen to the package, your life would become very difficult. Worse, if something happens to only one file, you could spend days figuring out what happened. Not just a file deletion or version update that breaks things, but maybe a registry INI file is modified and a value is changed by one digit, and the change only happens after 75 percent of the deployment is done. Now you have a troubleshooting nightmare. The solution is easy. Ensure the package-prepared source files are read-only and limited as to who can modify them. If that means you burn a CD-ROM of each application and store it with your other CD-ROM source files or you lock down the central location server, the end result should be to protect those files from any modifications after the package is ready to go. If you rely on a server lockdown, check your backup strategy on a regular basis and confirm that the data is valid on the backup media. A couple of minutes each week to confirm that the backups are working will save you from trying to rebuild a package that was created last year.

I recommend a combination of the CD-ROM and server methods. After the package is setup and ready to distribute, I like to burn a CD-ROM copy of the application and store it in the central corporate store. If it is a package that I've created or worked on, another CD-ROM is created and stored in my desk drawer for my own archives. In large companies in which duties change and re-organizations occur on occasion, the source CD-ROM location can sometimes get lost in the shuffle. Plus, I'm paranoid and believe you can't have too many backups.

For the source server, you want to do two things:

1. Ensure everyone that needs access can get to the data when they need it.
2. Prevent unauthorized users from changing the data, even if by accident.

To accomplish these two goals, the source server needs some minor adjustments. The source server does not need to be a dedicated box (unless your volume is that large and warrants the cost). Because it only needs to provide file storage, nothing fancy is needed in the processor department. Sharing the load with other functions is a good cost-saving mode, which will endear you to the accounting folks.

Read-Only User ID

For the read-only user ID, create a userid that is used solely for read access to the source location directory. Set the password to something easy and don't change it. Ensure that the user ID is a simple one, either a domain user or machine resident, depending on the role of the server. For example, a user ID called SourceRead or AppView is generic enough to be remembered easily by the technical staff. You can set the password to be the same as the user ID to make it even easier to remember them both. I've seen implementations that have created department- or enterprise-level read-only IDs. That allows further segmentation of the source directory and who gets into it.

> Why a user ID with a simple password or the same password as the user ID? It is easy to remember, won't be locked out as much, and is for general use by all parties. It keeps the unwanted out by virtue of the user ID and password, but still provides read-only access. You only keep out 50 percent of the computer population.

Set the directory permissions to read-only for that user ID. Make sure you leave an administrator user ID with full access (more about this requirement in a couple of paragraphs). Remove the Everyone group. Yes, you could also set the Everyone group to read-only. By requiring a separate user ID to access the data, you prevent the temporary mail clerk from downloading all your software when he accidentally stumbles across the source location. Using a specific group to do the same thing might work, assuming you restrict the user IDs in the group. You can use this method to track which departments or groups are using the directories.

Create a share name for the read-only user ID to connect to. I suggest that this is the exception to normal share creations. Limit the access to the read-only user ID—you want to make the access available to those that know of it, not to the entire company. (Unless that is your goal, then leave it wide open). Make sure your technical staff is aware of the read-only ID and password.

Of course, make sure your security staff is aware of the need for and the use of the read-only user ID. They should be aware of the process and the simplification of the password, as it might be contrary to the security policy for the company.

You need to ensure that other groups or user IDs have access to the directory. SYSTEM needs full access for backups and other basic OS needs. Be sure to remove Domain Admin or Administrators from the directory access. Why? You want to control who has change access to the directory. Create a separate group with full access. Call it CM or Release Mgmt or whatever you like. Place only specific users into that full access group and limit the number of those users. If the group is opened too wide, you defeat the purpose of edit control.

You have now accomplished both goals. Anyone that needs access to the source application files can get it with the read-only user ID while only a small group is available to configure changes to the directory and contents. You are one step closer to a Golden server concept in which the files resident on the server are a known quality and don't get accidentally modified (or deleted).

Be sure to turn on auditing for the server and for the directory and subfolders. If there is any discrepancy with the source content, you can use the Event Viewer to determine who was updating that particular location. It also helps to keep folks honest if they know they are being audited, so be sure to publish that piece of information.

Along the same lines of the packaged source location and including change-management functions, you may want to add a few procedures to protect this location. It depends on your environment. The larger it is and the larger the access to the source files, the more you need to track who is doing what. Some sort of paper trail is a good thing. Possibly include in the scheduling meeting when the files are to be moved to the application source server from the certification group. Nothing should be on the source location without being approved by the certification group. By having the responsible people agree when a package is ready to go, you add greater validity to the Golden server concept. The entire enterprise then knows the files located in the source server are Golden and are the source files for distribution.

If you place the applications from the certification group onto the Golden server immediately after certification, there is a chance that the application may be on the source too early. The application may not be scheduled for deployment until the following month and could be an update to an existing deployment. See the problem? The new data could be overwriting the old too soon or there may be dual applications on the Golden server. Which one does the field systems administrator use? The concept of the Golden server is to be the source location for all *production* applications. Therefore, don't move the files too soon.

Small or large, your enterprise should have the first two source setups: The original files and the Golden server distribution files. If your user base is widely dispersed (across the country or across the ocean), a geographically close copy of the source server would benefit everyone. It would reduce the network usage and provide faster local access to large files. Trying to download a ghosted image from Denver for every workstation you are installing in London would take a long time. Downloading the same image to London one time when the network usage is low (at night or on the weekend), then having the image creations calling the local server in London suddenly has increased the image speed, reduced the network bandwidth use, and reduced network conflicts with other business needs. Evaluate your connections, file sizes, and use to determine the best fit.

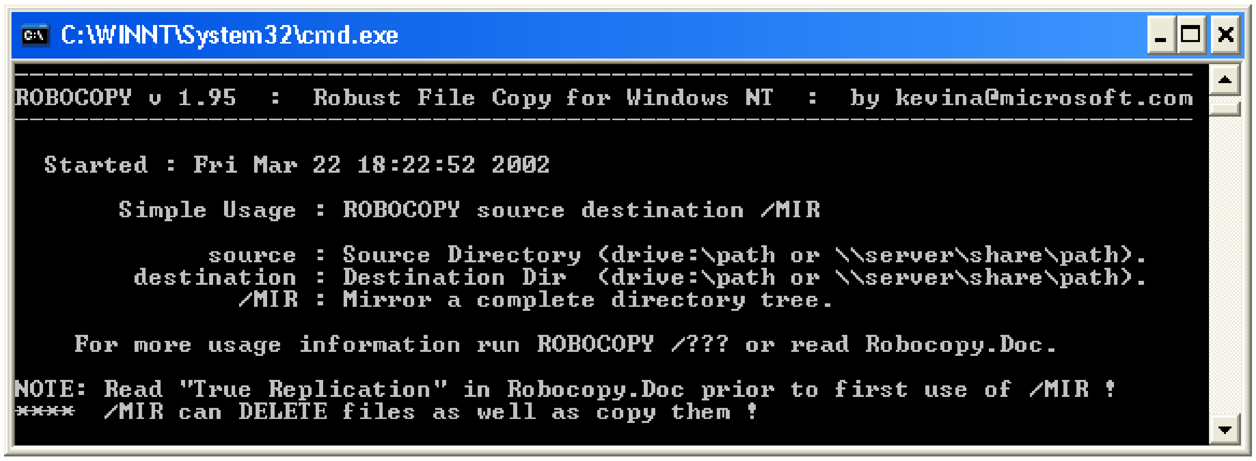
Mirroring

One issue with the duplicate source server is mirroring. Each source server must stay identical to the master Golden server at all times. Without fail and without question, the duplicate must be verified against the Golden server master on a regular basis. Lock down the duplicate servers just as tightly as the Golden server. Possibly even tighter if you can, so the only way that new files can be placed on the duplicates is from the master.

There are several methods available to provide the mirroring needed for the slave servers. The commercial applications have some enhanced features that allow for real-time mirroring. Mirroring means that any changes (additions, deletions, modifications) to the source location are duplicated to the slave locations. This functionality requires some added capital investment. Your ROI evaluation will determine whether mirroring is worth the cost for your situation. If you perform large numbers of modifications to the Golden server, having such a product is worth it. If you only make one or two (or less) modifications a week, the added cost may not be worth it.

One commercial replication solution is provided by Legato Systems. For information about RepliStor, check out http://portal2.legato.com/products/replistor/.

As Figure 6.5 shows, you can use the NT resource kit ROBOCOPY utility for replication. Use a release version of 1.95 or later, as it supports the /MIR (mirror) and /SEC (security) options. By setting up an AT command on the Golden master, you can have the master server replicate to the slaves on a fixed time schedule. Be aware that the /MIR option will delete files as well as copy, so make sure you have the source and destination parameters set up properly. It would be disastrous to mix up the two servers and have the Golden server overwritten by the duplicate!



```
-------------------------------------------------------------------------------
ROBOCOPY v 1.95   :   Robust File Copy for Windows NT   :   by kevina@microsoft.com
-------------------------------------------------------------------------------

  Started : Fri Mar 22 18:22:52 2002

       Simple Usage : ROBOCOPY source destination /MIR

             source : Source Directory (drive:\path or \\server\share\path).
        destination : Destination Dir  (drive:\path or \\server\share\path).
               /MIR : Mirror a complete directory tree.

    For more usage information run ROBOCOPY /??? or read Robocopy.Doc.

NOTE: Read "True Replication" in Robocopy.Doc prior to first use of /MIR !
****  /MIR can DELETE files as well as copy them !
```

Figure 6.5: You can use ROBOCOPY for replication.

Whichever method you use, be sure to copy the security settings on the files and directories so that you don't have problems. Otherwise, plan on adjusting the security settings after the copy process is done each time.

When replicating to slave servers, whether via a commercial product or ROBOCOPY or other means, don't rely on only replicating just the changes. Do a full replication of the master server to the slaves on a regular basis. That way, the consistency of the files is always confirmed. In a Golden server and slave server environment that experiences heavy source modifications, verify the consistency once a week.

Verification

If the commercial application distribution software you use can reliably replicate data without problems, verification is not needed. In fact, it may be impossible to force a replication or to update the data outside of the commercial application distribution software's method. You will need to carefully review the process and watch it, especially when you first install the software.

Verification is an issue. How to do it quickly is always a concern. WINDIFF, that infamous file-verification utility is a good way to check things, but its speed is not the best for large directories or over long distances. Check the Web for freeware utilities. ROBOCOPY works, as it only replicates files that it believes are different, though it tends to only check the filename, size, and date. Servers in different time zones can cause false readings. If you use the /XO (eXclude Older) switch, it will prevent some time zone issues. Test it for your situation.

The commercial application distribution software you use may have slave servers that it uses to speed up the distribution to the clients. Often the structure of the software's servers is not directly editable by anyone, so the contents are safe from accidental modification. If the commercial application distribution software places the files in standard directories, you always run the danger of having a file or setting modified by accident. Check into the software's documentation for information about having the files verified on the distribution endpoints to keep those accidents from happening.

Be careful about locking down a commercial application distribution software distribution point. Review your documentation carefully and check what access rights the software is looking for. If you lock down the distribution point to only your read-only user ID and a few select user IDs, the software (or GPO, if you use it) may not have access to directories. Make sure you don't lock the systems out. If your software relies on service user IDs, keep those IDs functional in the directories. Keeping SYSTEM at full access generally keeps the OS considerations functional. Test it before you roll it out.

If you don't use commercial application distribution software for deployment but still opt to have remote distribution points, you can use the same Golden server that I discussed earlier. Keep the access restricted and things should be fine.

The main points about the source location are:

- Keep the source files locked from unauthorized modifications. Accidents happen. Don't assume the Domain Administrators will never adjust the files. Assume a mistake will happen and prepare for it.
- Keep the original source separate from the packaged deployment source.
- For replicated servers, verify the files and replicate changes on a regular basis.

- Track who makes changes to the files and keep that access to a very small group.
- Keep the Golden and slave (duplicate) servers consistent at all times. Don't let their reputation ever become tainted or it will be difficult to trust the contents in the future.

Hardware Thoughts

In previous chapters, I brought up hardware: I discussed the need to know your targets, ensure that you test against a good representation of what you have in the target list, and have a good understanding of the hardware drivers. The importance of this topic warrants a short recap of the subject in this chapter.

The question often pops up whether it is better to have a mass-marketed computer or to create a scratch-built unit. The discussion usually centers on the initial cost of the unit. The common argument states it is cheaper to go to the local PC store and put together a system out of parts on sale this week than spend the extra money to purchase a mass-marketed system. The technical folks usually argue that they can build a better system cheaper with more features than can be purchased in a pre-made configuration. To those statements, I always agree. By carefully shopping around on a regular basis, you can save a few hundred dollars in piecing a system together.

However, the issue is consistency. For the mom-and-pop shops that have maybe a dozen machines, piecing together a system each time you need one and saving money is the prime concern. Because in this type of environment most distributions are probably done via Sneakernet, saving the money is more important.

For any organization that uses an automatic deployment method and has several machines to deploy against, knowing what you have out there is more important than the initial cost savings. Having a consistent and known hardware environment is important as it saves the back-end costs of troubleshooting and having to adjust packages for one or two machines.

Suppose that your environment has 100 machines that were all built from spare parts. You don't really know how many different video cards are out there, which driver versions there are, or even how many different manufactures of the cards exist. For example, if you roll out a service pack update, you can't test the video driver effectively.

Now take that same 100 machines and figure that they are all from a major maker (insert your favorite hardware company). If it took 2 years to build up the 100 machines, you have a good understanding of what is out there. Major makers don't bounce around much in the major components. You will see different versions of video cards but you can be pretty certain that the cards are from the same maker, thus the number of variances are smaller. And most companies work to keep the drivers backward compatible within a generation or two. Instead of having to locate the video driver for each card variance on the video card Web site, the PC maker may incorporate the variations in a driver download.

A side note about vendor hardware: watch the model lines. Buying nothing but Dell or Compaq or IBM machines does not mean the systems will be consistent between the various lines or within the model line. Different model lines within the vendor may have different end-of-life schedules for components. As a general rule, higher-end models aimed at the business world tend to stay consistent and compatible longer than the entry-level lines aimed at the consumer market. Do your research.

The point is that you can save money upfront to the tune of a couple hundred dollars (maybe) per unit and pay higher back-end maintenance costs in the form of distribution rollout troubleshooting. Or you can go for the longer term savings of lower maintenance costs in the rollouts with a slightly higher up-front cost.

The same goes for the servers. Don't build Frankenstein-type models and expect them to be the same from region to region or offer identical functionality. Keeping a known environment will solve many problems and you can adjust the packages and distribution methods accordingly beforehand because you know what issues to expect. Consistency is the mantra to follow.

Repeatable Processes

Big or small, an enterprise gets the best cost savings (money and labor) from deployment methods and procedures when the entire process is repeatable. The methods talked about in the past few chapters describe how to set up a distribution method and testing environment and how to get the applications rolled out. After you've done the first deployment, the subsequent methods are easier to use as the setup work is done.

The change-management methods discussed in this chapter—the process and forms—are time consuming for the initial setup. After they are developed, the same steps are followed each time. What may have taken a day to set up initially is now an executable step that takes only a few minutes.

When you are doing the design work, keep in mind that everything needs to be done again and again by others. Review the process with an eye toward a new person being able to read the documents and continue the process without problems.

Software Metering

Software vendors have an annoying tendency to nag you for payments. They like getting paid for all the copies of their software. Payment may be a flat fee, per user, per concurrent use, or by some other rather creative method. Every contract is different, and the same software license can even vary from company to company, depending on how sharp your purchasing department is. Compliance with the terms of the license relies on you. Paying for a one-workstation copy and sending it to 3000 workstations is not compliance. The software vendor and their lawyers would have issues with such a manner of deployment.

In recent years, software piracy has been raised from the classic stealing and selling of the software on the black market to the soft lifting of businesses not paying for all the versions they have. Unless the software license specifically allows it, a copy at the office is not permitted for home installation. Making a copy for archival purposes is allowed per the Copyright Act, Title 17, Section 117 of the US Code. If you aren't sure about the rules for your software, check the license very carefully.

Staying compliant with the software license is not just the venue of the budget department. With the publicity that the Software & Information Industry Association (SIIA) has been doing about reporting software piracy at work and other areas, doing so has become a revenge tactic for disgruntled employees. If the Software Publishers Association (SPA—a division of SIIA) comes into your shop, your company needs to show how compliance is being done with the license agreements. Suddenly, you are in a legal hot seat.

Check out http://www.spa.org/piracy/default.asp for details about SPA's report software piracy program.

You can use software metering to help avoid any conflicts. Software metering isn't just doing an inventory of what is on the workstations in your control, but actively controlling use in compliance with the various agreements. Life would be easier if all the license agreements were complimentary.

The big problem that faces you is wading through the myriad of agreement terms. For the licenses that require payment for each copy resident on a computer, all you need to do is gather your inventory and count the instances of residence. Most commercial application distribution software offers an inventory method, or you can do a simple check for critical files when a user signs on. However, if you are being charged by concurrent use or you can make a copy for the office and the home, the inventory solution won't work. Trying to figure if a copy is duplicated at a paired home and office system or how many copies are being used at the same time at any point can drive you crazy. How does your license agreement handle terminal server sessions? The software is on one system but is being accessed by multiple users. Does the one-copy per-machine clause work in this case or do you need a separate copy for each session? The issues keep getting bigger and bigger.

Software metering is one solution. It not only covers the inventory aspect but can also control who gets a copy of the program. Because your license may be based on active use, not just workstation residence, knowing which copies are actively being used is critical. Metering has been around for a long time. The idea is to monitor actual use, not just inventory residency. With most metering, a high-water limit is set and enforcement is either activated or set to a warning notice. For example, suppose application XYZ can have 3000 active users. When user 3001 tries to activate the program, the user is locked out from use until a license opens up or a warning is generated indicating the high-water level has been raised.

When you first install metering, it is a good idea to set it in a warning mode for awhile. You may have licenses for 3000 users but find active use is 4000 or even 2000. By setting the metering software to monitor and not lock out, you can find the true active use without upsetting users by suddenly and perhaps needlessly locking them out.

Monitoring software and not enforcing lockouts or license limitations is termed *passive metering.* Enforcing license limits is termed *active metering*.

When metering is first installed in a company, there will be problems and complaints when enforcement of limits is activated. Over advertise the need and reason for the software to ease the calls to your Help desk and managers.

Metering software requires an agent to be loaded on the client piece. If it is included in a suite such as SMS 2.0, the installation is easy. Otherwise, it is another service application to roll out. The client piece will contact a central controlling interface and relay the use on the local client. Even if applications are only resident on the workstation, the client metering piece contacts the central location for approval. Some metering software lets the application start while the confirmation is being negotiated. Should use exceed the limit, the metering client may cancel the application (annoying as it is). If the central location can't be reached, the application continues to run. This behavior allows offline use on laptops and the like.

☞ Each metering software vendor has its own methods for handling offline users. Check the vendor carefully when you are shopping for a product.

An important software-metering feature to be looked for is a VIP override switch. You don't want to prevent your CEO or CIO from opening Excel because of a software license limitation. Especially if it happens to be near the time for your raise. You do want them to get access but to have the use reported to you if it exceeds the license limit. Don't allow exceptions to the license limit. Instead monitor the excessive use and true up the license as needed.

The software-metering product should also report to you the number of users that are waiting for a free license to become available. To keep licenses available, a timeout parameter should be set so that a user who opens a copy of Excel on Monday morning shuts it down before Friday night. If the user actively uses the copy for the entire time, the user has a legitimate right to it. However, if the user simply uses the copy occasionally, a timeout parameter would catch that and remove the license. Check to see how your metering software handles inactive users.

Getting the money to purchase metering software can be hard. It's one of those essential items that does not bring a visible benefit to the organization. Instead, it constantly brings up the need for more licenses, is considered an invasion of privacy, and is a software cop that prevents people from freely using the software as they want. Try this selling point: It can save money by pointing out a reduction in licenses is needed: When you purchased the XYZ application, the price was based on concurrent use. Unless you knew exactly how many users would use it on a high-water mark, the license number was probably guessed at and guessed high. Installing a metering package and monitoring the active use, the high-water mark shows only 589 concurrent users out of 3000 licenses for 10,000 users. Add a little for growth, then return 2400 licenses. At $25 a copy, a savings of $60,000 is realized. Not a bad savings.

☞ For a partial list of metering software, check out http://www.spa.org/priacy/asset/default.asp and http://www.appdeploy.com.

Summary

Deploying software in the enterprise is a process and procedure that needs to be documented. As techies, PP&D are areas we tend to avoid as much as possible. However, to have a deployment system that is workable without our direct involvement in all stages, the steps need to be known by others. The best way to get that known and repeatable process is to clearly document the policies of the deployment use, the procedures to create and implement the applications, and the entire process. If PP&D is effectively implemented, vacations can actually be vacations and not a series of check-ins.

Copyright Statement