



realtimepublishers.com[™]

The Definitive Guide™ To

Windows Software Deployment



Making Complex
Technology **SIMPLE**
Since 1985

Chris Long

Chapter 5 Deploying to Remote Users.....	119
Generic Remote Issues.....	119
Types of Remote Users.....	121
Common Issues.....	122
Scheduling.....	122
Network Transmission.....	124
Deployment Status Reporting.....	125
Connectivity Durations.....	125
Security.....	129
User Impact.....	133
Remote Hardwired.....	134
Remote Stationary.....	135
Roaming Users and Machines.....	136
Rarely Connected.....	138
A Real World Example.....	140
Summary.....	141

Chapter 5 Deploying to Remote Users

The thought of getting a deployment out to a remote user can bring even a strong network administrator to the edge of a panic attack—especially for that first company-wide deployment when the project has a white hot spotlight of attention on it. However, this task does not need to be a difficult endeavor. Just like the slogan says No Fear—you’ve already got the information and as an administrator, you’ve got the attitude, all you need is the experience. For a remote administration, the principles that we’ve talked about in previous chapters still apply:

- A package is delivered to the target machine
- The install process is started

Physical distance is not a large issue. A target machine can be sitting next to you or half a world away, and the deployment method is basically the same. That is, unless you try to Sneakernet the install. I’m sure the company travel department and your finance administrator will have something to say about all the travel.

To clarify for the purpose of this chapter, the term *remote user* will be any user or workstation that is not within the immediate vicinity of the network administrator. The user in the next building is considered a remote user because the need to send applications to the user has the same hurdles as a user in a building 2000 miles away. Where the fun begins is dealing with machines, such as laptops, that move often or are not directly attached to the corporate LAN/WAN. Deployments to those systems are where this chapter is heading. As the laptop is on the extreme side of the remote user scale, I’ll use them as examples in this chapter. After you get the laptop deployment issues resolved, the rest of the machines will be a snap. Grab a box of your favorite chocolate chip cookies, a couple of caffeinated beverages, and read on.

Generic Remote Issues

Focusing on the laptop issue (remember they are used as the extreme example), there are a couple of generic issues you must account for:

- Line speed
- Connectivity duration
- Windows security (domains, ACLs, and the like)

When a laptop user is on the physical premises for a full day and connects to the hardwired LAN, the laptop is running at your normal LAN speed and can be considered a local connection. No problems at this point. However, laptops spell mobility to the user, and these systems can also connect to your network at other speeds. Connections can range from a phone line running on a 56Kbps (or slower) modem to a DSL line running at 256Kbps+. Often, laptop users will connect with various methods in a single day. Take my own laptop connections as an example. Depending on where I am, the connection styles will vary greatly. It is not uncommon for my connection day to cycle from a Gigabit Ethernet hardwire to a 56Kbps dial up in a single day. Direct connect, VPN by way of the Internet, password access, and token security access are methods added to the mix. If your deployment method uses fixed network timing as a means of triggering an event, you can see how identifying the proper connection timing would be difficult.

☞ If your deployment method has the ability to disconnect users after a fixed clock time, be sure to set the time parameter long enough to support the remote user who is dialing in. This setting will prevent premature cancellation.

The following scenario is common: There is a requirement to get an application out to all your machines ASAP if not sooner. Of course, the application is not simple, and the package is a 15MB file. Dealing with the remote user who only signs on through a phone connection long enough to transfer his or her email and get email messages makes it difficult to get a large application loaded. If your deployment methods do not take such a case into consideration, your support will be viewed as poor by the users. To ensure the application deployment completes, you could force the user to stay online, have the user come in to a central location, or use any of the other methods we talked about in earlier chapters. Most commercially available deployment software will provide a good means to solve this issue for you. Transmission compression algorithms, checkpoint restarts, or smaller files all combine to get the data to the target in a quick and workable manner.

Administering applications for the remote user can be just as difficult. How do you determine whether an application is resident on a system if you can't run a report when you need to because the system is not online? Your packaged application includes reporting, so you will need to look at the archived data. Although this information isn't real time, it is reliable as of the report date.

☞ Any reports you create stating success rates should always include a disclaimer line stating the as-of-report date. Not the date of the report creation, but the date the data was recorded. Status reports are only a snapshot of a time period.

Another challenge, domain security can prevent an application from loading. Laptops don't always have domain residency. Configuring them to a workgroup status is common (as is domain residency, true), which will provide added hurdles in getting your applications deployed. Windows security and domain issues will be discussed a little later in this chapter.

The primary needs of the remote user are the same as the local user that is hardwired to your LAN:

- Applications need to be deployed in a timely manner
- A report on the deployment status will be needed
- Application deployment needs to be administered

There are added issues involving the remote user that need to be considered in the solution formula. Obviously, not all issues apply to your environment and there may be additional issues unique to your situation:

- Remote users do not consistently log on to an NT domain or AD. They may only log on locally to the workstation that is either a resident of a workgroup, uses cached credentials, or changes domains often. Relying on domain resident logon scripts or GPO enforcement would not be effective in this case, as the security context and availability of the source location cannot be 100 percent guaranteed. Even locally attached machines can't be 100 percent guaranteed because users still have the annoying habit of turning off their computers.

- A department, office, or other business location may never hard-connect to the corporate network. They may rely on dialup connections (soft-connect) or a VPN solution via the Internet. VPN security and download times are a concern. Stuffing large files down the 56Kbps or 128Kbps pipe during business hours is not advisable if the users need the connection to make money. Bandwidth control that uses a percentage of the available bandwidth and adjusts itself is one solution. Just be careful that the bandwidth control does not extend the download time too much. Scheduling downloads for off hours is a common solution; just ensure that the targets stay online or you have a caching system at the target end of the pipe.
- The remote machine is at home. The system is used at home along with the kids' typing tutor program, the family book-keeping program, several different games, and an email program that is not on the corporate approved software list. The machine connects infrequently to the corporate network, and any application install could have difficulty with any of the other applications installed on the system. Your packaging process should address this issue for file and version conflicts. In this case, your enforcement of installing applications should be reviewed; do you really need to get applications to this user? If all you are concerned about is virus protection, it is easier to focus on protecting incoming data from the server side. Of course, you could also make it a corporate policy for only corporate assets to be used in connecting and any non-standard applications will be disabled or removed (if found).
- A user will only sign on for a few minutes at a time. The connection could be abruptly terminated at any time. Any download that is in progress may be interrupted.
- Mobile targets are hard to hit. For the machine that moves from one site to another, forcing the user to obtain all the data from the "home" site may be prohibitive due to line speed, a firewall, or other network restrictions. You will need to either wait for the target machine to come back to the home site or have a method to automatically change the source location on the fly.

Types of Remote Users

All remote machines or users are not the same. For my own sanity, I tend to split remote distribution targets into five main groups:

1. Common issues
2. Remotely hardwired machines
3. Remotely stationary machines
4. Roaming machines
5. Rarely connected

The common grouping handles most situations. Aim your overall deployment strategy planning at this group. As the target list becomes defined for a deployment, issues regarding deploying to the other groups (2 through 5) will come to light. Fine tune the distribution if needed to ensure the package is delivered smoothly to the entire target list. I'll focus the remainder of this discussion on the five groupings.

Common Issues

This grouping covers the issues common to all remote deployments. In the early stages of planning a software deployment when all the specifics are not known, plan to support this grouping. If the deployment can satisfy the common issues, chances are that the unique opportunities (a positive way of saying “problems”) of the individual targets can be resolved. Whether you are going the custom delivery route or will use commercially available distribution software, you need to address the same concerns:

- Scheduling considerations
- Network transmission
- Deployment status reporting
- Connectivity
- Security
- User impact

Commercially available deployment software can provide most of the answers for you. Because the software vendor is in the business of software deployment to make money, they have a strong interest in addressing common problems and conduct testing to work out the bugs. At least that is the law of supply and demand. If they didn’t address the issues, you wouldn’t have purchased their software.

Scheduling

Scheduling is more than determining how soon the application gets out to the workstations. You also need to consider the end users and the business impact. Some of these issues were brought up in Chapter 4, and I will review them as well as some new concerns in the following section.

Is there a dark window when the user is not on the system and applications can be delivered as needed without interrupting the user? A dark window is a time frame agreed upon by all parties involved. All parties agree on this time as being a safe period. For example, the IT department and the business unit(s) agree every night between 2:00 AM and 4:00 AM and all day Sunday are periods when the computer folks can do anything to the computers (and infrastructure) without negative impact. This timeframe is when you do such tasks as forcing reboots, performing service pack updates, updating network drivers, doing backups (which may impact the network bandwidth), and update applications that the business unit uses heavily during the day. If a user is active during the dark window, the IT department is not held responsible if a reboot wipes out the unsaved report the user was working on for the past 5 hours. Be sure to remind users of the dark window on a regular basis so that everyone stays aware of the agreed upon time. Additionally, consider activating the logon time parameters in Windows; this feature helps to ensure that users are off during the dark window times.

Are there times when the workstation is doing critical functions and can’t be mucked with? This timeframe is not a dark window situation. You need to know which times are critical to the end users and ensure that those times are not impacted. Because we are talking about doing things remotely, you can’t see the target machines in person and may not be aware of what the user is doing at that point in time. Depending on the application and its user impact, your distributions can occur at all times of the day and night. You may choose to stage a large file to the

workstations and activate the actual install at a later time. If the download takes over the available bandwidth during times when the business unit is heavily using the network, problems may ensue. Common times of concern are the beginning of the business day, heavy customer service hours, or defined account close-out times. You know the computing needs of your clients; make sure the deployment does not conflict with those critical times.

How do you trigger the start time of the application? The delivery mechanism will get the application to the target. Your deployment could kick off the package immediately upon transmission completion or it could wait for a scheduled update time. The start time needs to be addressed either by the packaging or the client portion of your delivery strategy. For machines that only connect for short periods of time, delivery may be done hours or days before the safe install time. Your trigger must take into consideration the variances to be met. This process works great if the target machines are online and turned on at the activation time. You also need to deal with machines (such as laptops) that could be offline or powered off at the activation time.

For example, suppose a cashiering application ties into a database server in your data center. The database server is undergoing an upgrade to a new version and a new ODBC on the client is required. The old clients will error out if they try to work with the new structure. Aside from having to deploy the update to hardwired machines that are online and powered up, you need to get the update out to your roaming laptop users who only connect to your network once or twice a week. To prevent needless problems, you stage the deployment to the target machines a few days in advance and set a trigger within the commercially available deployment software to perform the install on Friday at midnight. Thus, you get the data out to the users early but don't trigger anything until later. If you don't have commercially available deployment software, you can use the AT command (or JT from the Win2K resource kit) to schedule the activation of the client update.

Within the packaging, you also take into account the issues of late delivery. For the previous example, you would need to ensure the package kicks off the install if the trigger time has passed. As part of the initial package, you also worked to close the application if it was running, thus the reason for the midnight install. Applications can be closed remotely either through a command line specific to the application; using the NET STOP command for any NT, Win2K, or XP services; or through a combination of the TLIST and KILL commands, as Listing 5.1 shows.

```
net stop TestService
c:\util\kill TestService.exe
```


Listing 5.1: Simple application- stopping commands. The Kill command will stop all instances of TestService.exe.

 **TLIST.EXE and KILL.EXE are available in the Microsoft resource kits.**

For those late installs, you can prevent the user from starting up the application after you stop it by renaming the primary executable. After the install has completed, rename the key executable back to the original name. Although this process might seem like overkill for a midnight update, you need to plan for the late installs that might occur when the user powers up their machine on Monday morning. Remember the earlier discussions about setting the end time or do-not-activate-after time? This is where it comes into play. We've all seen the fast-click users who start clicking applications or who set up their own startup scripts to launch applications fast. The

timing between a startup and your package installation could be enough to have the user startup the application and mess up your install.

Another option for the late execution issue is to use the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce registry key to start up the trigger. Commands placed in the RunOnce registry key will execute once upon startup of the OS and clear out the key of entries. Remember the RunOnce command will activate the commands in the security context of the logged on user. Should you need higher authority levels, you will need to plan accordingly.

 Don't confuse the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce key with the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run key. The Run key will cause the entered program to run each time the OS is booted.

This process prevents a never-ending loop of installs down the road. If your install depends on other tasks or functions happening prior to a particular time, use caution to not start the application early. Have the RunOnce key activate the trigger script only. The trigger script then compares the current time to the scheduled install time. If current time is after the scheduled time, the install runs. If the current time is less than the scheduled time, the trigger reinserts the RunOnce registry entry and terminates. This way, you are ensured to run the critical application prior to the user starting anything. Listing 5.2 and 5.3 show the commands for this process.

```
@echo off
regini runonce.ini
```

Listing 5.2: This line will call runonce.ini, which will insert the timer trigger into the RunOnce registry key.

```
hkey_local_machine\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce
test = c:\temp\TestTrigger.cmd
```

Listing 5.3: The registry format to insert the timer trigger into the RunOnce registry key.


We've now defined dark windows, start times, and business critical times. You need to make sure the installation does not creep into those critical times. Can you force an application to stop installing or stop processing at certain times? That is rather tricky. It is more of a timing issue on the startup. Looking at the issue backwards is the method to use. As we discussed in the last chapters, you should have a good idea of the time it takes for a package to install. Add a fudge factor of 10 percent or so, subtract the total time from the must-end-by time, and set the install drop dead time accordingly.

For example, if your application takes 30 minutes to install, and your dark window is from 2:00 AM to 4:00 AM, you want to ensure the installation does not exceed the 4:00 AM cut off time. A 10 percent flex factor is only 3 minutes, so round it to 5 minutes. All the clients are on fast connections, and the package is not large enough to worry about transmission delays. So at worst case, your application needs to be running by 3:25 AM to make the must-complete-by time.

Network Transmission

If the application is only going to remote users who are hardwired and online to the network at all times (such as call centers or corporate buildings), your primary concerns about the network

will be the bandwidth considerations and the scheduling time of the deployment. However, if your deployment will go to remote users, be sure to identify the worst case transmission line to be dealt with. I don't mean to figure out the exact worst data line, general assumptions are good enough at this stage. Sending a distribution to a branch office at night through a dedicated 128Kbps link will ease the concerns into the same as the hardwired scenario. If it is to remote user dialing in, you need to understand the size of the files, the delivery method (compressed, checkpoint restart, or bandwidth controlled), and how to handle interrupted transmissions. Always plan on a transmission experiencing an early termination or a dirty phone line so that you're prepared when it happens. Commercially available deployment software confirms the transmitted file is intact prior to acting on the package. If you are scripting your own, at least do a simple compare of the file size between the original source and the file on the target. Checksum comparing is best in these cases.

 You can find checksum utilities from any good utility site. Search for checksum at <http://Download.cnet.com> or <http://zdnet.com>.

 Windows Installer also supports a checksum confirmation using the /f switch.

For the most part, the transmission concerns are out of your hands. You really don't have much control over what the local telephone company does with their lines. You can, however, plan how to handle the common file interruptions that are bound to happen. Comparing the file checksums, checking on segment completions, sending email to users about the deployment, and making the deployment a manual pull could assist in ensuring that the download completes.

Deployment Status Reporting

Deployment status reporting doesn't need much explanation. The primary issue is ensuring you get a status ASAP, hopefully while the user is still online. If not, you need the report on the next connection. Yes, most commercially available deployment software resolves this issue for you. If the collection point doesn't report back right away, a status is sent the next time the software client can talk to your collection point. If you are scripting your own deployment package, you need to get the data yourself.

The biggest determination is the level of reporting you require. Do you want simple success and failure codes? Or are you looking for confirmation-specific files or verification that registry entries have been properly inserted? If your software does not present the data you need, attach some logic to the process to pull the data and report it in a format you like. This format is up to you, and the possibilities are endless on how to create the report data.

Connectivity Durations

There are two types of connectivity durations: too short and long enough. Many users only connect to your corporate network long enough to transfer whatever data they are responsible for. Sales spreadsheets, timesheets, email, review of the corporate Web site, budget data, and maybe the latest joke .wav file or two. Then they log off and go about their business. The entire session could be completed in 10 minutes or so. In a day or so, they repeat the cycle. With short connections, it can be difficult to transfer application packages if they are large.

The long enough connections are a blessing. These users log on and stay online for hours (sometimes days) at a time. This timeframe gives you time to do all the automated administration tasks you would like to perform in much the same fashion as the hardwired remote user is dealt with. You have to deal with a dropped connection or two, but you know that the next session will be long enough to recover and continue. Obviously, the long enough connections are not a major concern. It is the too short connection folks that are your challenge. What you do to support the too short connection crowd easily translates to the long enough connection crowd, so I'll focus on the short crowd.

The delivery process will depend on your standard delivery mechanism. With the numerous and varied methods of deployments available, the answer is going to be just as varied. So the answers and examples I'm going to present to you are only suggestions, there are numerous other methods that will work just as well. Your choice depends on your environment and structures.

If your commercially available deployment software handles remote users, the task is not so bad. The client piece on the machine will have a trigger method of some sort: a timer that checks for a connection to a destination address every few minutes, a listener that waits for a broadcast poll from a server, or maybe a script to start up the commercially available deployment software client when the user activates the dial-up script. No matter the trigger, the end result is that the client piece talks to the deployment software and begins processing your instructions or the updates as you have scheduled. Installation then follows your normal processes. This easy method is another good reason to pick up good deployment software.

If your distribution is done through scripts, you will need to get creative. The final solution will depend on how your users contact the corporate network:

- If your users dial directly into the network using a modem-to-modem connection (not via the Internet), you can set up a trigger from either end of the connection.
- From the client side, you could set up a scheduled time for the computer to dial, connect to your network, poll for any updates, and disconnect. This option works best for the remote stationary client who is connected to a phone line at all times.
- If your users are contacting the corporate network via the Internet, you can tie in to the security application the users activate. A VPN or token synchronization appliance are two examples. In those cases, you can activate the deployment method from either end, though triggering it from the client might be easiest, depending on the host methods you are using.

Looking at the modem-to-modem connection, your triggering processes will either be keyed from the client side or the server (or corporate) side. In either case, an event will occur to indicate a connection has been established and the deployment process can begin.

Other alternatives include having a polling process run at a regular interval or including a service or executable that always runs and monitors various API calls or processes within the OS. If you don't use commercially available deployment software or your deployment software does not have a trigger method, I've listed a couple of simple methods that can perform the functions for you without much of a problem.

Because users will be calling in via a modem, having a script for them to use is the easiest method. Include an after-dial script to run a process calling your custom distribution client. The client should do a check for connectivity to a known point within the corporate network. A ping

against a static IP, a server name, or a flag file on a server are all easy items to check for in a loop process. After the connection is made and the known end point is confirmed as reachable, the client piece can proceed with any downloads you have. The advantage to this method in a manually activated dial-up situation is the automated check and response the deployment process has. It does not depend on a polling interval that might miss a short connection window, and it does not rely on the end user to activate a second process. It is basically all-inclusive and functional. One downfall is the ease of editing scripts. Scripts are subject to modification by anyone who can do modem script edits, and your deployment client might be edited out.

Although I mentioned that the dial-in script could call another batch file, it could just as easily start a deployment service. If your client piece is configured as an executable and installed as a service on the target machine, it can be activated by the dial script. By leaving the service stopped until the dial up is completed, the service does not try to connect when there is no open channel. In short, it keeps the computer from burning cycles on a process that will never succeed in contacting the server. There are a number of methods to convert an executable into a service; the NT Server 3.51 and 4.0 and Win2K Server resource kits all contain two utilities, SRVANY.EXE and SC.EXE, that you can use.

☞ SRVANY.EXE and SC.EXE are available in the resource kits. Use the documentation in the resource kits for the syntax. Check out the following Microsoft Web sites for more help <http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/prodtechnol/winxp/proddocs/sc.asp> and <http://support.microsoft.com/directory/article.asp?ID=KB;EN-US;Q137890&>.

Client pieces that rely solely on timed polling events need to be carefully adjusted and monitored. In a timed polling process, the system periodically checks for access to a known location. Either a ping or a query to an HTTP location is used. Obviously, if the machine is not online, the polling process will fail. If the machine is online but not connected to the company network, the process still fails. Setting the polling time too short could place a burden on the client system. Setting the polling time too long might miss the connection window. When the polling process is a service that activates on boot up, the initial queries will fail if the system needs to dial in first.

Take the case of a remotely located laptop user with a polling service set to poll at system startup and then every 30 minutes afterward. The first poll will meet with failure as the laptop has not connected yet to the company network. The user dials in, gets authenticated to the corporate network, does the normal email transfer, and disconnects from the network. 10 minutes later, the polling service queries again and meets failure. It just missed an opportunity to connect. The solution to this problem, assuming the polling service does not create burdensome overhead, is to set the service to poll every 1 to 2 minutes. Leaving it to poll at startup is a good idea as the machine may be hardwired to the LAN at a company building once in a while.

If the polling service could be activated by a dial-in script, the short polling period issues would not be a concern. Connecting the service to the startup of the email application is another way to catch when users are online.

On the corporate side, we will assume the users are calling into a central modem pool or a monitored location. I'm naturally paranoid about unauthorized access to the business network and have a problem with the general user population being given the ability to dial in to private modems. If they dial in to a central modem pool, monitoring is easier and you can prevent the

common snooper from gaining access to your systems. (For a brief reminder about security, see the sidebar “Modem Security.”)

Modem Security

One side step from the connectivity subject: modem security. If a user is calling in to your network, make sure it is the right user! Have some sort of security—don’t leave things wide open. Password access (with a password that changes regularly), call back to a specific number, or token access are all easy to implement and not really intrusive to the user. *Be sure to monitor the logs daily.* Daily monitoring of who called in to your network is a requirement so that you know when someone has hit your network.

OK, back to the subject. I’ll take the point of view that all your users call into a central modem pool. Or maybe several geographically based modem pools. Either way, access is limited to getting in via a monitored and controlled method.

- If the dial-in security is call-back controlled, the deployment trigger script is activated when the call back is completed. After-dial scripts are common, so this part should be documented with the modem software you are using. After the connection is made, your deployment method can investigate the client machine and perform your selected actions.
- If the modem pool relies on password or token registration, once the access port is open to the client, a polling script can be used to query the status of any mandatory software updates. You would want to query against the opened port. Scanning the logs for successful activations can trigger the call against the user ID or the machine name. Details will depend on the security method and program you are using. Check the documentation for methods to use.

The question of infrequent connections has been answered: You need to have a trigger activated depending on your deployment standards and connection method. This will get the package download started.

What about the short connection time? Now the fun starts. You need to really know your users to decide on the proper method. How long is the average connection time? What about the average connection speed? If all your users are coming in via cable modem or DSL and the application is only 500Kb, chances are you won’t have to worry about short connection times. The download will happen quickly, and the user won’t even know it. Problems start popping up as the package size increases and the connection time slows. If you use commercially available deployment software, the issue of dropping a connection during a download is addressed.

Deployment software offers two common methods of transmission support: Prism Deploy has a unique delivery scheme that is able to download files across a 56Kbps line in a short time. According to Lanovation, the schema is successful enough that the company doesn’t bother with check point restarts or parsing out files into multiple segments. The file either gets downloaded completely or it does not. (I discussed the specs from Lanovation in Chapter 4.) For most application installs, this issue won’t be a problem for the users that are online for 10 or 15 minutes at a time. Marimba uses a WinZip type of compression and relies on checkpoint restart with checksum verification for resolving interrupted transmissions. This method means a user can sign on for a few minutes at a time and will get a little more of the application each time until they get the complete download.

If you are doing your own downloads, you have to deal with the problems of interrupted downloads yourself. Some suggestions to aid in the process:

- Break up the large files into smaller chunks and assemble them on the target system after they all arrive.
- Use a compression method that squeezes the data into the smallest foot print possible. The WinZip and PKZIP utilities are commonly used.
- Send a wrapper process down first to monitor the download of the individual segments. The wrapper would be activated by the deployment trigger and would be the one to call the individual sections. On each connection, the wrapper would check to see what has been downloaded, then call the next section in line. After the wrapper detects that all the pieces have arrived, it can decompress everything and activate the installer.
- Your package could be designed to transmit set files. When the package starts up, it checks for the existence and size of the defined files before it begins the install process.

Stale or out-dated data is one big problem with infrequently connecting machines. The machine may not connect long enough to ever fully download the package or checks in so infrequently the data share point has changed content. If you split your packages into segments, be careful not to overwrite older segment names with new data, otherwise a system could end up with segments from two or three (or more) versions. Then you have the issue of remote support for a highly confused application.

Again, the issue of stale data due to partial downloads is usually addressed by commercially available deployment software. The processes from the vendors either reconfirm the partial downloads or restart the download from the beginning. Either way, the vendors remove the headache of data validity for you.

For machines that dial in but remain online for long periods of time, life is easier than for those in the previous scenarios. You still have the concerns of the trigger piece, using commercially available deployment software or custom created methods, and activation of the package on the machine. The entire scenario takes on the tone of the hardwired remote user. The long connect clients are available and can be accessed; they just take a little longer to get the data due to the smaller connection pipe.

Small remote satellite offices or work-at-home users fall into this category. They generally contact the corporate network often and remain online for longer periods of time (hours or days). Their desktop machines don't travel much (hopefully), and the administration is simpler. The primary concern has to deal with the slower line.

In all cases, ensure your deployment method has a reporting structure to report when the machine completes the install. Because the machines don't stay online all the time, they need to report immediately to a collection location. Trying to gather installation statistics when a machine is not online will add extra overhead to your already busy day.

Security

Security for software deployments is an area not to be over looked. Deployment security is concerned with network and Windows security. If you work in an organization in which the IT responsibilities are split to different departments, staying friendly with your network staff will pay off. If they make changes to the firewall access list and accidentally block a needed Windows port (say the server side TCP 135 RPC port) or block a port range above port 5000

after you have set your remote clients to use RPC calls above 5000, your remote users may suddenly have difficulties ranging from not connecting at all to package delivery failures.

You could easily spend days investigating the distribution failures. When only one router access list is changed, it will complicate the issue even further, causing you to experience gaps in your hair growth as you slowly pull out the individual strands. Stay in the loop on changes to anything in the network that could impact your data flows. Ask the network folks to include you on any network change notifications they send out. Simply review any changes for potential impacts and save yourself wasted life cycles in troubleshooting preventable problems.

The other primary concern is Windows security. File permissions, domain trusts, transient trusts, and a host of other Windows security features can present unique problems to your distribution efforts. For stationary or static machines, your only concern is the user ID permissions: can the user access the source files or can the commercially available deployment software client contact the proper server? For roaming machines such as laptops, your concerns multiply.

How complex is the domain structure of your environment? If your company works with one domain, domain security aspects are not as much of a concern. When you add another domain and populate it with users and machines, you have complicated your delivery process. The juggling act between accessibility and install speed are two more bowling balls you get to keep in the air when sending out distributions. As if you needed any more to deal with.

As Figure 5.1 illustrates, in the simplest form, if Laptop is part of Domain A, and the applications source is in Domain B on a machine called Server, you need to ensure Laptop has access to Server or the trusts between Domain A and Domain B allow machines to talk to each other. Otherwise, the application install will fail due to Access Denied errors.

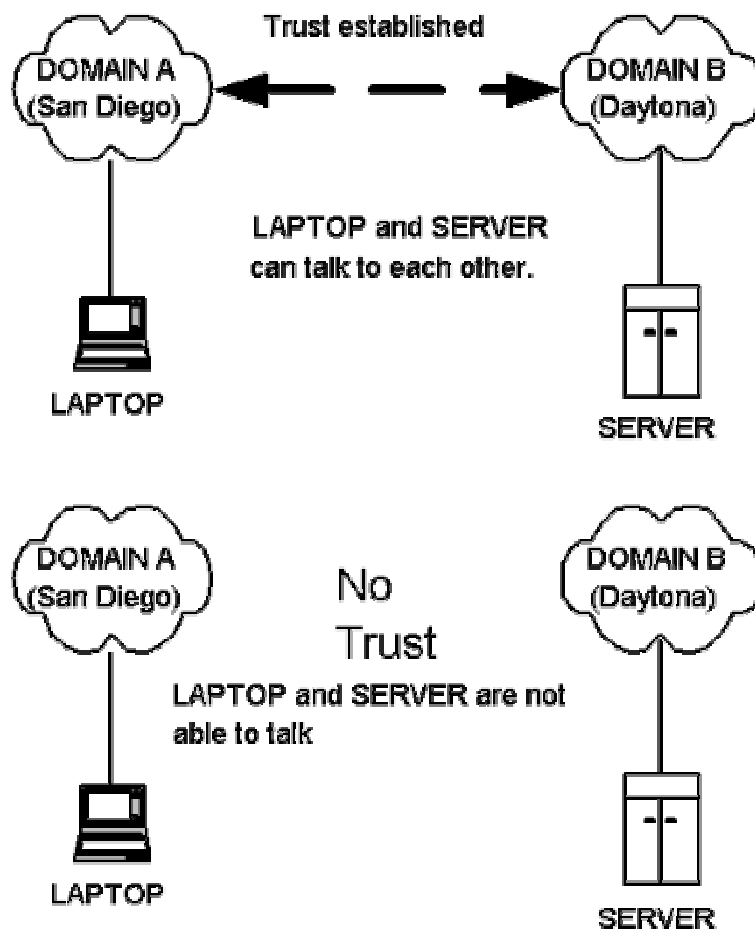


Figure 5.1: How domain trust affects deployments.

Another solution to the same problem is to place the files on a standalone server to provide access to the group Everyone from both domains. HTTP is becoming a popular method for distributing files. Many commercially available deployment software products are moving toward that direction. Either the entire deployment software process is HTTP based or it utilizes the ability to pull the source files from a Web server. This process can provide an easier method for complex security issues as the Web server connection can be configured independently of the Windows domain/file security structure.

To expand a little more on the example, assume the following scenario, which Figure 5.2 shows: Laptop is still a member of Domain A, which resides in San Diego. The distribution method points Laptop to Server A in Domain A to obtain all the applications. Laptop travels to the Daytona office for a long-term visit. The Daytona office is part of Domain B. All the machines in Domain B get their applications from Server B. To make it a little easier, assume there is a trust setup between the two domains and there is a slow link connecting the two domains (128Kpbs or so). Are you still with me? Do you let Laptop pull its applications from Server A in San Diego over a slow link or do you move Laptop into Domain B in Daytona to obtain the applications faster without impacting the slow network link? Leaving Laptop in the Domain A would be best if the Daytona visit is a short one. If Laptop will remain in Daytona for several weeks, moving it into Domain B would be the best method.

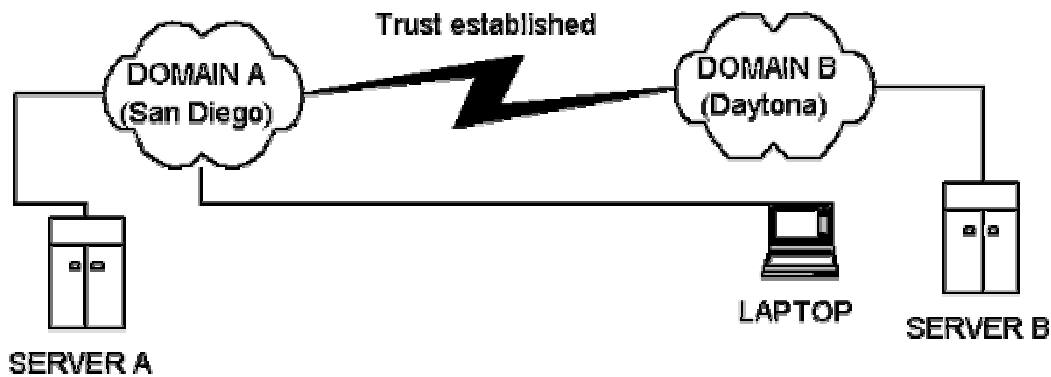



Figure 5.2: Showing the distance Laptop must travel to get data from Domain A Server A.

As you can see, the remote user presents you with added security concerns. If your applications are pulled from a Web server, do you require all users to connect to the corporate network to reach the server? Or do you place the server in a DMZ and grant restricted access from the Internet? Having the users authenticate to your network will provide a level of security for the Web server and is highly recommended.

 A word of caution about opening a distribution server to the Internet for application installs. Be very certain your security is tight. You don't want the entire Internet using your server to download licensed applications. The legal ramifications can be painful.

If you opt to send applications to remote users on a CD-ROM for them to load, you now have another facet of the security issue. Aside from the obvious issue of the CD-ROM getting lost at the airport and someone else installing Office XP using your corporate license number, you need to be concerned about any corporate data present on the CD-ROM. Weigh the potential of a lost or stolen media against the importance of having sensitive corporate data available on the easily transferred medium. Users tend to realize rather quickly they have lost their laptop with all the data and connection ability at the airport. You can take steps to prevent the laptop from accessing your network and begin changing any passwords that may be compromised. A lost or misplaced CD-ROM can go unnoticed for a long time and present an open window to your company without you knowing of it.

If you send out media with sensitive data on it, encrypt the data and keep all the passwords off the media. Send the activation codes in an email or letter or phone call. An encrypted CD-ROM without the activation code is useless to 99 percent of the world that might find it. Hard-core hackers can still break into it if they find it and want to put in the effort. The issue then becomes theft and not just a lost media situation.

The credit card industry is a great example to follow. You get a new card in the mail but it is useless until you call in and activate it. The credit card company requires you to call in from your home phone so the caller ID can be verified. Or they talk to you and ask for specific data to prove you are you. The PIN is sent in a separate mailing, usually in a plain envelope with no account data associated with it. If it works for them and the millions of users they deal with, it

can also work for you when shipping an encrypted CD-ROM of your company payroll information to remote pay clerks.

To repeat a concept from Chapter 3, when a package is installing, it needs to have the proper authority level. In a locked-down environment, users generally don't have the authority context to adjust the registry. The package would need to run under a higher authority level of some method. Security is a large issue that often comes in conflict with distribution methods. Locking down a user ID to prevent the user or unknown users from getting access to your system(s) can also prevent the distributions of approved packages. Windows Installer from Microsoft provides the ability to install the packages under an administrative context. Test your installs under a user's context to ensure it works. When you are doing remote distribution to clients who will be contacting a source location from various end points, test all the possible versions you can to ensure the security platform is not a hindrance to the install.

User Impact

The user will be impacted no matter what you do. However, you can limit the impact as much as possible. Your packages should take into consideration the normal distribution issues to be dealt with as we discussed in past chapters. Remote users place an emphasis on delivery and timing.

In the section about scheduling, the issue of timing was discussed. Dark windows, cut off times, and system critical use times were reviewed. Those are issues of user impact that can cause problems. You don't want the package to be installed and force a reboot during the business day. Remember if the target machine is not being used to make money for the company, your budget can't buy all the neat toys you want.

If your distributions have any chance of causing an impact to the user, make sure the user is aware of what is happening. Pop up message boxes, email notifications, or phone calls can be used to ensure that the user is aware of distributions. How much you use these methods (and others) are up to you and how much you want the user to know of the distributions.

Let's talk about pop-up message boxes. These are wonderful items if used properly. In principle, when a package is being installed, a pop-up box would appear indicating the status of the install, a progress bar of the download and of the install, and any informational data the user might need. The user then knows what is happening on his or her system and can adjust accordingly. I'm bringing this up in the chapter about remote users to emphasize the point of communicating with the targets so that they help you out.

If a package has the potential of negatively impacting the user (causing a reboot or a termination of a program), decision pop-up boxes would be appreciated by the users. Having the option to defer a package keeps the users happy as you show consideration for their needs. I would strongly suggest you limit the deferrals to once or twice and clearly state it in the pop-up box. Otherwise, the user might never allow the application to install or reboot the system.

For large downloads that will take a while on a slow line, pop-up boxes with a download status bar help the user. Assuming downloads are silent, the user may not know when the download is complete, and either terminate the connection too soon or keep the connection up longer than needed.

One option is to present a deferral option to the user with a pre-defined delay time. If they click Later, a second box appears stating the package will activate at a specific time (such as 4:00 or 5:00 in the afternoon) or within an hour. The second box states very clearly there will not be a

second chance for a deferral. This option gives the user plenty of time to contact you if there are issues or to plan for the impacting action. Either way, the user is generally appreciative of having an option to install the mandatory application at a time a little more convenient to them.

If you do use deferral boxes for a package that does reboots, place a little logic into the package. Assuming the package has done the install and only needs to reboot and the user defers the reboot to later, your countdown timer needs to identify if a reboot was done earlier than the program scheduled it for. The RunOnce registry key can be used to clear out the countdown timer if a reboot happens. Otherwise, your attempt at good will is negated by the user doing a reboot and you forcing another one later.

Pop-up boxes, while great, can be annoying if they interfere with actions. You've experienced the annoyance. You are in the middle of doing some complicated action and a message box appears, taking the focus away from your active window and interfering with your work. A minor annoyance, but one you need to be aware of. Use of pop-up boxes is good if done within reason.

Keep in mind how you handle pop-up boxes on your own systems. When a message comes up, do you always read the lengthy ones or just click OK or hit ENTER and take the default? The users you are dealing with are even less likely to read and react to messages than you are. Make the messages short and clear. If a default action could cause a problem, have a confirmation pop-up box verify the action.

Remote Hardwired

Let's address the easy remote machine first. This machine is

- Connected to your LAN/WAN
- Physically located a distance from your location

This type of machine is in the corporate environment but located in a different location from where you are. These machines generally stay online most of the time, have a good fast network connection, and don't move around much. They classify as a remote machine due to the geographically different location. The major consideration is getting the package to the machine. Of all the remote machines you need to deal with, this machine is the easiest. Why? Your standard deployment process can get the application to the machine (we covered this in the last chapter) without any unusual considerations.

The common issues I discussed earlier apply to this class of machine. Take particular care about the security aspects. Normal Windows security will apply if the target is in a different domain from you or your package source location. Your testing should have covered the delivery process and the application install, proving there were no problems.

Evaluate the packages. If there are reboots or major interruptions to the system that could cause issues with the end user, be sure to take into consideration any difficulties that might be encountered. Your delivery mechanism or the package can also deal with the common problem of leaving a floppy disk in the drive (by turning off the floppy drive search or setting the search order differently) if your application performs reboots. With regard to the floppy search order, you will need to check your hardware specs, some vendors provide methods to turn it off.

Remote Stationary

This type of machine:

- Does not move around much. It stays within one geographical location.
- Is not part of the hardwire LAN/WAN. It depends on a connection method that is not always on. Dial-up, cable modem, and DSL are examples of the connections used.

The main difference between the remote hardwired and the remote stationary machine is the connection. Unlike a laptop, which is meant to move easily, the remote stationary machine is a desktop unit meant to stay in one location most of the time. Picture the desktop units that are taken home for remote users to work out of their homes. Or a small office with desktop units that only connect via a modem. The issue of variable connection speeds is not a major concern because the machines will use the same method each time. What makes this class of machine difficult to administer is the fluctuating online status.

More of the challenges we discussed in the Common Issues section start coming into play with the remote stationary machine:

- Multiple user IDs
- Questionable domain or workgroup membership
- Controls on local security are not as strong or may have been comprised
- Uncertain levels of software due to users having a higher level of access on their IDs

Compared with the roaming machine, this machine group is the easier to work with. Most machines are in a known state when they leave your imaging process, and you know how the machine will connect to the network.

The concerns will center more on when the user connects. On each side of the extreme scale, there is the system used at home on one end and a business unit system that stays connected via a phone line on the other. While both pose unique issues with packaging and implementation, they are not any different than what was covered in earlier chapters. The concern is how to get the application data to the machine.

I'm going to stereotype the two extremes for a minute. The business system is going to remain online for long periods of time and have a good quality network connection. You will be able to reach the machine during known connection times, say during normal business hours. Package deliveries can be scheduled around the known connection times and easily worked with.

The home office type of machines is more of an unknown. The network line is a lower transmission level, probably depending on dial up via a 56Kbps modem. Connection times will not be as consistent as the business unit system, although the durations will be long enough to support distributions. The home office side of the scale will be connecting for more than 10 or 15 minutes a day.

This class of machine begins to loosen up the controls on the system (or the controls are forced open by the user), and your packages may begin to experience difficulties with conflicting and unknown applications. Given security on the machine is not as tight as a hardwired machine located in your corporate world, your packages will need to clearly define connections and security concerns.

The transmission issues outlined in the Common Issues section will play a large role in your distributions. Transmission interruptions will begin to occur more often than in the hardwired machines. Calls to a central repository that has been locked down may begin to fail if your packages don't explicitly define the access right for the connection. Pay closer attention to the packaging and access rights needed to install the applications. Your transmission integrity will begin to be tested but won't be as challenged as it will by the next class of machine.

Roaming Users and Machines

Roaming users and machines create their own unique issues. I'll discuss users that roam from machine to machine in the next chapter. For this chapter, the focus will be on roaming machines. In other words, laptops. Laptops are difficult to administer. When administrators are talking about laptop administration, colorful adjectives often preface the description of the task. Laptops present many challenges:

- Connecting to the corporate network may rarely occur. If ever.
- Laptop users may have a higher authority level than a non-laptop user. Given the users need for flexibility in installing applications, the laptop user is given added access rights. Maybe not full Admin level, but Power User group membership (which is nearly Administrator) for the machine.
- Passing a laptop from one user to another is common. A laptop pool to share the expensive asset solves many equipment purchase budget issues although it increases the level of administrative overhead for you. Some organizations re-image a laptop every time it comes back into the pool to reduce the influence of the previous user (and remove the various programs that might have been installed). If the shared laptop is not re-imaged each time it changes hands, the system integrity tends to diminish with each favorite game install or customization applied.
- Domain or workgroup residency may change.
- The logged on user may only authenticate to the laptop and not to an NT domain or AD structure. If the user logs on to the disconnected laptop with the domain or AD account, the credentials are cached. This behavior can cause issues if you rely on the logon process to apply dynamic changes to the machine. If the user logs on first, then dials in, the logon process won't be restarted. Yes, you can force the user to log on via the dial-up process, but be aware that most users quickly figure out how to get around this limitation.
- Connection speeds vary all over the map depending on the needs and line speed at the moment. Thus, relying on a set network timing parameter for a laptop delivery can be difficult.
- In the ideal world, the machine name would identify the laptop from a desktop unit. In the real world, it is often difficult to identify a roaming laptop from a stationary desktop machine. The delivery mechanism needs to have a means for identifying the machines. If the commercially available deployment software has the machines grouped accordingly, any checks you put in are for those non-standard imaged machines. For applications targeted to a laptop rather than to the desktop machines, you need a fool proof method to determine the difference. A corporate standard stating all laptops are from Compaq and all desktops are from Dell (for an exaggerated example) would make your job easier. Any

inventory you get will identify the differences immediately. Your packages/delivery methods could also check for vendor-specific drivers for video or network cards. For systems created with your standard image process, a common practice is to create custom company keys in the registry or add environment variables to identify the system as a laptop.

☞ To help positively identify laptops, check for the existence of unique laptop registry keys such as `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\Root\COMPOSITE_BATTERY`.

The roaming machine will stress your delivery process. Connection times are uncertain, the quality of the connection will be unknown, the duration of the connection will vary, and the process of calling source locations can be confusing. Remember the example of Laptop in Domain A when it visited Daytona in Domain B? The machine was roaming between the two domains but was generally static in each domain for a period of time. Keeping the laptop updated with data and applications was a matter of sending the laptop to the proper domain location after evaluating speed and security. What happens if Laptop travels between Daytona and San Diego a couple of times a week? And if a large chunk of your workforce are roaming machines?

The emphasis now moves to the infrastructure instead of just the machine. Your data source locations will need to be configured for the best access from various domains and connection styles. Centralizing all the data in San Diego, for example, may be fine for the folks on the West Coast, but the employees on the other side of the country will not have the same speed. Running the fastest dedicated network line you can afford between the Daytona and San Diego offices will help the speed issues, as Figure 5.3 illustrates. Have the roaming machines dial in to the closest office and connect via the infrastructure that way. Your central data store can stay central and the users will have a good connection.

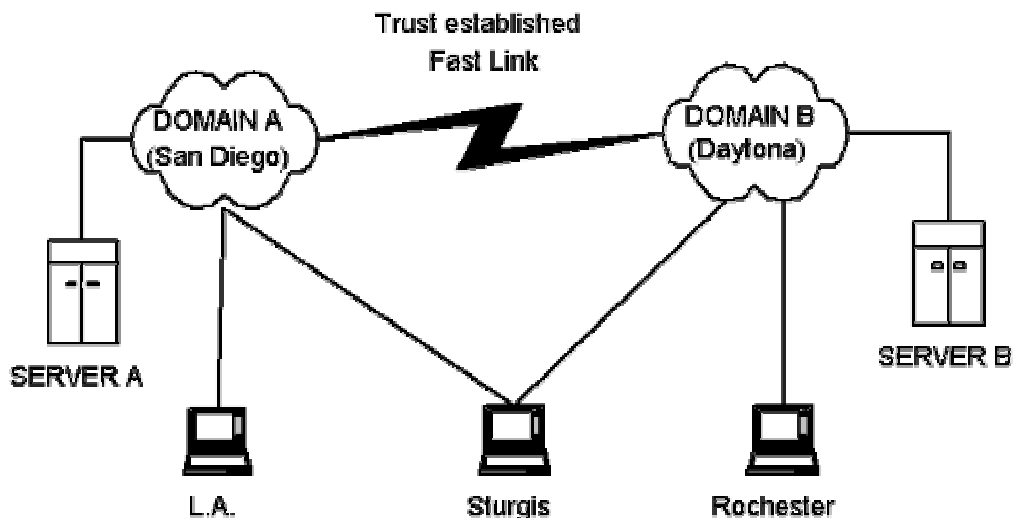


Figure 5.3: Remote users call into the nearest access area. Distributions can be based upon which access point they call into so that the closest server is used.

For the roaming machines that have data sources statically defined and that go from company location to company location, you still must face the trade off between speed and ease of administration. If your company uses DHCP, one solution is to compare the IP address of the current connection to the previous connection. By creating a master list of the IP subnets for each location, you can redirect the source locations for each machine on a dynamic basis. If the IP is the same, the machine can grab its data from the previous defined sources. If the IP address changes, a script would locate the new source server and direct the machine there. The subnet change process could be a part of the commercially available deployment software or your custom solution. Including it in the logon process has been done. Keep the subnet source list central so that you can update it without replication worries.

Yes, it takes a bit of time to set up. Large companies with numerous subnets will take a while to configure the initial setup process. You wouldn't have to check each octet of the address, only to the level your network is subnetted. If all your West Coast machines are part of at 10.x.x.x network and your East Coast machines are part of 192.x.x.x, you only need to check the first octet. Some commercially available deployment software products do this check for you in their structure. For dial-up situations in which the calling machine uses the ISP defined IP, you could either ignore the changes, which would force the machine to the last defined source location, or you could define a default location. The speed issue would become moot at this point due to the possibility of numerous masking schemes you would need to detect.

Another variation to the IP checking scheme is to have all the source locations hidden behind a URL pointer. The calling application installer or wrapper would contact the initial source location and offer the IP address. The initial source location wrapper would determine the closest source to use based on the IP, and send the information back to the calling program. The calling program then contacts the defined source system and begins the install.

Sorry, no easy answers for this one. The call is yours to make, depending on your environment needs and the needs of the users. Commercially available deployment software can provide the answers for you in a defined process, or you can create the added details if you are creating your own distribution method. Personally, if I were involved with a company that had the issues of speed over location, the costs of obtaining a deployment software compared with the administrative factor of the custom solution would help justify the deployment software purchase.

Rarely Connected

These machines will provide all the encouragement you need to make your vocabulary more colorful. This category includes both desktops and laptops, in your corporate buildings and at home sites, and connecting via the hardwire LAN or the worst dial-up connection possible. The commonality is the lack of connection. These machines rarely connect to your network. When they do, it is with the intention of grabbing one or two pieces of data only.

Rarely connected machines may connect once a week or once a year. With few exceptions, the machines are not directly involved in generating revenue but deal with functions such as word processing. My experience has mainly been with personal home machines being used to dial in and get email or a document for a user who is sick at home. I will not even begin to discuss the security risks and the connection issues for these users. Those are pretty obvious.

For software distribution, you can assume these machines don't have recent versions of any applications used within the company. The client software for your distribution method may not exist on the machine. The domain residency is also in doubt.

In many cases, it is easier to manually find the needed data for the machine and send it down after it connects. For those production-related machines that need to be updated with your distribution method, your challenges are many. The following list provides my recommendations to support this class of machine:

- Use status boxes for downloads. Providing data to the user will aid them in determining when they can disconnect from the network and will help prevent early termination errors.
- Ensure your packages do validity checking of the download. Otherwise, you may have several parts of distributions spreading over several versions. Trying to install the entire version spread will cause errors and generate support calls.
- Use a pull-type technology for the software distribution client. When the client connects and contacts the distribution points, the client piece will help sort out which packages need to be delivered. Relying solely on a push technology may cause problems, as the advertisements on the push server can age and terminate before the machine contacts the server. Although this problem can happen with pull processes, it tends not to happen as often.
- Be sure your packages are all-inclusive and not incremental in nature. Let the client determine how much to install each time. Incremental packages work best for clients that contact the network on a regular basis. Rare connection machines could miss several increments, causing the package to be missing critical files. If the data being sent is application data in nature and some increments are missed, there could be aged data deleted in some of the missed versions. The result is unreliable data.

Many companies have set policies on vacant connection durations. If a machine does not contact the core network in 1 or 2 months, it is not allowed on the network. With the advent of portable token access methods (thumb prints, synchronized RNGs, and so on) and Internet front ends, it is becoming more and more common for a user to be on a borrowed machine when they contact your network. User may never be on a machine again, but they expect the connection to function as if they were on their machines. We'll cover the roaming user in the next chapter, stay tuned for that one. In the mean time, be aware users can call in for assistance on machines that do not properly belong on your network and are not configured for your methods.

Your machine access policy needs to consider how to handle the rare connection. Do you require a known registry key or file as part of the validation process? If so, try to make your software distribution method the key to be used. If your deployment methods are not on the machine, the machine should not be allowed into the network. If machines are allowed on the network, do they need to load your commercially available deployment software? The answer will be in the expectations of the user to obtain packages and distribution data.

Setting up a common Web page that contains the packages for such users will ease the issue. The user can then obtain needed packages in one-off situations by going to the Web page and activating the packages manually. You can also include the distribution client on the page (or have it as the only item there) so that the users can install the client themselves, then let the automation take over for the package delivery.

The main issue with the rare connection machine is its unknown status. It may be a part of the domain at one time and associated with the right groups to activate automatic deployment of any distribution clients you have. GPO and logon enforcement methods will ensure the machine obtains the client. If the machine is removed from the domain or AD due to lack of use and is allowed back on the network at a later time, it may not be properly added back to the domain or AD structure. The machine may still show residency in Domain A, but the SID is not in the accounts database, so the machine is not getting the security tokens passed properly. Unless an administrator adds the machine back in properly. See the problems? If you rely on a human process to get a machine included in all the right groups to ensure the mandatory deployments are done (such as anti-virus software updates), humans will find a method to side-step the process because they “do not have time to wait.”

The best you can do is to provide a number of methods to resolve this situation. Set company policy properly, communicate to everyone on a regular basis how to ensure the rare connect machine can be included in the software-deployment process, check your logs, and set up options for others to easily activate the process for you (Web pages, email archives, or known source locations).

A Real World Example


Dealing with remote users is only as difficult as you want to make it. By insuring your application package has distribution intelligence and accounts for your worst case remote scenario, your distributions will work well. The basic distribution system needs a client piece and a server location. Whether you are using commercially available deployment software or a process you create, the distribution system needs to know when the connection is available.

In the fairly closed environment of the business world, your targets and network are of a known quality. Thus, your distributions are going to a known quality. As an extreme case of distribution flexibility, imagine sending your distributions out to a world-wide audience. The number of machines are unknown, the type of systems are generically known but not the specifics, the contents of the target systems are unknown, and the connection methods include every known method available. Your package must be available to do client data updates on a 7×24 basis as the clients call for it. The reporting process is as-of-report-date and updates a live database with the latest information at all times. Dreaming, you say?

Check out the SETI project. This project was the first world-wide computing effort that strives to take advantage of spare CPU processing cycles on Internet-connected computers to search for extraterrestrial intelligence (hence, the acronym). In short, a radio telescope records data signals from outer space, and sends the recorded bits to the University of Berkeley. Anyone wishing to participate in the effort downloads the client piece for their particular type of computer and OS. The client piece contacts the servers at Berkeley and downloads a chunk of the recorded data. Processing is done on the local computer. After the chunk of data is analyzed on the local computer, the client contacts the Berkeley servers, uploads the results, reports on the status of the processing, downloads another work unit, and continues the processing.

If the computer contacts the Internet via modem, the client piece has options to dial out. Control is given to dial at certain times and to disconnect after downloading the work units. For systems that use an always-on method, the client contacts the Berkeley server as needed. Proxy capabilities are included in the client so that firewall situations are resolvable.

This process is an example of all the discussed extremes. The program handles the reporting, the distribution to multiple clients, the automated download when a connection is made, and the updates to the central database with the latest data. Over 3.5 million users are registered in the database as of this typing. The total user count is not the actual number of computers running the client. A single registered user can run the client on multiple computers simultaneously. That makes SETI one of the largest software distribution projects ever!

 If you haven't come in contact with the SETI@home project, check it out at <http://setiathome.ssl.berkeley.edu/>.

Summary

Deploying to remote users is much the same as deploying using methods that we have discussed in previous chapters. About the only deployment method that does not work for remote users is Sneakernet. CD-ROM, email, Web, commercially available deployment software, and custom methods all work. The main concerns are the connection methods and the length of the connection. Make sure that your deployment strategy has a method to deal with interrupted transmissions.

The next chapter will focus on deploying software in the enterprise. The topics will include roaming users, metering, and management in both small and large enterprises.

Copyright Statement

© 2001 Realtimepublishers.com, Inc. All rights reserved. This site contains materials that have been created, developed, or commissioned by, and published with the permission of, Realtimepublishers.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 Realtimepublishers.com, Inc or its web site sponsors. In no event shall Realtimepublishers.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 Realtimepublishers.com, please contact us via e-mail at info@realtimepublishers.com.