Are your desktop hardware, operating systems, apps, and local and LAN-based data ready for the Y2K bug? This article examines a four-step process for bringing your desktops into Y2K compliance.

Your company just spent a gazillion dollars to clean up the glass house, to entice COBOL and RPG programmers out of retirement, to redesign hundreds of green screens and to convert terabytes of mainframe data. But what about the PCs that will be receiving that data or accessing client/server applications? Are your desktop hardware, operating systems, apps, and local and LAN-based data ready for the Y2K bug? Have you dodged the rattlesnake only to be menaced by a million mosquito bites?

Although desktop Y2K issues haven’t received as much attention as those in the data center, that doesn’t mean that they’re not as important. In fact, they’re worse. At least you know which applications run on your mission-critical mainframes and midrange systems, generally you know who wrote them, and you know who has the source code.

But think about your desktop and mobile PCs and servers. Think about one simple hardware question: How many makes and models do you have, and does each of them have a Y2K-compliant BIOS? Bear in mind that not all machines of the same model have the same BIOS, thanks to the manufacturer’s practice of slipstreaming improvements during the manufacturing process. And even worse, due to the fact that modern PC workstations and servers use Flash BIOS technology, even if you didn’t update the BIOS, perhaps a savvy end user did.

What about the software? Your organization certainly has a recommended load-out, but in most companies, that’s only a recommendation, perhaps backed up with a “you load it, you support it” threat. Most PCs have the recommended software at the recommended version level. Others may be downlevel due to deliberate choice, oversight or hardware limitations. Or, they might be using a newer version than what you’ve officially deployed, be using beta software, or be using “allegedly” compatible alternatives.

There may be “special-case” software not on your approved software list but which interacts with business-critical data. In some cases, your technicians may not even know such applications are installed until a machine fails to boot or an app breaks or corrupts someone’s local data, or worse, corrupts shared data files required by approved Y2K-compliant apps.

What can you do? The process of bringing your desktops into Y2K compliance may seem similar to that for the data center. But don’t let appearances deceive you; the devil is in the details. Let’s look at a four-step process, from discovery, to prioritization, to remediation, and then to deployment.

**STEP 1: DISCOVERY**

Before you can develop a Y2K plan for your desktops, and estimate the time, manpower and budget required, you’ll need to know what you’re dealing with. It’s a jungle out there! You must survey that jungle, identifying trouble spots and setting priorities.

One of the first steps should be an inventory and Y2K assessment of standard company data files, templates, documents, macros, etc. You can develop a list of critical data files that must be either replaced by new versions or migrated to a new format.

Next, you’ll need to determine what’s actually in the jungle. Some companies may wish to conduct a physical inventory by...
A much smarter approach, however, would be to use software that can conduct such surveys remotely; this not only can compare a system’s configuration against databases listing thousands of hardware and software products and versions, but has the added benefit of automating the report process. One such product to handle Y2K on the desktop, from discovery through remediation and rollout, is Attachmate’s e-Vantage Expert Assistant, but don’t worry, the approaches I will discuss aren’t specific to that product.

What else should you look for? You may need to know where company-critical files such as budgets or contact databases are stored for each user. Are they all on a file server? On a local hard drive? On removable media, such as Iomega Zip or Jaz disks? Take extra care to identify the non-Y2K-compliant files determined earlier. You might even need to interview users, especially “knowledge workers” who might have varied their system configuration from recommended norms.

A particularly messy situation occurs when the user’s equipment is located away from your main office. If you choose the physical-inventory route, laptop-toting road warriors can be sent email or login messages requiring that they notify IT next time they’ll be in the office so as to schedule an inspection or to bring their laptops to a testing area upon their return. Workers in small field offices without on-site IT support are trickier; unless their machines are already equipped for remote management, you may need to send a surveyor into the field, walk the local office manager through the inventory process, or hire a contractor. Similarly, the hardest situations are with telecommuters or contractors — many of whom may be using their personal hardware and software to perform critical company functions. Not only will this hardware and software be the most difficult to physically survey, but they will likely have the greatest variance from company standards.

Whether you conduct a physical or electronic survey, the outcome should be a comprehensive database that lists hardware, operating system and application types and versions, as well as the types, locations, and versions of non-Y2K data files. Based on this data, you’ll be able to see what’s installed and its Y2K compliance status (good, bad, or unknown). Determining the Y2K status of those “unknown” items should be a parallel task, with research being conducted as soon as a new product is identified.

For a controversial topic: what to do after the inventory? Some analysts and consultants recommended “locking down” machines after a physical inventory; you don’t want to have to go back again! Other analysts recommend that such a lock-down be imposed after all Y2K “fixes” are made. Consult with users, managers and executives while developing such a policy; in this case, grass-roots approval can make all the difference.

The process of bringing your desktops into Y2K compliance may seem similar to that for the data center. But don’t let appearances deceive you; the devil is in the details.

**STEP 2: PRIORITIZATION**

The next step is to decide what must be repaired in order to keep the business running smoothly, when those repairs must be complete, what’s required to make the repair, how long they’ll take and how much it will cost.

Not all Y2K problems are created equal. If an application tracks four-digit year information internally, but only displays the last two digits on screen, that problem might go on your “fix it later” pile. If you have a PC whose internal clock will not provide the right time in January 2000, but that can pick up the correct time from a time server, again, that’s not really a problem at all. But if you have an order-entry macro that prompts for a two-digit year and automatically appends a “19” to the front of it, well, that needs to be high on the priority list.

Here’s a small item that’s often overlooked: file-naming conventions. Many departments and individuals use dates as part of file names. A status report written on March 12, 1999 might be named “990312sr.doc.” How should such files be named next year? Should existing files be renamed to accommodate a new scheme? Must a new Y2K-compliant scheme fit within DOS’s 8.3 file-name constraints? For an individual’s private files, that problem may not be hyper-critical, but if CGI scripts on a Web server expect files to use that convention, or if spreadsheets use macros to load other data files based on that format, it’s pretty important.

Based on your analysis of those and other problems, develop a priority list, sorted by the date that the work must be completed. From there, determine the time and staffing required to perform the repairs — including programming, field work, data conversion, IT training and testing. Prepare a budget based not only on the staff and travel costs required but also for contractors and temporary labor. The biggest item may be hard costs such as buying new desktop or laptop PCs, operating system or application upgrades. If you’re upgrading user applications, don’t forget to budget time and money for retraining your IT staff to install and support them.

A tricky issue for many companies will be where to budget the money. If you’re replacing a battered 486-based laptop, is that a Y2K expense or just a normal upgrade? What does this do to depreciation schedules? Work closely with the CFO’s office on such matters.

Once you’ve developed a preliminary timeline, it is critical to get buy-in from outside the IT department, from both line-of-business managers and corporate execs. Be up front in discussing which Y2K problems will be addressed first — and which tasks will be delayed. Be sensitive to their needs and concerns, but also be realistic in quantifying the costs of accelerating the program.

**STEP 3: REMEDIATION**

Now that priorities have been set and funding approved, it’s time to determine the exact fix for each problem. With desktop hardware, you have four basic choices: ignore it, replace it, upgrade it (usually by flashing the BIOS or other firmware), or install software that patches or masks the problem.

If hardware is nearing the end of its useful service life, you may wish to take “advantage” of the Y2K situation to do an early replacement. Fortunately, with most PC hardware,
users are barely affected by remediation, other than by the inconvenience of having the equipment out of service for a short time.

Software is far trickier. Once you’ve determined that a piece of software requires some form of Y2K fix, you run into issues of cost, data compatibility and migration, customization (i.e., identifying and replicating macros, templates, in-house applications), look-and-feel, policy-based management — the works. If possible, don’t make remediation decisions in haste, particularly if there is already discussion about certain software standards. Moving to that latest version of a certain app for Y2K reasons may resolve the Year 2000 issue, but that decision may make it more difficult to consider alternative products later.

Choosing a new piece of software, whether an upgraded version of the application already in use or an entirely new product, is relatively easy. The upgrade itself is hard. It will be necessary to test the product to see what else needs to be changed. All macros, scripts and templates need to be tested, and many need to be rewritten — and that will need to be added to your deployment task list. If you’re modifying data files, then you’ll also need to touch every other app that uses that data — even if that app is Y2K compliant. Test, test, test. Different research firms and analysts give different numbers, but there’s no doubt that Y2K-remediation activity will create a number of new problems. The more you invest up front to finding those problems, the less trouble you’ll have later.

What else will you need to change? Well, any network management scripts that push or monitor applications will need to be changed, as will user policies and permissions. How about network access? Don’t forget the license or key servers. How will data be migrated and tested? How will you ensure that all users and their data are upgraded simultaneously or in such a way to guarantee that some users aren’t modifying old and obsolete datasets after the migration? (Oops, don’t forget those branch office, mobile, and telecommuting workers!) How quickly can you train users on the new system? Plan for increased help desk activity surrounding new or upgraded apps or scripts.

Once these steps have been taken to determine the best way to repair your desktop Y2K situation — and they have been fully tested — then it’s time to begin deployment.

STEP 4: DEPLOYMENT

Every situation is different. You have a lot of deployment choices to make, depending on your staffing level, physical distribution of IT resources and of your end users, and the urgency of the Y2K problems.

If upgrades will require on-site calls by IT staff or contractors or major inconvenience (and loss of productivity) to enterprise end users, you may wish to consolidate the fixes and apply them in a single visit. On the other hand, if only a few major repairs must be completed to a small number of machines — such as firmware upgrades or upgrading one small group of holdouts who are using obsolete software — perhaps those could be scheduled separately. In general, however, it would be preferable to only “touch” each machine once.

The Y2K problem outside the glass house is very different than the mainframe/midrange problem. The applications are myriad, the source code unavailable, and end user discipline often lacking (as the number of unapproved applications installed on your LANs will testify).

What about sequencing? That’s a matter of problem priority, resources, and, well, politics. Do you apply fixes department by department? Floor by floor? City by city? According to whoever uses a particular application or needs access to a particular data file? According to whoever has taken the new application training class? Again, every situation is different, so make sure that all contingencies are covered — such as determining when to upgrade people who work in multiple departments.

When it is time for that big roll-out, how are you going to accomplish it? Sending warm bodies from office to office to load new operating systems and applications, flash firmware, patch executables, install new templates and scripts, and migrate data will be painfully slow and should be avoided if possible. A better approach would be to install desktop-management software once on each machine — a “push client,” if you like — perhaps as part of the discovery process. Then, whenever you deem it appropriate, fixes can be pushed out to the desktop or mobile clients from a central server. That’s about as close to “simultaneous” as you’re going to get.

While you’re deploying fixes, you’ll need to do something to keep those machines clean. Whether you call it isolation, quarantine or lockdown, many experts have been emphasizing the importance of keeping your systems in a Y2K-compliant state — and that includes file servers. You don’t want to migrate over a spreadsheet just to have someone resave it in the old format!

From an internal PR perspective, you might want to claim that you’re isolating or quarantining “unclean” machines, but really, your emphasis should be on setting up clean zones. The stricter your policies (especially if they’re electronic policies, as in policy-based management), the unhappier many users will be. But the looser your policies, the sloppier your Y2K compliance program will be. It’s up to you, the corporate culture, and “the powers that be.” (See the accompanying sidebar, “Power to the People.”)

As part of the roll-out stage, don’t forget that things are going to go wrong. You’ll need to provide 24x7 coverage — the word “overtime” is one that budget hawks will get used to very quickly. Write lots of FAQ files for your intranet and make sure that whoever answers the help desk line is empowered to do triage.

Consider setting up a Y2K SWAT team with sufficiently broad authority (and budget) to deal with crises as they arise. Don’t allow management to Forget that although Y2K is a big issue, it’s not the only situation your department will have to deal with. Printers will still break, laptops will still be dropped or stolen, Web sites will still be hacked and hard disks will still crash, whether the clock says 1999 or 2000. Try to ensure that the Y2K issue doesn’t suck up all of your troubleshooting bandwidth.

Again, automated management might be the way to go, wherever possible; however, you might also wish to issue fresh pager batteries to your troops on December 31.

SWAT THOSE PESTS

The Y2K problem outside the glass house is very different than the mainframe/
Power to the People

The battles between IT and corporate users are legendary. Y2K is one area where, if at all possible, it would be better to avoid this traditional conflict.

As with many intra-company conflicts, communication is the real issue. Your users are worried about Y2K, worried about the loss of business, and worried about personal inconvenience. Going into their department with a heavy-handed “lock ’em down” approach is sure to fan flames of discontentment and resentment. Why not view users as helpers rather than nuisances? Recruit trusted users, both management and staff, to be part of a corporate Desktop Y2K Committee. Your best candidates will be “opinion leaders” who have the trust of their peers. Make sure they’re really involved with you, not merely showing up for a monthly brown-bag luncheon meeting.

There are many ways to use Desktop Y2K Committee members. They can help you understand business priorities and cycles that might be affected by the Y2K problem — and even if you don’t need their help, ask for it anyway. They can help you identify unique situations that may need to be handled with extra care. Ask the Desktop Y2K Committee to help you communicate with the rest of the company on everything from informational login messages to status-report posters, from a the Y2K newsletter to setting up training sessions. Let them help with evangelizing new software solutions and soften the blow when old “favorites” are slated for deinstallation.

Who knows? Perhaps it’ll start with Y2K but end with a whole new style of IT-user relations.