

Note: This document was pulled from the Windows Supersite, built and maintained by Paul Thurtott. The link to this site is www.winsupersite.com. The original web document was edited and converted to pdf format by Thomas Jerry Scott in 2003. The original document had many pictures that are not included in this document.

Windows Server 2003

Part One: The Early Years

During a recent trip to Microsoft's Redmond campus with Janet Robbins and Mike Otey, we had the chance to sit down and chat with two of the most notable figures in the history of Windows, Mark Lucovsky and David Thompson. For those of you not familiar with the early days of Windows NT, known then simply as NT, both Lucovsky and Thompson played key roles in the development of this important software project. Mark Lucovsky, Distinguished Engineer and Windows Server Architect at Microsoft, joined the company with the original wave of ex-Digital Equipment Corporation (DEC) employees that accompanied NT architect Dave Cutler. Known primarily for his unusual ability to grok how the thousands of components in NT work together, Lucovsky is widely hailed for his technical acumen and his early efforts to change NT from an OS/2-based system to one that ran 32-bit Windows applications. David Thompson, Vice President of the Windows Server Product Group, joined Microsoft in 1990 and led an advanced development group in the company's LAN Manager project before joining the NT team later that year. There, Thompson guided the development of NT's networking subsystem, ensuring that the product would work not just with Microsoft's products but with the outside world.

Here's how it all began.

The NT team arrives at Microsoft

"We came together as a group in November 1988," Lucovsky told us, noting that the first task for the NT team was to get development machines, which were [then] top-of-the-line 25 MHz 386 PCs with 110 MB hard drives and 13 MB of RAM. "They were ungodly expensive," he said, laughing. The first two weeks of development were fairly uneventful, with the NT team using Microsoft Word to create the original design documentation.

"Originally, we were targeting NT to the Intel i860 (code-named 'N-Ten)', a RISC processor that was horribly behind schedule. Because we didn't have any i860 machines in-house to test on, we used an i860 simulator. That's why we called it NT, because it worked on the 'N-Ten.'"

-Mark Lucovsky
Distinguished Engineer
Windows Server Architect

Finally, it was time to start writing some code. "We checked the first code pieces in around mid-December 1988," Lucovsky said, "and had a very basic system kind of booting on a simulator of the Intel i860 (which was codenamed "N-Ten") by January." In fact, this is where NT actually got its name, Lucovsky revealed, adding that the "new technology" moniker was added after the fact in a rare spurt of product marketing by the original NT team members. "Originally, we were targeting NT to the Intel i860, a RISC processor that was horribly behind schedule. Because we didn't have any i860 machines in-house to test on, we used an i860 simulator. That's why we called it NT, because it worked on the 'N-Ten.'"

The newly named NT team had a basic kernel mode system up and running on the simulator by April 1989. "We started with five guys from DEC and one from the 'outside' (i.e. Microsoft), a guy named Steve Wood," Lucovsky said. "And we stayed a tiny group for a long time, through the summer. We thought, 'How hard could it be to build an OS?' and scheduled 18 months to build NT. But we had forgotten about some of the important stuff--user mode, networking, and so on."

By late 1989, the NT group began growing. They added a formal networking team and expanded the security team beyond a single individual who, incidentally, had also been previously burdened by file system and localization development. "We grew that first year to 50 people or so," Lucovsky said. "And within a year, we finally had the first functioning i860 prototypes, so we could use those instead of the simulators. We started looking at context switch times, to get an idea of how well it would perform. It became obvious almost immediately that the i860 would never work out. So we started looking at the MIPS architecture, another RISC design."

In December 1989, the NT team made the decision to ditch the i860 and target the MIPS R3000 chip instead. "Within two or three months, we were booting NT on real hardware in Big Endian mode," Lucovsky told us, "and our architecture really paid off. We had designed NT to be portable, and we proved it would work almost immediately when we moved to MIPS. We made the change without a lot of pain."

By this time, the NT team started expanding rapidly, with most of its members now coming from the ranks at Microsoft. The graphics team was greatly expanded, once a new style of doing graphics was created. They also started an NT port to the Intel

i386, which was the mainstream PC processor at the time, but Lucovsky explained why it was important to the team that they didn't target the i386 initially. "We stayed away from the 386 for a while to avoid getting sucked into the architecture," he said. "We didn't want to use non-portability assumptions." If they had targeted Intel's volume chip from day one, he said, they would have had a higher performing system initially, but it would have hurt NT in the long run, and made it harder to pursue new architectures as they did recently with the 64-bit Itanium versions of Windows Server 2003.

NT becomes Windows NT

"By the spring of 1990, we had the MIPS version limping along and we started the 386 version in earnest," Lucovsky said. "It was another huge growth spurt." That May, Microsoft released Windows 3.0 and, suddenly, the world took notice. Windows was a smash success, and the obvious future of PC-based graphical computing. "We started looking at Windows 3.0 and said, 'What if, instead of OS/2, we did a 32-bit version of Windows?'" Lucovsky noted, casually throwing out the question on which the next decade of computing hinged. "Four guys--Steve Wood, Scott Ludwig, a guy from the graphics engine group, and myself--looked at the 16-bit Windows APIs and figured out what it would take to stretch them to 32-bit. We spent a month and a half prepping the API set, and then presented it to a 100-person design preview group to see what they thought."

The key characteristic of the new API, eventually named Win32, is that, though it was a new API, it looked and acted just like the 16-bit Windows APIs, letting developers easily move to the new system and port their applications. "We made it possible to move 16-bit applications to NT very easily," Lucovsky said, "and these applications could take advantage of the unique features of NT, such as the larger address space. We also added new APIs that weren't in the 16-bit version. We added major new functionality to complete the API, making it a complete OS API, but we did this using a style that would be familiar to the emerging body of Windows programmers."

The reaction within Microsoft was immediate. "They loved it," he said, "when they saw how easy it would be. It was basically Windows on steroids, and not OS/2, which used a completely different programming model." Making NT a 32-bit Windows version instead of an OS/2 product, however, introduced new issues, not all of which were technical. Microsoft had to get ISV and OEM approval, and of course alert IBM to the change. "We did an ISV preview with IBM, and had this deck of about 20 slides, and we said, 'look, this is what we're going to do.' At first, they thought Win32 was a fancy name for OS/2. Then you could just see it on their faces: 'Wait a second, this isn't OS/2.'"

The decision to drop OS/2 for Windows forever damaged the relationship between the two companies. "But we had executive approval, and started the port," Lucovsky said. "So instead of working on an OS/2 subsystem for NT, we picked up Win32." At that moment, he said, the product became Windows NT.

NT's modular architecture paid off during this change as well. "Thanks to our micro-kernel architecture, with the kernel decoupled from application environments

"Our core architecture is so solid, that we were able to take NT from 386-25's in 1990 to today's embedded devices, 64-way, 64-bit multiprocessor machines, and \$1000 scale-out server blades."

-David Thompson
Vice President
Windows Server Product Group

like POSIX and Win32, we didn't have to change the kernel or start a new programming effort," Lucovsky told us. "The deep guts of the scheduler didn't have to change. We had C command line applications up and running within two weeks. This was September 1990."

Thompson elaborated on the importance of NT's foundations. "Our core architecture is so solid, that we were able to take NT from 386-25's in 1990 to today's embedded devices, 64-way, 64-bit multiprocessor machines, and \$1000 scale-out server blades. We've been able to deliver a whole array of services on it."

September 1990, truly, was the turning point for Windows NT. Not coincidentally, that's also when Dave Thompson, previously heading Microsoft's LANMAN for OS/2 3.1 advanced development team, joined the NT team. "We threw the switch," Thompson told us, "and the team went from 28 to about 300 people. We had our first real product plan."

RTM and beyond

The first version of Windows NT, Windows NT 3.1, was released in July 1993 and named to match the version number of the then-current 16-bit Windows product. That NT version featured desktop and server editions and distributed security in the form of domains. Since then, the NT team has worked on a progression of releases, all developed on the same underlying code base.

The next release, Windows NT 3.5, was code-named Daytona, and shipped in September 1994. "Daytona was a very rewarding project," Thompson said. "We focused on size and performance issues, and on "finishing" many of the first-release features of 3.1. Daytona also had significant functional improvements and enhancements." The original themes for Daytona were size, performance, compression, and Netware compatibility. Two of those goals were emblematic of the time: DoubleSpace-style compression was a hot topic in the early 1990's because disk space was at such a premium, and Netware was the dominant network operating system of the day. "We eventually dropped compression," Thompson said, "but the Netware port was strategic. Novell was ambivalent about the NT desktop – they didn't know if they wanted to build a client. We offered our assistance, but they kept messing around and ... well. We did our own. And it just blew them away. Ours was the better Netware client, and customers used ours for years, even after they

finally did one. That client enabled the NT desktop, because Netware was the prevalent server in the market. We wouldn't have been able to sell NT desktops otherwise."

Daytona also benefited from new compiler technology which enabled Microsoft to compress the code size and enable realistic NT desktops

NT Timeline

October 31, 1988: David Cutler arrives at Microsoft
November 1988: Work begins on NT project
July 27, 1993: Windows NT 3.1 ships
September 21, 1994: Windows NT 3.5 ships
May 30, 1995: Windows NT 3.51 ships
July 31, 1996: Windows NT 4.0 ships
February 17, 2000: Windows 2000 ships
October 25, 2001: Windows XP ships
April 24, 2003: Windows Server 2003 ships

on lower-end systems than the original version. "The results were measurable," Thompson said.

Windows NT 3.51 was dubbed the Power PC release, because it was designed around the Power PC version of NT, which was originally supposed to ship in version 3.5. But IBM constantly delayed the Power PC chips, necessitating a separate NT release. "NT 3.51 was a very unrewarding release," Thompson said, contrasting it with Daytona. "After Daytona was completed, we basically sat around for 9 months fixing bugs while we waited for IBM to finish the Power PC hardware. But because of this, NT 3.51 was a solid release, and our customers loved it." NT 3.51 eventually shipped in May 1995.

Fittingly, the next NT release, Windows NT 4.0, became known as the Shell Update Release (SUR), another challenging task that would once again prove the benefits of NT's module architecture. "We wanted to build a desktop that had the 95 shell but used NT technology," Lucovsky told us. "We eventually moved the Win32 GUI components and hosted them as an in-process driver. Performance was one side effect. We had had problems taking that API and running it in a different process. So moving the code to the same context as the runtime solved a lot of issues. We didn't have to do dead lock detection for GDI and USER. It was significant work, but it solved a lot of headaches." NT 4.0, a watershed release for the product, shipped in July 1996.

Windows everywhere

With the next release, Windows NT would lose the NT name and become, simply, Windows. Thompson says the decision came from the marketing team. "A guy from the Windows [9x] marketing team moved over to NT marketing and said we should use Windows everywhere. We were all uncomfortable with the name change at first, because NT had a solid reputation. But because of the reliability push with Windows 2000, people started talking about how much better Windows 2000 was than 'that old NT stuff,' even though it was the same architecture. So it was actually kind of

fortuitous how it happened." Incidentally, Windows 2000 didn't have a codename "because Jim Allchin didn't like codenames," Thompson says.

Since the completion of Windows 2000, the biggest decision the Windows team made was to split the client and server releases with the Whistler products, which became Windows XP and Windows Server 2003. "This lets us focus on the server customers, who want it rock solid, rather than right now," Thompson told us. "Desktop software has to ship in sync with [PC maker] sales cycles. There is no holiday rush with servers."

Windows Server 2003 Part Two: Developing Windows

One element about the NT family of operating systems--which evolved from Windows NT to Windows 2000, XP, and, now, Windows Server 2003--that has remained unchanged over the years, though the details have changed dramatically, is the build process. Somewhere deep in the bowels of Microsoft, virtually every day, at least one Windows product is compiled, or built, into executable code that can be tested internally by the dev, or development teams. For Windows Server 2003, this process is consummated in Building 26 on Microsoft's sprawling Redmond campus, where banks of PCs and CD duplicating machines churn almost constantly under the watchful

eyes of several engineers.

The details of NT--excuse me, Windows--development have changed dramatically since the project first started in the late 1980's.

"There are 5000 developers on the Windows team generating over 50 million lines of code for Windows Server 2003. It's an enormous task, the biggest software engineering task ever attempted. There are no other software projects like this."

-Mark Lucovsky
Distinguished Engineer
Windows Server Architect

"Back in the early days, we started with 6 people," Microsoft Distinguished Engineer and Windows Server Architect Mark Lucovsky told me. "Now there are 5000 member of the Windows team, plus an additional 5000 contributing partners, generating over 50 million lines of code for Windows Server 2003. Getting all those people going in the same direction, cranking out code, is an enormous task. Building the results of their work, compiling and linking it into the executable and other components that make up a Windows CD is a 12 to 13 hour process that is done every day of the week. It's the biggest software engineering task ever attempted. There are no other software projects like this." And Microsoft compiles the whole thing--all 50+ million

lines of code, almost every single day, he said. "We're evolving the development environment all the time," Lucovsky noted.

"When we turn the crank, we compile the whole thing," he said. "We have to be able to reproduce the system at any point in time as well. So developers check in code, we press a button, and out comes a system. We should be able to reproduce that [build] three years in the future, using the various tools, compilers, and scripts we used at that time."

David Thompson, corporate vice president of the Windows Server Product Group at Microsoft, elaborated on the process. "The key here is that we built up the system over the years, advancing it in three dimensions," he said. "First is the product itself. Second is the way we engineer the product. And third is the way we interact with a broader and broader set of customers. The product evolution is pretty straightforward. The source code control system we use now is new, because we really pushed the scale of the previous version with Windows 2000. Mark [Lucovsky] personally lead the development of the new system and introduced it post-2000. We started with some acquired technology. We now do have a staged build [for the first time]. But every day the [staged builds] are rolled up into the total build. So we can scale but maintain stability--we know where we stand every day."

Just eat it: Microsoft serves up dog food

Lucovsky reminisced a bit about the early days, when the first NT prototypes were built in his office with only a single person overseeing the process. That person would simply send out an email to the NT team when a new build was ready, and then 50 people or so would "eat their own dog food," testing the build on their own systems and run stress tests. "I used to just walk around the building and write down the problems we found," Lucovsky said. "That's how it was pre-NT 3.51. Now we have 7 builds labs. Dave [Thompson] has his own [build lab] for the 1200 people he oversees. The main build lab cranks out the official build, which goes out to thousands of people daily. Notification is automatic, and is sent out in multiple stages using the backbone servers across the campus. It's all automated. Those little things have now scaled up."

"Originally, we had a certain time of day [up to which time] we could check code in and then we stopped," Thompson said. "After that, we threw the switch and built the new system. Eventually, we grew the team to 85 people and serialized the process for more control. [NT architect] Dave Cutler--who we all worked for---ran the build lab for about a week, and he required people to personally write their check-in requests on a whiteboard in the lab. He forced it into a mold. I sat in there for a while too. One day I accepted 85 check-ins, the most we had ever had to that point. Now we can take in over 1000 every day. It's a completely different scale. Even the whiteboard is electronic--Web based, actually--now."

"There are no other software projects like this," Lucovsky said, "but the one thing that's remained constant [over the years] is how long it takes to build [Windows]."

"One day I accepted 85 [code] check-ins [on NT], the most we had ever had to that point. Now we take in over 1000 every day. It's a completely different scale. Even the whiteboard is electronic--Web based, actually--now."

- David Thompson
Vice President
Windows Server Product Group

No matter which generation of the product, it takes 12 hours to compile and link the system." Even with the increase in processing horsepower over the years, Windows has grown to match, and the development process has become far more sophisticated, so that Microsoft does more code analysis as part of the daily build. "The CPUs in the build lab are pegged constantly for 12 hours," he said. "We've adapted the process since Windows 2000. Now, we decompose the source [code] tree into independent source trees, and use a new build environment. It's a multi-machine environment that lets us turn the crank faster. But because of all the new code analysis, it still takes 12 hours."

Dogfooding their code has always been a key requirement of the NT team, Thompson told me, and an integral component of Microsoft's culture. "This is one of the things we've always done, back to the earliest days," he said. "We were just joking about this today, actually, talking about our email program. Back when we first got NT running on desktop [PCs], our email program wouldn't run because it was a DOS application, and we didn't have DOS compatibility mode working yet. So I ported our internal email app, WizMail, to Win32 so we would be able to use only NT systems."

"When you are forced to use the system yourself, you see bugs and you see the performance issues," Thompson added. "And you'd go and find the person responsible for the problem and ask them to fix it." One of Thompson's primary responsibilities when he joined the NT team was to deliver the file server over to NT so that it could be used as the source code server. That required a moment of faith, especially since NT was then using a prototype version of the NTFS file system. "The networking group took this very seriously," he said, "and made sure it was ready for internal deployment. Once it was rolled out, we never backed away. Obviously, if the file server goes down, it's a disaster. So it was a big moment for us, getting over that hump."

Later, as the development of Windows NT 4.0 wound down, Thompson's team took on Active Directory (AD), Microsoft's first directory service, which debuted publicly at

the Professional Developers Conference (PDC) in 1996. "Before AD we had NT domains for our infrastructure," he said, "and going to AD was even more complex. We deployed AD very early, first with our team, and then the wider Windows group. Then we threw the switch on Redmond [campus] AD in April 1999."

Microsoft rolled out AD to the rest of the company in stages, Thompson said, using careful planning. The campus went to a multi-forest AD topology with Windows Server 2003 last year. "With all of the infrastructure servers, we always do a complete deployment internally, then push it out to the JDP (Joint Development Partners), who test and deploy it in production in over 250 usage scenarios. We get bug reports, feature feedback, and complex scenario testing that really proves the product."

Windows Server 2003 hit 99.995 percent availability at the Release Candidate 1 (RC1) stage last summer, and the Microsoft.com Web site was fully deployed on WinServer 2K3 when RC2 rolled out in November 2002. "Heavy usage internally and by close customers is key," Thompson told me, "and we have a more mature view of what the product is now [compared to the early days]. We're not just shipping bits in a box, but are also shipping a wide range of complementary tools, products, services, and documentation." And Thompson explained that the teams working on Outlook 11, Exchange Server 2003 ("Titanium") and Windows Server 2003 are all working much more closely together to implement complete end-to-end scenarios that meet customer needs. In the past, these products were often developed more independently.

Are you being served? A look at product maintenance

"Servicing has definitely matured over the years," Lucovsky added. "We do a lot of work figuring out the right mix of service packs, hot-fixes, [product] development branches, betas, and JDP customers for each product." (More information about development branches can be found in the next section.)

"We've really extended the time that we service our products," Thompson said, because when Microsoft ships a server product, customers may use it for up to ten years. So-called volume, or mainstream, service lasts seven years, but the company has constantly evolved the way it supplies updates and fixes over time. First, Microsoft has to be sure that bug fixes are applied to all of the applicable development branches. "Our work in rapidly addressing security vulnerabilities means that we now aggressively issue hot-fixes when we can," Thompson noted. "As well, it used to be that [service packs] were flexible, a way that we could deliver features as well as fixes. But customers made it clear that they wanted bug fixes only [in service packs]. That leads to an interesting question, though: What, exactly, is a bug? Is a missing feature a bug? Customers often have different views themselves. But [Windows] NT 4 SP3 was the end [of major new features in services packs]."

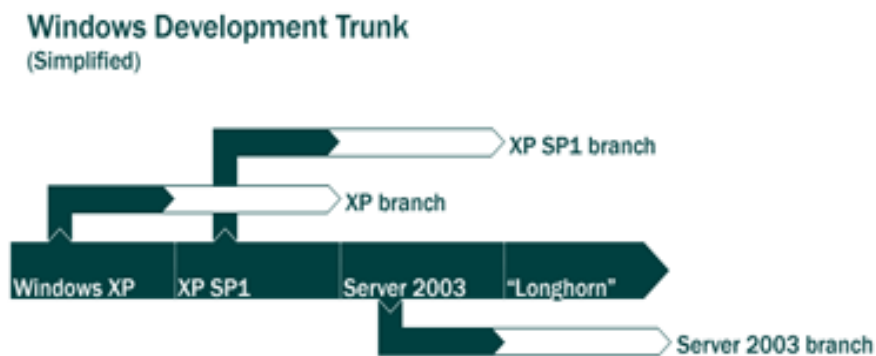
One side effect of trunk servicing is that Microsoft must maintain test environments for every permutation of its recent operating systems. That means that the final, or "gold" release of Windows 2000 is one branch, Windows 2000 SP1 is another, Windows 2000 SP2 is another, and so on. "And dogfooding is important to providing service packs, too. In our IT organization, we maintain a [separate] Windows 2000

infrastructure just so we can do live rollouts to Windows 2000 systems and test them in a production situation," Thompson said. "It's a big expense, but worth it."

Hot-fixes are treated as narrow releases that should fix only one specific problem and not affect other parts of the system. Thompson said that customers should generally only apply a hot-fix if they're affected by the problem the fix addresses. However, security fixes are another issue altogether. "We expect all of our customers to install the security fixes," he said, "so we are very careful with them, and do the right kind of testing. They are Generally Deployable Releases (GDRs), just like service packs."

Trunks, trees and branches

As noted earlier, the various Windows versions require a series of product development code forks, where each different Windows product "branches" off the main development "trunk" over time. So each Windows release builds off the last, and at least two different versions--Windows Server 2003 and Longhorn, at the time of this writing--are in simultaneously development. Because WinServer 2K3 was split from XP, the server product basically builds on XP. Longhorn, a client release that will succeed XP in a few years, is actually building off the server branch code base, and not XP as you might expect.



"The mechanics of doing this are mind-numbing," Lucovsky told me. "We have a main branch of code for the current Windows version, and that branch becomes the source base for hot-fixes and the next service pack. Once we spit out a service pack, that becomes a branch and now we have two branches we have to test for hot-fixes and service packs. We can't tell customers to install, say, SP1 and then do this hot-fix. And this is going on for every [Windows] release, so some have 2 or 3 service packs, many hot-fixes, and many security fixes. Every one of these is a managed collection of 50 million lines of code. It's a pretty big accounting issue."

Additionally, for each main branch in active development, Microsoft also has roughly 16 team level branches to allow team level independence/parallelism while working on a common main line branch. Each team maintains a complete build lab environment that builds an entire release including the team's latest changes and periodically integrates their tested changes back into the associated main branch so that others can see their tested work.

Going to War:

Triaging Bugs in the War Room

During the mad dash towards RTM, the heartbeat of the project is the War Room, where the War Team meets two to three times daily, five days a week--six days a week now that Windows Server is in its final days of development. "The War Team goes over reports and metrics to see where the project is at every day," Thompson told us, an understated explanation that did little to prepare us for the horrors of the War Room. "Everything is automated now, but back then we came in and passed around paper reports that showed us how we were doing. There were, maybe, 15 to 20 people in the room. Now it's very different."

It sure is.

For Windows Server 2003, the War Room is run by Todd Wanke, who we eventually found to be an amazingly likeable guy. However, in the hour-long War Room sessions, Wanke rules with an iron fist, asking trusted lieutenants for advice here and there, but moving the process inexorably forward with little patience for excuses or, God forbid, product team members who don't show up for the meeting .

Here's how it works. Every morning at 9:30 a.m., representatives from various Windows Server 2003 feature teams meet to triage bugs. They file into conference room 3243--whose exterior sign has been covered up by a handwritten note that reads "argument clinic"--in building 26. There's a large conference table in the center of the room, but most of the participants have to stand, and the room is always overflowing with people. On the day we attended a War Team meeting--the first time any outsiders were allowed to view the inner sanctum for Windows Server, and only the second time overall during the entire development of NT and Windows--the team progressed through about 50 bugs, most of which were simple branding problems, though I've agreed not to discuss the specifics of any bugs discussed that day. (Because we attended War Room very late in the development of the product, and the biggest outstanding issue was the last minute name-change from Windows .NET Server 2003 to Windows Server 2003.)

Every bug is logged in an incredible bug tracking system, each accompanied by a dizzying array of information about how the bug was found, which customers, if any, were affected, and a complete history of the efforts made to date to eradicate the problem. Wanke moved quickly through the bugs, calling out to members of specific feature teams to explain how the fixes were progressing. If there are one or more bugs in IIS, for example, a representative of the IIS team needs to be present to not only explain the merits of the bug, but whether customers are affected, how the fix might affect other parts of the system, and how soon it will be fixed. This late in the development process, bugs are often passed along, or "punted," to the next Windows release--Longhorn--if they're not sufficiently problematic.

The atmosphere in War Room is intimidating, and I spent most of my time in the room, silent and almost cowering, praying that Wanke wouldn't turn his attention to me or my group. Heated argument and cursing are a given in War Room, and the penalty for not being on top of your bugs is swift and cruel ridicule from the other team members. The most virulent treatment, naturally, is saved for those foolish enough to blow off a War Room meeting. On the day I attended, one feature group

had four of its bugs punted to Longhorn because they had failed to show up for War Room. When someone argued that they should be given another day, Wanke simply said, "F#\$% 'em. If it was that important, they would have been here. It's in Longhorn.

Next bug."

Once the hour long meeting was over, we sat down and spoke with Wanke, who was almost a

"If people aren't [in War Room], I lay into them. I'm the ass kicker."

- Todd Wanke
Product Manager
Windows Server Release Management

completely different person in private. "You run a mean meeting, Todd," I told him, as we sat down. Wanke's background includes stints with NCR, America Honda and an unspecified and mysterious sounding security-related assignment as a US government contractor, and he's been with Microsoft for nearly eight years. Before joining the Windows team, Wanke was one of the original architects of the Microsoft.com Web site and he spent three or four years as an "Internet guy" at the company before all of Microsoft found the Internet religion. In our meeting, Wanke explained how he fell into his new job, what he does now at Microsoft, and how the War Team works.

"My job is to manage the day-to-day operations with regards to shipping Windows," he said. "I'm responsible for 8000 to 10,000 developers, program managers, and testers, and I have to make sure they're doing the right things every day."

War Team, he said, consists of a very broad set of people within the Windows team, all of whom are responsible for different areas of the project. They are test leads with responsibility for such things as TCP-IP and other low-level technologies, some developers, people that do the build every day, people that do build verification tests, and others. "Every area of the project is represented," he told us. "The daily marching orders [for the Windows Server team] come from War Team, and also from the broad mails I send out. These emails are almost always Microsoft confidential, or even higher than that, emails that are very confidential and sent only to a much smaller group of people."

As we witnessed, War Room is a very structured event, occurring at the same time every day and lasting exactly one hour. The team members look at the same bug system every day, and often go over the same bugs until they are fixed. "If you're not there, it's not good," he said. "Microsoft people have a strong sense of ownership for the product and they want to make sure the right thing is happening. But if people aren't there, I lay into them. I'm the ass kicker."

In addition to the morning War Room meeting, the Windows Server team holds an afternoon meeting from 2 to 3 p.m. and, if needed, another one from 5 to 6 p.m. The daily build usually starts at 4:30, but can be delayed to 6, so this last meeting gives the team a chance to go over any final bug fixes that will be added to that

day's build. "The structure is very important," he said, "and we need to know where the build is at all times. We look at the quality of the build, various stress levels, and all of the things that run overnight, anything that we need to follow-up on. We get detailed reports, and review everything that goes into the project."

In addition to the main War Team, each of the feature teams have their own War Rooms, so there could be as many as 50 such meetings each day, each going over a specific component of the system. These other War Room meetings occur at 8 a.m., every day. When a bug fix passes the local War Team process, it's introduced at Wanke's meeting. "They can't come into War Room unless they're fix-ready," Wanke said. "They must be fix-ready." Because there isn't a single person making decisions, there is a system of checks and balances through which each bug fix passes before it's introduced into the build.

The complexities of building Windows are staggering. "To simplify things, let's say Windows consists of 100,000 files," he said. "Usually, there are seven source code depots, each containing an exact replica of all of the sources, though at this point, we're down to just one. Every development group has its own depot, so that when a developer writes a fix, he can compile it into the depot for testing. If the build compiles locally with his fix, they can test it there and then check it into the main depot in the main build lab."

Not every build is successful, of course. Occasionally, Windows Server suffers from what Microsoft calls "build on the floor," when a fix breaks some other part of the system, rendering the build unusable. "That's brutal," Wanke told us. "There was a point about a year ago, when we didn't get a build out for seven days. We had to send an email to the product group executives at the company explaining the problem," and the company entered into its private version of Defcon-5. "All the red flags went up," he said. "It's very ingrained in the developers not to break the build. They do their fix, do a buddy build, and then check it in. But they can't go home. We've sent out calls at 3 a.m. when the build is broken, find the developer that broke it, and get him into work right then and fix it immediately. The developers are on call 24 hours a day. There's definitely an escalation process. A broken build is considered a critical, severity-1 problem."

As the Windows Server 2003 development cycle wound down, the bug count fell dramatically, and the process was getting simpler each day. And then Microsoft announced the name change. "We just have to live with that poor decision," he told us. "They should have made it six months ago. Back then, we all agreed it was the right thing to do. But at this late stage--they brought in [CEO] Steve Ballmer to talk with all the War Teamers about why we made the change." The speed at which the team was able to fix all of the branding graphics, text, and registry entries in the system is a testament to the company's dynamic process for fixing bugs, Wanke said. The problem was that several thousand changes needed to be made, and that would normally require several thousand new entries in the product's bug tracking system. "I went out and handpicked the three best developers on the team and said, 'just go and fix it.' One developer fixed over 7,000 references to [Windows] .NET Server. Let's just say that there are people I trust, and people I don't trust. I told these guys, 'don't tell me what you're doing. Just do it.'"

Entering the home stretch

On the day that we attended War Room, on January 21, 2003, Windows Server 2003 had hit an "absolute historic low" for bugs, according to Wanke. "We're shutting down the project this week," he said. "It's done. We're going to ship it." On that day, WinServer 2K3 had just a few active bugs, and at least a quarter to one-third of those bugs were simple branding issues. "So let's say there are about 150 outstanding issues to address," Wanke told us. "Of that, we'll fix about 100. All of the bugs are severity rated from 1 to 3, plus they get a priority rating. We have [a few] severity-1 bugs left to fix, and those all have to be fixed for us to ship."

Wanke said that the server team had already fixed all of the known security vulnerabilities. "We're very happy about security," he said. "It's fun to see where we are [with security]. I'm personally very impressed with the work that went into it, the fixes and the thought process. We all think it's very secure. The [Trustworthy Computing] security push [last year] was a big milestone for us, and everything will be easier going forward because of it. It's easier on the developers because they all have the same mindset and goals now, the same education about best practices. There used to be different methodologies between different groups. The security pushed unified it. Now it's easier for everyone to communicate and see the end goal."

With the completion of Windows Server 2003 development, the development team will enter a transitional period. First, the product will enter escrow, and the build process will be frozen. That build is then deployed around the campus, including Microsoft's corporate infrastructure. "That is the final build," Wanke noted. "Then we sit on it for a period of time, during which there are no core fixes made to the product." The escrow build will also be handed out to testers and JDP members, he said.

If any issues do arise during the escrow period, the War Team makes case-by-case decisions about whether to fix the bugs. If a bug necessitates a kernel fix, a new build will be created, and escrow is reset. "A change to a core component could delay RTM," Wanke told us. "We run it prior to asking customers to, and have to run it a number of days before signing off on it. It's a long haul." Every feature team working on Windows Server 2003 must run the escrow build for 21 days without restarting before the build can be declared golden master and released to manufacturing.

But Wanke isn't worried about the exact schedule, as the outcome is finally a foregone conclusion after years of work. His team is now preparing its RTM party-- outside on one of the campus' many soccer fields, weather permitting; inside a garage if not--and Wanke has other RTM-related concerns he must address, including the launch venue. "I'm working with the launch team to book a venue," he said. "They need 95 percent confidence dates." They're also talking to OEMs to ensure systems are ready for launch, ISVs, marketing folks for signs and posters, and so on. "And I have to make sure that the 8000 people who deserve a ship award get one," he added.

In the end, all this dedication will result in the most secure and reliable operating system Microsoft has ever created, and it's impossible to overstate Wanke's contribution to this project. "I basically haven't missed a single War Team in a year and a half, give or take a day or so for personal reasons," he said, "every day, six

days a week at the end of the schedule. We let people bring their kids in on Saturdays, it's a family day. There's no swearing allowed on Saturdays. But you still have to be there, and we still have to make a build."

So would Wanke run War Team on a future Windows version?

"No way," he said, laughing. "No way."

Windows Server 2003 Part Three: Testing Windows

As the development of Windows 2000 wound down over three years ago, Microsoft was making a transition of another kind: The company's development focus was moving from delivering technology to delivering solutions that met real customer needs. It sounds like an obvious strategy, but consider the ramifications: In the past, Microsoft would determine what features to include in each revision of its products, deliver as many of those features as it could in the time allotted, and then move any dropped features into the next version. Often, the company would tout customer feedback as one of the prime inspirations for the features that were included in each product version, but customer feedback was just one of many criteria that the company considered, and it certainly wasn't the primary one.

One must consider the competitive landscape of the past decade and the markets Microsoft was targeting to understand why this was the case. For much of its history, Microsoft was the fiery upstart, scrambling for market share and paranoid that the Next Big Thing would come along, leaving the company behind. For many of its core markets in the 1990's--desktop operating systems, office productivity software, and workgroup-based corporate computing--Microsoft was primarily concerned with low costs, simplicity, and features, as it sought to outdo competitors such as IBM, Lotus, Novell, WordPerfect Corporation, Borland, Apple, and others. Customer research basically amounted to "more is better": The thought was that most users would compare bulleted lists of features and pick the software that best met their needs. And because Microsoft was often able to undercut the prices of the competition, the choice was usually obvious.

Flash-forward a decade and the computing industry has changed dramatically. Having secured its domination of the desktop operating system and office productivity markets, Microsoft had to make two key changes. First, it had to think up ways to drive sales in those markets where its mature products already dominated. Second, the company had to enter new markets to drive growth. Some of these new markets include handheld computing and other non-PC devices, video games, consumer electronics products, and, of course, the high-end enterprise computing market. Windows NT 4.0, released in 1996, saw some scalability advances, but it wasn't until Windows 2000 that Microsoft made a credible product for the markets that were traditionally served by high-end UNIX boxes. And with Windows Server 2003, the company has finally fulfilled its long-term goal of supplying products that meet the needs of virtually any type of corporation, large or small.

However, the high-end enterprise market runs on a different set of rules than the desktop market or even that of small- and medium-sized businesses, both of which often demand cutting-edge features as soon as possible. In the more mature, slower-moving enterprise market, stability and reliability are valued far above any other criteria, and these operations often employ the same software for tens of years, not two or three years. It was a market with which Microsoft had little experience through the late 1990's. And it caused the company to make dramatic changes across the board. For example, Microsoft has twice extended the support lifecycle of Windows NT 4.0, simply because its customers have been using this technology far longer than the company had originally anticipated. For Windows 2000 and, more obviously, Windows Server 2003, Microsoft understands that customers will likely be running this software for several years to come. Its customer will expect this software to be easy to deploy, administer, and update, of course, but they also expect it to just work, and to just work forever.

Developing simple software that just works is difficult enough, but when it comes to software as complex as Windows Server 2003, with over 50 million lines of code, several different product editions, and an unknowable number of future updates, security fixes, hot-fixes, and service packs, traditional software testing methodology isn't enough. Microsoft realized early on that it could not release this product into the wild, wait for the inevitable problems to arise, and then fix them later, with many customers--as usual--waiting for the Service Pack 1 (SP1) release to start deployment plans. No, Windows Server 2003 would be different, with the Release Candidate 2 (RC2) build considered "final" and the Release To Manufacturing (RTM) or "Gold" release of the product, due April 24, 2003, the equivalent of an SP1 release under the previous development methodologies. Customers should feel confident in rolling out the release candidate, knowing that that the RTM version would be even better, and follow just months later.

Windows Server 2003 would need to be tested in real world situations, and deployed live in the field, and in far larger numbers than in the past, before the final version was made publicly available. But this requires a certain level of trust between Microsoft and its customers, and, for the software giant, a software testing infrastructure that goes far beyond the standard stress tests it employed in the past. And one of the key components of this new testing infrastructure and, indeed, the new Microsoft, is the company's Enterprise Engineering Center (EEC).

Real-world testing in the Microsoft EEC

Itself the result of two years of testing, the Microsoft Enterprise Engineering Center (EEC) opened its doors in April 2002, during the Windows Server 2003 beta. The EEC is the brain-child of George Santino, the Product Unit Manager of Windows Integration Scenario Tests at Microsoft, who wanted to establish better criteria for long term product testing. It's located at Microsoft's Redmond campus, and was designed to let the company's customers duplicate their specific environments in a lab setting and see how various Microsoft software upgrades, migrations, and deployments perform using those company's real-world data and systems. The EEC is made available at no cost to customers.

"We built this facility to allow us to engage with customers while products are still under development,"

"We need to be deployable when products go out to the door, not after SP1, not after QFEs."

-George Santino
Product Unit Manager, Windows
Integration Scenario Tests

Santino told me, as we walked by a massive bank of servers at the EEC. "We wanted to test products directly with the customers that would deploy them, and understand their network topologies and the issues they're trying to solve. This helps Microsoft make products that are more enterprise-ready for these customers when they ship, and it allows us to engage customers directly." Microsoft benefits from the EEC by ensuring that its products work in the real-world scenarios that, in the past, weren't fully tested until those products were actually released. And customers benefit from one-on-one interaction with the product teams responsible for the products in question.

In many cases, a positive experience at the EEC will result in customers deploying Microsoft solutions in pre-release form. "Customers like jetBlue went off and immediately deployed Windows Server 2003 and Titanium [Exchange 2003] Beta 2 live, in production, after visiting the EEC," Santino told me. "They came here, met with people, tested their environment, and saw that it would work, and be more scalable. We were able to fix issues for them before they left."

In the past, many of these customers might have previously waited until the final product release or, more typically, until the first service pack. "We need to be deployable when products go out to the door," Santino said, "not after SP1, not after QFEs."

The first round of EEC testing involved almost 50 customers, all of whom tested Windows Server 2003 upgrades, migrations, and deployments. Customers typically spend two weeks at the EEC--though the Australian tax office was in for almost six weeks--and Microsoft can host up to five customers at the EEC simultaneously. More important, several problems and 650 design changes were made to Windows Server 2003, thanks to input from EEC attendees. And there have already been a number of repeat customers, such as Intel, Siemens, Chevron-Texaco, and Continental Airlines.



EEC control center montage

Bring on the customers

Physically, the EEC currently includes three enterprise customer labs (a fourth is now being built and a fifth is planned) and a massive server room called the control center that features just about every kind of hardware imaginable. The three labs are filled with IBM, Hewlett-Packard (HP), and Dell equipment, respectively, and are used to host end-user tests. "We let customers come and test their actual environments," Santino told me. "We configure the test environment exactly the same way the customer configures their real-world environment. It uses the same software they use, the exact same hardware, everything. It's their total environment. This helps us find interesting bugs, and we can get developers to come down to the environment, and learn more about what our customers are doing." Once an environment is set up and running in the EEC, Microsoft can perform additional tests, such as migrating from Exchange 5.5 to Exchange 2003, or whatever, build it out, and see what breaks.

One obvious question involves how customers can take part in the EEC. Santino told me that customers come to the EEC from a variety of avenues. "Our account teams in field bring in customers, as do MCS [Microsoft Consulting Service] people, and a number of customers come because they are in the JDP [Joint Deployment Program]. We also have some that come from our hardware partners and other OEMs. Once we get pinged, we love to get 3-4 weeks notice to engage and get information about how to build out the environment, and get all the software installed. But we've done it with just a couple of days notice. We can scurry around and get it done in a few weeks if we have to."

Testing at the EEC isn't confined to a single Microsoft product running against a customer's environment. The company asks customers ahead of time which software products they'd like to test, including their own proprietary solutions, and competitor's software and hardware. "We've got just about every kind of server imaginable in here," Santino says, laughing. "If it's in their environment, it needs to work. Another thing that amazes customers is that they assume we won't test that [non-Microsoft] stuff. But that's not true. If you're using [IBM] DB/2, fine. We will test what they are using in their business."

Also, customers are asked who they want to meet with at Microsoft. "We also have a program manager who essentially lives with them while they're here," Santino said.

"We find different bugs because we're working with products that are still under development. But we can get the developer working on that code right here, have him find out what the customer is doing and why. And then he can go back and get a fix before the customer has left the EEC."

While the customers are at the EEC, Microsoft gathers information and writes a business model for each customer, noting how they make money, but also what their issues are, and where their problems lie. "We ask, 'What do you do with SQL Server? Exchange? Tell us more,'" Santino notes. In many cases, the answers are surprising, and Microsoft is discovering that its customers use its products in ways it never imagined.

After the customer is gone

After the customer has left, Microsoft keeps an image of their environment, minus any real-world data. "We keep it around, send out an email to various product teams and tell them, 'we have this environment, how would you like to come and test on it?' The test teams used to just test generic environments, but now they can see how their products react to real-world situations. It's an amazing opportunity." Over time, when a product team is working on various products, they can go through the environment archive and pick the environments that they'd like to test on. "We can build it back up in a couple of days and let them test on it," Santino says.

"Before the EEC, we were limited in how we could test larger engagements, where more products are involved," Santino told me. "It's amazing how these things interact. Most environments we've tested are heterogeneous. For example, Chevron-Texaco was interesting, because they merged two totally different environments. But Barnes & Nobles was almost pure Microsoft, and there were no conflicts."

Customer reaction to the EEC

To help customers see how the EEC works, Microsoft holds regular tours of the facility. "Customers come over from the Executive Briefing Center [on campus] and say, 'we're hearing about how Microsoft listens to customers. Prove it to me.' Well, this is where it's happening. This is where Microsoft works with customers. They can come here, take the tour, see the hardware, and the actual customers doing testing. Ultimately, the EEC helps make products enterprise ready, sure, but it also makes them enterprise credible."

So what's the reaction been like? "Customer s say it's been beneficial," Santino said. "They

"The EEC helps make products enterprise ready, sure, but it also makes them enterprise credible."

-George Santino
Product Unit Manager, Windows
Integration Scenario Tests

say, 'You guys really do care, you're fixing problems, and I can see it first hand.'



They are telling us what they think, and seeing the changes based on their feedback happen in real time. Best of all, there is no cost involved. We're not marketing or sales. It's about shipping higher quality products that the customer can actually use. Ultimately, the EEC will generate sales, sure. But we want to get them in there--at no cost--and we want them to help us understand how they use our stuff. We want real data to stress the products, not a simulation. It has to be real. And the customer knows, if they're going to deploy it, they need to find out if it works now and not later."

Santino says many of the customers who participated in the EEC would live there if they could. "They see the value of coming here and testing, and meeting with developers and program leads. That face time is extremely valuable to them."

Indeed, the customers I've spoken with who have used this facility are uniformly impressed with the quality and depth of the help and expertise they receive during their stay at the EEC. And all of them say they'd come back in the future to test other products. One of those customers, the Kentucky Department of Education, had a particularly interesting story to tell.

The Kentucky Department of Education heads to Redmond

The Kentucky Department of Education (KDE) maintains 1,400 schools in over 175 school districts statewide, a massively distributed computing environment that serves over 600,000 students, and 125,000 educators, staff members, and other employees. Previously, the KDE environment consisted of over 4400 servers, most running Windows NT 4.0, in almost 400 isolated and autonomous NT domains. Obviously, this infrastructure presented a huge problem moving forward to Active Directory (AD), the more modern directory services infrastructure that debuted with Windows 2000. The task of moving this environment forward to one that was more centralized and easily managed fell to Tim Cornett, the Active Directory Architect for KDE, and John Logan, the Exchange Architect for KDE.

"Every district had at least one NT domain, but most had two," Cornett told me. "We have to take these roughly 400 domains and collapse them to 178 domains in one Active Directory forest, and have the entire forest structure up by the end of 2003." When I spoke with Cornett and Logan in February 2003, they had just completed the first pilot program. "The first pilot was here at the Department of Education," Cornett said. "We migrated from NT 4 to Windows Server 2003 RC2." Six school districts piloted in April, and the remaining 170 school districts will migrate between May and November 2003.

Architecturally, the KDE environment will consist of a single AD forest with 178 domains. There will be a root domain, one domain for KDE employees, and one domain for each of the state's 176 school districts. This structure will provide Cornett and his colleagues with the central management that eludes them with the current, scattered, NT 4-based solution.

Why the EEC?

Originally, the KDE worked to upgrade its infrastructure to Windows 2000 on its own. But after spending 18 months evaluating and testing, Cornett realized they'd need some help. The department contacted Microsoft Consulting Services (MCS) to ask about architectural guidance, and hired a full-time technical account manager from Microsoft's Enterprise Services group. Eventually, the KDE joined Windows Server 2003 Rapid Adoption Program (RAP), which allowed them to begin working with the product early in its development process.

"Microsoft gave us the opportunity to come to the EEC to test our environment," Cornett told me. "Because of our size, it was impossible to sit here, throw some servers together and say it works. So we headed to Redmond. Everything was prepared in advance. They pre-built 178 different domains in VMWare sessions using NT 4, on about 40 different servers. And when we arrived, they helped us build the new forest root and work through the upgrade process."

"There were no compatibility issues at all," Cornett told me, "and everything ran really well. They used VMWare GSX Server on Windows 2000. We had the occasional hardware glitch here and there, but overall everything ran really smoothly. The folks there were excellent." While they were at the EEC, the KDE met several times with various product groups at Microsoft. The experience convinced Logan to move the KDE's Exchange infrastructure to Titanium, and it's likely they'll be back at the EEC soon to help facilitate that change. "Our Exchange layout is similar to our NT layout, in that it's complex and spread out," Logan said. "We want to look at greatly collapsing our Exchange topology as well. The Exchange team is very interested in talking to us about that. We're testing Titanium and Outlook [2003] now." Now part of the Rapid Deployment Program (RDP) for Exchange 2003, the KDE will likely roll out a Titanium migration by the end of 2003. The department will also deploy Microsoft Operations Manager (MOM) to help manage the massive infrastructure.

Service, service, service

Cornett and Logan came away highly impressed with the EEC and Windows Server 2003. "Any time there was a Windows Server glitch, within 20 minutes there was a person from the product team that showed up from across campus and started digging into the code," Cornett said. "It was pretty intense. We were there for 5 very long days, though they had offered it for two weeks. But it was right before Christmas. We were there 16-18 hours a day, but they were willing to stay with us 24-7 if we wanted them to. They were excellent. The resources at your disposal are seemingly unlimited. It was pretty impressive."

"Our only regret," Cornett said, "was that we didn't know enough about the lab before we got there. We would have prepared better. We had a plan, but we figured they might have just a few machines, and we'd need to spend days just getting them

ready. But when we went there, the number and quality of the machines far exceeded anything we expected. By the end of the first day, we were already at Wednesday on our plan. We could have accomplished more if we knew what resources were available before we went." Indeed, the EEC visit was so impressive, Cornett added, that it helped convert some anti-Microsoft sentiment in the UNIX crowd that went on-sight at the campus. "We took some folks with us that were anti-Microsoft, or extremely skeptical--Solaris guys, mostly. But they came back with a very positive outlook of the product and Microsoft in general."

Cornett said the big concern when they first arrived at the EEC was that Microsoft wouldn't have enough disk space to duplicate their environment, which has an enormous Global Catalog (GC). "They asked us how much space we needed, so we did the math, and it came to about 1 terabyte (TB)," Cornett told me. "On the first day, we left the EEC at 1:00 am on Tuesday, and were back in the lab at 8:30 am that morning. They had run fibre to our room, and given us access to a SAN with two 500 GB drives. We had a need and immediately, it was solved. They said, 'We were here all night but we got it for you.'

There was no question or complaint."

Before the KDE left the EEC, Microsoft backed up their environme

"In the education world we run stuff until its dead. We're hoping to get a long-term lifespan out of Windows Server 2003 without having to do major upgrading."

-Tim Cornett

Active Directory Architect for the
Kentucky Department of Education

nt so that they could put it back online and test it if the department ran into any problems when they did the production rollout. And of course, Microsoft's various product groups can come to the EEC and test their products on the complex KDE environment if desired. "Now they have a simulated environment of our size, minus all the names and other real-world data, of course," Cornett told me. "But they can use it to test, so it's good for both sides."

Meanwhile, back in Kentucky

Now that Cornett, Logan, and the other KDE team members are back in Kentucky, the real work can begin. Before Windows Server 2003 came along, the department was running on everything from aging Pentium 75 servers with 64 MB of RAM to quad Xeon boxes with 4 GB of RAM; all were running Windows NT 4.0. With the upgrade to Windows Server 2003, all of the districts' servers will be dual-processor-capable Dell PowerEdge 2600 machines. During the pilot, each of the servers will be run in single-processor mode to gauge processor utilization; if it goes over 40 percent, they'll be upgraded to dual processors. And because they got 6 years out of Windows NT 4.0--it was implemented in Kentucky in 1997--Cornett hopes to get a similar amount of use out of Windows Server 2003. "In the education world we run

stuff until its dead," he said. "We're hoping to get a long-term lifespan out of Windows Server 2003 without having to do major upgrading."

Regarding the Exchange migration, Logan told me that the overall goal there was similar to what the KDE is doing with Windows Server 2003: Simplicity of management. "We'd like to collapse our 184 Exchange sites into potentially one routing group, or a few routing groups," he said. "With Titanium, Outlook [2003], and Outlook Web Access (OWA), we will be able to do a significant amount of centralized hosting. We have an infrastructure in place to host all of the districts if we have to. This will help us more easily do backups, perform virus protection, and so on. We really want to centralize the services."

Logan's grand plan is to reduce the current collection of 320 Exchange 5.5 servers scattered around the state to just 20 Exchange Titanium Servers. The only problem with this plan is political, rather than technical, however. Some of the larger school districts might not want their Exchange Servers centrally managed. But for the smaller districts, where there is often just one technical person on staff, responsible for managing, backing up, and protecting Exchange, the centralized management will be quite welcome. There's also a small overall cost savings involved.

One might wonder how a cash-starved educational agency can afford all of this upgrading in these trouble economic times. Logan says it's all about size. "First, each district gets 100 percent matching funds for up to \$20 per student per year, so the state matches that, providing a total of \$40 per student to spend each year, plus any local funds. That's a large buying power to provide contracts for hardware and software to get it to as low a cost as possible. We get some really good discounts."

--Paul Thurrott
March 28 - April 28, 2003