realtimepublishers.com™

The Administrator Shortcut Guide™ To Configuration Management for the Windows Enterprise

Configuresoft

Don Jones

Introduction

By Sean Daily, Series Editor

Welcome to *The Administrator Shortcut Guide to Configuration Management for the Windows Enterprise*!

The book you are about to read represents an entirely new modality of book publishing and a major first in the publishing industry. The founding concept behind Realtimepublishers.com is the idea of providing readers with high-quality books about today's most critical IT topics—at no cost to the reader. Although this may sound like a somewhat impossible feat to achieve, it is made possible through the vision and generosity of corporate sponsors such as Configuresoft, who agree to bear the book's production expenses and host the book on its Web site for the benefit of its Web site visitors.

It should be pointed out that the free nature of these books does not in any way diminish their quality. Without reservation, I can tell you that this book is the equivalent of any similar printed book you might find at your local bookstore (with the notable exception that it won't cost you $30 to $80). In addition to the free nature of the books, this publishing model provides other significant benefits. For example, the electronic nature of this eBook makes events such as chapter updates and additions, or the release of a new edition of the book possible to achieve in a far shorter timeframe than is possible with printed books. Because we publish our titles in "real-time"—that is, as chapters are written or revised by the author—you benefit from receiving the information immediately rather than having to wait months or years to receive a complete product.

Finally, I'd like to note that although it is true that the sponsor's Web site is the exclusive online location of the book, this book is by no means a paid advertisement. Realtimepublishers is an independent publishing company and maintains, by written agreement with the sponsor, 100% editorial control over the content of our titles. However, by hosting this information, Configuresoft has set itself apart from its competitors by providing real value to its customers and transforming its site into a true technical resource library—not just a place to learn about its company and products. It is my opinion that this system of content delivery is not only of immeasurable value to readers, but represents the future of book publishing.

As series editor, it is my raison d'être to locate and work only with the industry's leading authors and editors, and publish books that help IT personnel, IT managers, and users to do their everyday jobs. To that end, I encourage and welcome your feedback on this or any other book in the Realtimepublishers.com series. If you would like to submit a comment, question, or suggestion, please do so by sending an email to feedback@realtimepublishers.com, leaving feedback on our Web site at www.realtimepublishers.com, or calling us at (707) 539-5280.

Thanks for reading, and enjoy!

Sean Daily

Series Editor

Introduction i
Chapter 1: Introduction 1
What Does Configuration Management Mean? 1
The Need for Configuration Management 3
    Managing Drift 3
    Managing Shift 7
Business Factors for Configuration Management 8
    Stability 8
    Security 9
    Agility 9
    Reduced Cost 9
The Complexity of Configuration Management 10
Configuration Management Life Cycles 11
Case Study: Improving Business Agility with Configuration Management 15
Summary 16

Copyright Statement

Chapter 1: Introduction

What *is* configuration management? Why do you need it in your enterprise, and how is it accomplished? Although many administrators feel that they are familiar with the topic, most have truly only scratched the surface. The IT industry in general has come a long way with configuration management in just a few short years, and the term now encompasses much more than simply building client systems by using a standardized drive image. In this guide, I'll help you understand what today's enterprise can expect from configuration management, why configuration is such a critical business need, and—most importantly—how you can add modern, enterprise-class configuration management to your network.

What Does Configuration Management Mean?

The term *configuration management* has certainly evolved quite a bit over the past few years. In the beginning, configuration management was more or less the exclusive domain of software developers, who would use software packages such as Microsoft Visual SourceSafe or the open source CVS to manage changes to an application's source code. Developers would check code out, work on it, then check in code that they had revised. The management software—CVS, Visual SourceSafe, and others—would maintain each checked-in version of the software's source code, allowing developers to roll back to an earlier version at any time, if necessary.

> Configuration management, in the software development industry, is often referred to as *change management* or *version control*. However, the underlying activity is the same: a database of past versions and strict control over changes made to source code.

Systems administrators and engineers co-opted the term *configuration management* and began applying it to the concept of maintaining a consistent, stable configuration on client and server computers. At the time, every level of IT was struggling with supporting a greater number of users and computers on a growing number of different systems. Many companies sought to reduce their support costs by adopting a single, corporate-standard brand name for desktop and laptop computers as well as servers. Support personnel would then have to be trained in supporting only one brand.

The next step was to standardize the software running on those computers, often by deploying the systems with a standardized hard drive image that contained an operating system (OS) and all approved corporate applications. This overall process, called *provisioning,* could be managed to ensure every new computer was identical to every other new computer, giving the support team fewer variables to worry about when they needed to troubleshoot those machines.

> Provisioning is generally considered to be the initial configuration of a computer; subsequent changes to that configuration might be called *re-provisioning,* but are more commonly referred to as reconfiguration, software deployment, patch management, and other terms.

Administrators quickly learned, however, that computers aren't static devices—they change over time. Users configure their desktop wallpaper, install new software applications (such as the latest must-have instant messaging software), uninstall other applications, and more. Administrators' reaction was simple: lock down the desktop. Technologies such as Windows System Policies and Windows Group Policy were deployed to restrict what users could do, helping to remove potential variation in deployed systems.

At the same time, administrators inadvertently reduced the flexibility of those systems: Users with a legitimate business need for new software often had to go through a lengthy approvals process before they could have it. Administrators were the only ones allowed to deploy new software, often using centralized deployment software such as Windows Group Policy or Microsoft Systems Management Server (SMS). In fact, this description of *configuration management* is probably close to the one your organization uses or at least strives for: A rigidly controlled environment that reduces support variables while providing a minimal degree of flexibility.

A notable feature of traditional configuration management is that it is almost entirely a *push process,* meaning configuration changes—including software updates—are deployed to machines regardless of the machines' current configuration. Deployment systems such as SMS or Group Policy typically offer very little in the way of filtering, allowing you to perhaps deploy an update only to Windows 2000 (Win2K) computers, but not providing the granularity to deploy an update only to Win2K computers that have a particular version of a piece of software installed.

Traditional configuration management also doesn't typically include a feedback mechanism. For example, you can use SMS to deploy software patches, but SMS doesn't automatically re-inventory the computer to ensure that the patch was installed and doesn't continue trying to deploy the patch if the installation fails in a way that SMS can't readily detect.

Finally, traditional configuration management is generally a one-time occurrence: Once a software update, for example, is deployed, that is the last consideration given to it. If the patch somehow becomes uninstalled or overwritten, systems such as SMS won't automatically detect the change and re-deploy the patch.

The problem is that the locked-down desktop with centralized software distribution only provides the *illusion* of control. Configuration changes still occur:

- Security updates are released and often not deployed consistently to existing computers
- Many centralized deployment systems lack a feedback mechanism to ensure that deployments actually occur successfully
- Users are still able to make changes to their desktops

In today's enterprise, the term *configuration management* is changing again. Today, the term includes not only standardized provisioning and centralized software deployment but also an automated and continuous process that monitors computers for compliance with the corporate standard. *Configuration management* encompasses maintenance, ensuring that computers not only receive updates and patches but also that they continue to possess those updates and patches even as their underlying configuration evolves. In fact, today's enterprise demands so much more from configuration management that an entirely new term is appropriate: *continuous configuration management*.

Chapter 2 will be spent entirely on continuous configuration management, including enabling technologies, enforcement mechanisms, and so forth.

The Need for Configuration Management

Industry estimates vary widely, but the general consensus is that between 60 percent and 80 percent of all IT problems are caused by misconfiguration. On a relatively simple device such as a network switch, that misconfiguration comes from one place: an administrator mistyping information into the device's configuration file. Complex, multipurpose devices such as client and server computers offer innumerable possibilities for misconfiguration. The following list offers a small sample of computer problem causes—factors that could be prevented by properly implemented configuration management:

- Viruses—Although most major antivirus software has the capability to automatically download new virus definitions, how can you be sure the software is doing so? Even forcing the auto-download configuration settings by means of Group Policy doesn't ensure that the downloads are actually reaching the computer.
- Hackers—Even if you are the most conscientious of administrators, ensuring that every security update issued by Microsoft is installed—and not all administrators can make that claim—there is still the possibility that users will install some additional piece of software that undoes or overwrites a security update.
- Crashes—You might have used Group Policy to deploy the latest version of an application—the version that doesn't crash, that is—but you've not taken any steps to ensure that the deployment was successful to each machine.

The whole point of configuration management is to manage computers' configurations so that viruses, hackers, and crashes are impossible—or at least, a lot less likely.

Imagine if 60 percent to 80 percent of your IT support calls suddenly went away. In today's "do more, with less" economy, organizations are often spending too much time fighting fires and handling problems rather than implementing new projects and growing the IT infrastructure. If even half of your support problems suddenly vanished, imagine the kind of effort and attention you could devote to new projects, new technologies, and new business agility. There are two main factors that are helping to keep you focused on support calls: *drift* and *shift*. Even if you've got a solid provisioning system in place to deploy new software, drift and shift will still keep your support call volume as high as it always has been.

Managing Drift

Drift is what causes configurations to vary over time from their original baseline. For example, suppose you deploy every new computer with a standardized corporate image. That means all computers begin with the exact same configuration baseline. In theory, if you don't deploy any new software to the computer and if nothing changes, that baseline configuration will remain the same over time, meaning all of your computers will continue to have identical configurations. Figure 1.1 shows this lack of change in chart form.

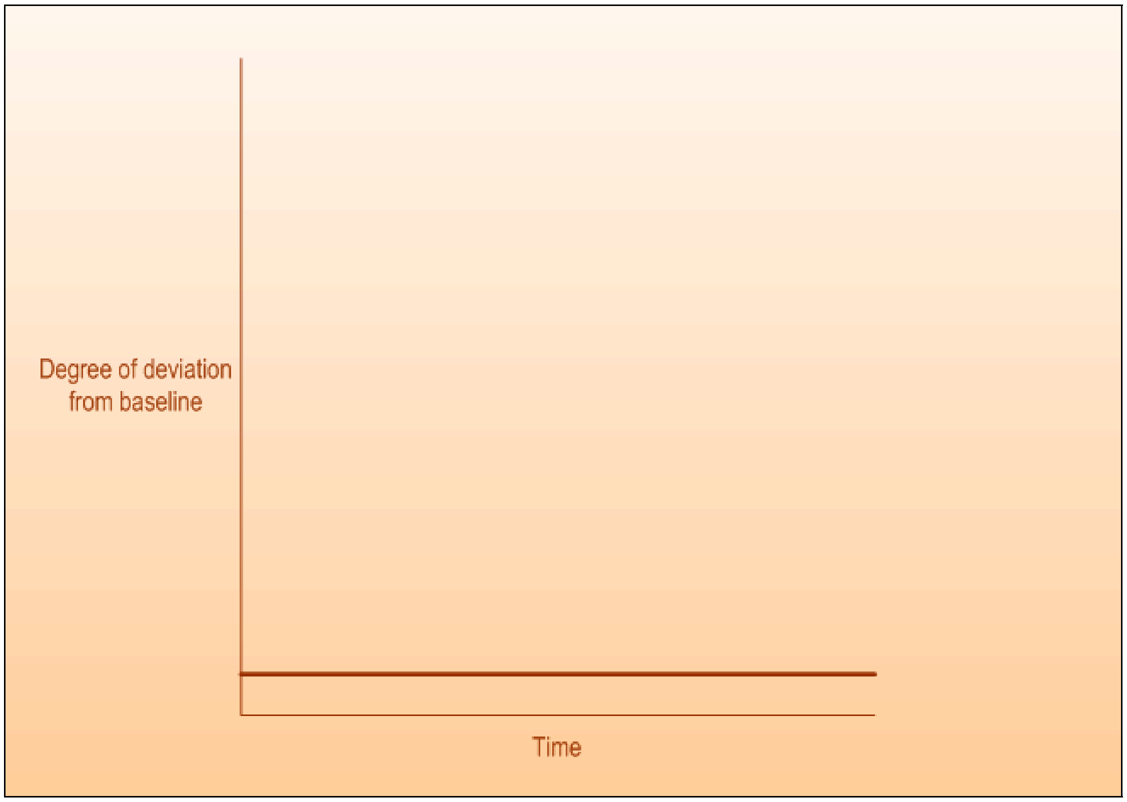

Figure 1.1: In theory, a baseline configuration shouldn't change over time.

Even if you deploy new software components—applications, patches, and so forth—the deviation from the baseline should still be zero, because *every* computer is receiving the new updates. In effect, you are simply updating the baseline. In reality, though, users install software that overwrites patches, their computers miss updates, and so forth. Figure 1.2 shows the degree of deviation, or *drift*, that the computer's configuration has over time.

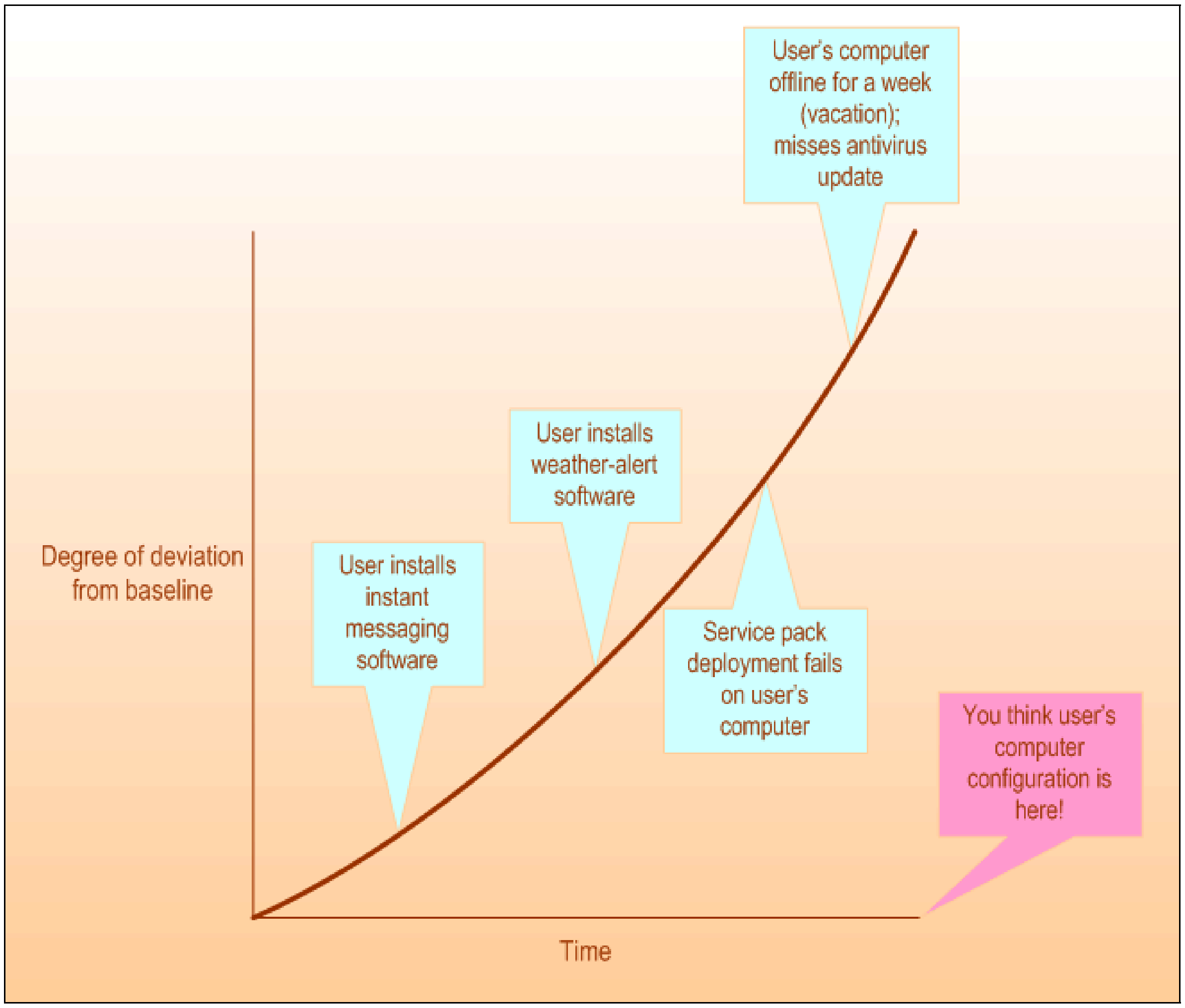
User's computer offline for a week (vacation); misses antivirus update
User installs weather-alert software
Degree of deviation from baseline
User installs instant messaging software
Service pack deployment fails on user's computer
You think user's computer configuration is here!
Time

Figure 1.2: Configuration drift causes increasing deviation from the baseline over time.

The problem with this scenario is that you're deploying patches and updates based on what you *think* the computer's configuration looks like; the fact that the computer's real configuration is markedly different means that you're not effectively managing its configuration. In this sense, drift occurs for a couple of very basic reasons:

- Configuration activity outside your control
- Configuration activity within your control that doesn't complete properly

But you can't simply correct drift by locking down the user's computer even further. Simply preventing unknown changes from occurring won't stop configuration drift.

For example, consider Figure 1.3. The brown line shows the changes in your baseline configuration that you would expect all computers to have. The dotted blue line shows the actual configuration of a particular computer. Everything's fine up until the point when that computer crashes, and you restore it from a backup. At that point, the computer's configuration resets to the point in time when the backup was made. Although updates continue to be applied to the computer, it's no longer in compliance with the evolving baseline as you might expect. This deviation ca cause problems because the "missing" portion of the baseline might include critical security patches, antivirus updates, or even new applications.

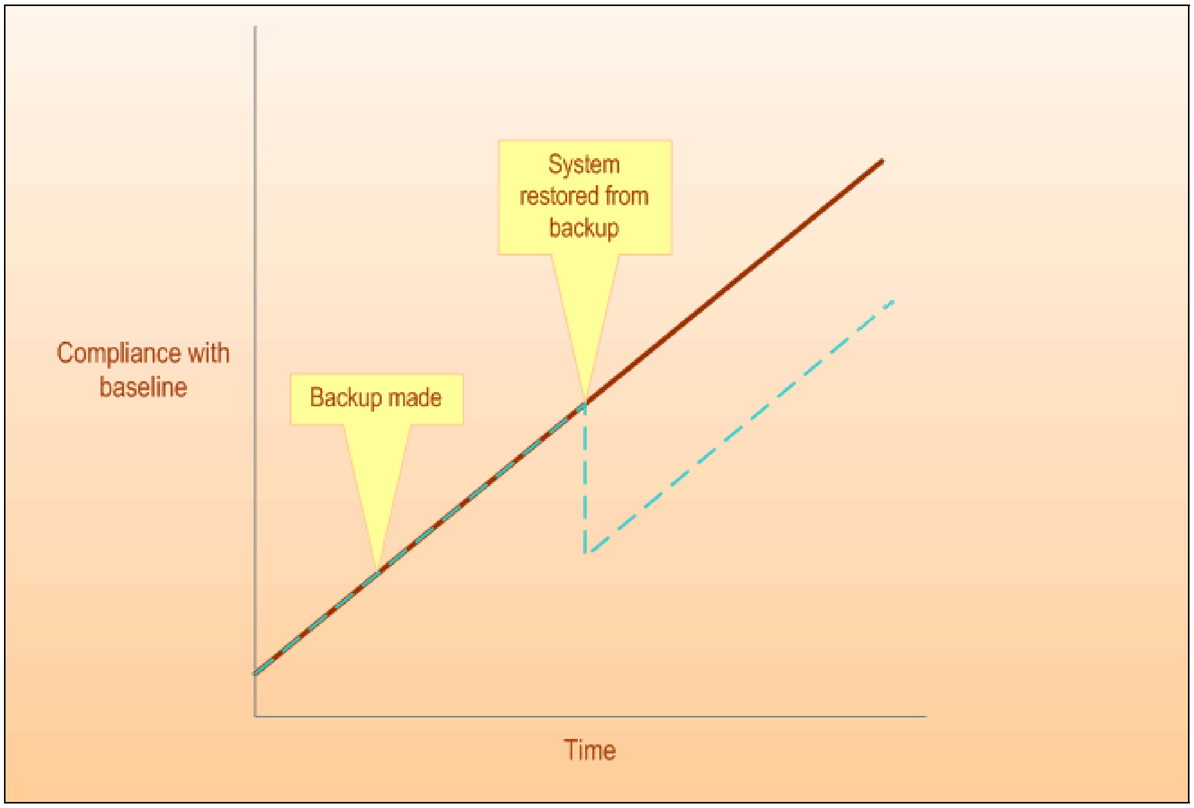

Figure 1.3: Point-in-time restores result in configuration drift.

These system restores don't need to be manual events involving backup tapes: Windows XP includes a System Restore feature that can roll back a computer's configuration to undo software or device driver installations. Rolling back a System Restore checkpoint undoes everything that has occurred since that checkpoint, and can make it nearly impossible to tell what legitimate, desirable updates were also removed by the rollback.

Traditional configuration management doesn't catch these problems because the management is a one-time occurrence: Computer A already received Patch A, so there's no reason to re-deploy Patch A to that computer. Traditional configuration management systems aren't designed to realize that the underlying system configuration has changed in an unexpected fashion. As an administrator, you might realize that a bunch of past patches need to be reapplied, but any system that relies on human realization and memory is bound to fail because people will make mistakes: they'll forget to reapply patches or which patches need to be applied.

Microsoft's Windows Update is one form of configuration management that is relatively continuous. Each time a computer visits the Windows Update Web site (http://www.windowsupdate.microsoft.com), the computer is scanned for all patches that apply to it, regardless of the computer's past patch-application activity. If a patch is applied and then subsequently removed, Windows Update will detect the shortcoming and reapply the patch. Unfortunately, Windows Update only covers software patches, and only for certain software products at that; it's far from a complete configuration management system and isn't designed to be one. In addition, Windows Update is a *pull* system. There is no centralized control over which patches will be deployed; Microsoft simply makes the patches available, and the client determines which updates to download and install.

Configuration drift also occurs to servers, which can be more severe than client drift because servers are typically assigned a role within an organization. For example, consider an organization that has Web servers, file servers, and database servers. It's not unusual for these to be completely separate machines; they typically have their own unique baseline configurations and application updates and OS patches are applied accordingly. Now suppose a file server has Web server software applied, allowing it to become an intranet Web server. Suddenly, the machine's role has changed—it has expanded and now fills two roles and needs a new, hybrid configuration. Its freshly installed Web server software is missing the updates that have already been applied to other Web servers, and the dual nature of its function on the network will make it impossible to simply include in both the file server and Web server patch programs. For example, a patch applied to a file server—one which wouldn't ordinarily be applied to a Web server—might create a security vulnerability in the Web server software that isn't addressed by a patch you've installed. The hybrid server's configuration has now drifted outside the baseline for both file and Web servers, making it more difficult to keep the machine maintained in the future.

The lesson to be learned from all of this is that *you cannot stop configuration drift* by using traditional techniques. What's more, you shouldn't try. Traditional techniques primarily involve eliminating all potential for change by locking down the systems involved. The problem with this technique is that it removes all flexibility and agility from the business' computers, making it more difficult for users to gain the functionality they need to do their jobs. You can, however, *respond* to configuration drift through a process of continuous configuration management.

Again, the point here is that traditional configuration management relies on a static baseline and seeks to prevent drift and shift; in doing so, it often prevents or degrades business flexibility at the same time. Continuous configuration management—as a result of its continuous nature—doesn't rely entirely on a baseline and can accommodate shift and drift much more readily, meaning systems can remain more flexible while still conforming to a standardized configuration.

Managing Shift

Configuration *shift* is the other enemy of baseline maintenance. Shift occurs when your desired baseline configuration changes. For example, new best practices might evolve, dictating a fundamentally different set of configuration values for your computers. One good example of this evolution is the second service pack for Windows XP Professional, which, by default, installs and enables the Windows firewall software to help better protect the computer against incoming attacks.

Another example is when users' usage patterns change, such as when a company begins to shift employees' responsibilities. Users who used to work entirely in an office might now find themselves in the field more often, substantially changing the way they need to use their computers and changing what they need from the baseline configuration. Unlike configuration drift, which is often subtle, configuration shift is often more dramatic in scope, requiring significant changes to the way systems are configured.

Configuration shift is a good thing, however, because its fundamental purpose is to provide needed functionality for critical business functions. You need to accommodate shift; the danger simply lies in the fact that shifted systems will deviate from your carefully prepared baseline, making it more difficult to ensure that the systems are properly configured for their roles.

Again, the problem of maintaining the configuration on shifted systems is that most configuration management techniques rely on a fixed point-in-time baseline configuration, then rely exclusively on “push” updates to the system. Configuration management techniques don’t traditionally incorporate a feedback mechanism that allows systems to be maintained at the correct configuration in a more dynamic fashion. The following list provides a pie-in-the-sky description of how configuration management might better accommodate shift:

- A user receives a new computer configured with a baseline that is optimized for use within the office.
- The user is reassigned to a field position and now needs a different set of software applications and different access configurations. You use a central deployment technology, such as Group Policy, to deploy the new software.
- A background continuous configuration management process scans all systems on a regular basis, looking for patterns. When the process finds the pattern indicative of the new applications and access configuration, it deploys the associated updates.
- That same background process performs a complete, from-scratch check each time it runs. If the user somehow reconfigures the computer or reinstalls an application, the process again detects that certain updates are necessary and applies them.

This process accommodates both shift and drift because it is constantly working to keep systems at a desired baseline. It can accommodate shifts in that baseline by continually analyzing each aspect of the system, and it can accommodate drift by continually monitoring the system and comparing thousands of configuration values to their desired state.

Business Factors for Configuration Management

Businesses need configuration management. There is rarely any doubt of that; most organizations will happily describe the drive-imaging techniques that they use to deploy new computers, and demonstrate the central software deployment technologies that allow them to push out security updates overnight. But most businesses fail to clearly define the business reasons behind configuration management, which is why configuration management often winds up not fulfilling what the business actually needs from their IT resources. The following sections explore a partial list of business factors that should drive configuration management:

- Stability
- Security
- Agility
- Reduced cost

Stability

Above all else, businesses need computer systems to remain functional. Configuration management can provide stability by ensuring that computers are always configured in a known-stable condition, with tested software drivers; the latest, tested software updates and patches; and so forth.

The problem with traditional configuration management is that it's too easy for factors such as software drivers to get updated outside administrators' control or for other stability factors to become compromised—that's what configuration drift is all about. If the configuration management process were *continuous*—rather than one-time in nature—approved, updated drivers would be deployed to any system that does not have them, even if the drivers had previously been deployed to a system and subsequently removed or overwritten by the user.

Security

Security is most often a combination of configuration settings and software updates. Configuration management can provide security by ensuring that computers *always* remain configured with an approved combination of settings, and that *all* software installed on the computer always has the latest approved patches. The problem with traditional configuration management is that it relies on words such as "always" and "all." Centralized software deployment pushes a software update once; after that, the update isn't redeployed automatically, even if the update is somehow removed or overwritten on a particular system. In other words, traditional software deployment is a one-time process—not a continuous process—commanded by an administrator.

Agility

Information systems are designed to support business operations, not the other way around. Given that simple fact, you should expect information systems to quickly change and evolve to meet rapidly changing business situations. In fact, few businesses actually enjoy an agile IT infrastructure because IT professionals often fear change. Change causes problems: new software contains unknown issues, new configurations create unanticipated problems, and so forth. In fact, much of traditional configuration management is dedicated to limiting agility by severely limiting what users can do with their machines.

A more agile, *continuous* configuration management process can help eliminate some of the fear of change. Users can be given the ability to install new applications or to make configuration changes to their computers because the continuous configuration management process will ensure that standardized configurations, updates, and applications remain in effect, allowing users only to change less-sensitive portions of their systems.

Reduced Cost

Systems that have problems require support, and support costs time and money. In today's economy, every dollar spent on supporting existing systems is a dollar not spent on new systems, which means high support costs can impact a business' ability to evolve and meet new challenges. Traditional configuration management seeks to reduce cost by eliminating or reducing change, which can have the same effect of reducing business' ability to evolve.

Continuous configuration management, however, embraces change. Change is permitted *and* support costs are reduced, because the continuous configuration management system is built with the understanding that change occurs and that the end configuration of a system can still be carefully managed even while that change is occurring: Permitted changes are analyzed and configured to a secure, stable condition, while disallowed changes are reversed or removed. The entire system is continuously examined for configuration compliance, allowing change to occur while helping to improve security and stability and to reduce costs.

☞ Perhaps the most fundamental concept behind continuous configuration management is that it does not rely entirely on a baseline; instead, it simply considers the current state of each computer and acts accordingly to modify the configuration, deploy software updates, and so forth. Computers that experience drift or shift are accommodated by the continuous nature of the process.

The Complexity of Configuration Management

One reason that configuration management hasn't evolved to meet complex business requirements is that configuration management in Windows-based systems (or in any other OS, for that matter) is incredibly complex. Consider configuration management in a network router: The entire configuration exists in a single file, which is typically only a few kilobytes in size. The configuration is in a straightforward, text-based format, and can be easily queried through technologies such as Trivial File Transfer Protocol (TFTP).

Now consider the total configuration of a Win2K computer: Some information is stored in the Windows registry, which is a flat database stored in a proprietary binary format, and for which only a few interfaces exist for retrieving and examining the data. Additional information is stored in a local Security Accounts Manager (SAM), which is accessible through a completely different set of interfaces. Server software—Dynamic Host Configuration Protocol (DHCP) servers, Domain Name System (DNS) servers, Windows Internet Naming Service (WINS) servers, Web servers, and more—all have their own unique databases for storing configuration information, and each provides a completely different means of accessing that information programmatically. Some information, such as the contents of a WINS server, is practically impossible to access. Other configuration data, such as the state of Win2K disk quotas, is stored in a practically hidden location within the file system and is difficult to access without using the OS's graphical user interface (GUI).

Although Microsoft has made great strides in recent years to develop a centralized means of accessing management and configuration information—Windows Management Instrumentation (WMI)—the technology is vast, complex, and still doesn't cover even half of the configuration information within a Windows machine. I haven't even mentioned Active Directory (AD—which has its own management interfaces) or installed applications, patch management, or system security—all of which have their own unique interfaces and structures. In short, it's sometimes a wonder that human beings can configure a Windows system, and no wonder at all that automated configuration management systems haven't traditionally been able to do so.

Even complex systems management software such as SMS doesn't provide a complete solution. SMS is an incredible piece of software, and it can provide an incredible amount of information about your systems. However, it can't easily tell you that a user's computer isn't configured to use Encrypting File System (EFS) or that a particular SQL Server installation has a blank password for the built-in systems administrator account. This shortcoming is not a fault of SMS; it is simply a byproduct of the incredible complexity within every Windows-based system.

What's more, there is no sense in trying to reduce that complexity. The complexity arises out of the flexibility offered by Windows; flexibility which allows a computer to become a database server, file server, print server, certification authority (CA), Web server, and more. That complexity exists to help businesses become more agile and more efficient; in fact, that complexity is the sole purpose behind modern computing (and it isn't just Windows—Linux- and UNIX-based systems are equally as complex). In other words, the complexity is the reason you're using a computer; removing the complexity would remove much of the reason for having the computer.

However, the complexity doesn't make it any easier to manage the computer's configuration. Configuration management of complex systems, then, requires a complex management system; one that recognizes the variety of ways in which configuration information must be accessed and is prepared to deal with it.

In Chapter 2, I'll discuss continuous configuration management systems and the underlying technologies they use to accomplish their work.

Configuration Management Life Cycles

Configuration management must recognize the concept of a *life cycle*, meaning a continuous process with defined stages and sub-processes. This idea isn't as complex as it sounds—simply consider the average computer in your organization. It's deployed with a fresh image, receives various applications and updates over the months and years, has problems occur, has problems fixed, and is eventually retired. During that life cycle, it might also receive a brand-new image and be redeployed to another user for another purpose, effectively beginning the life cycle over again. The term *life cycle* is important, especially the *life* part: Life is rarely a neat, orderly process, and the life cycle of most computers is rarely neat and orderly, either.

Consider the life cycle that Figure 1.4 shows. In this figure, two things are happening:

- Centralized configuration management, which takes care of the machine's initial provisioning and handles deployment of approved applications and updates.
- Unmanaged, unexpected changes, such as an unapproved application, a deployed application update that didn't install properly, and a user who installs unexpected updates.

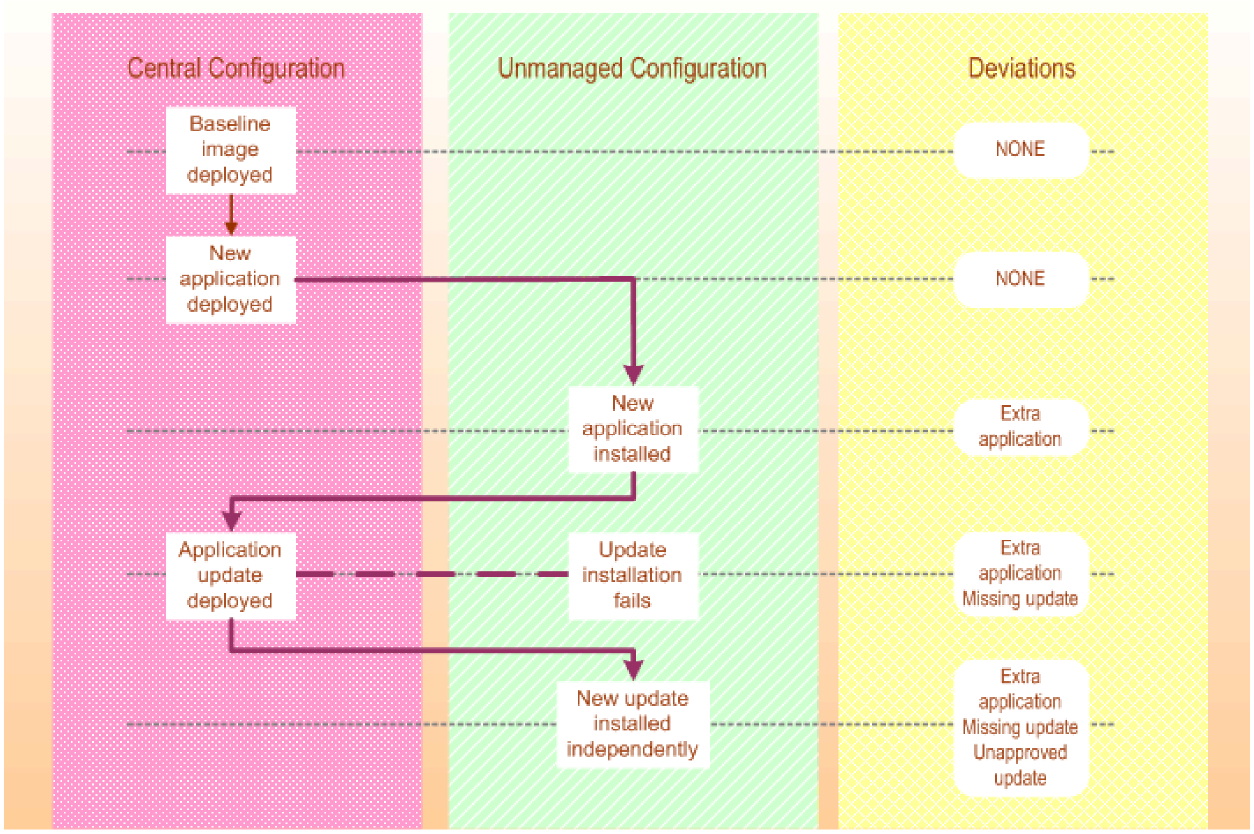
Central Configuration
Unmanaged Configuration
Deviations
Baseline image deployed
NONE
New application deployed
NONE
New application installed
Extra application
Application update deployed
Update installation fails
Extra application Missing update
New update installed independently
Extra application Missing update Unapproved update

Figure 1.4: The life cycle of an average computer.

The end result is a strong deviation from the expected norm, which is what drift is all about. Figure 1.6 illustrates what is missing from most configuration management techniques: recognition of the reality that unexpected configuration changes do occur, like it or not. This model of configuration management is blindly pushing out changes without any back-and-forth between each computer to accommodate individual drift and shift.

Next, consider Figure 1.5, which shows a more continuous process that includes a feedback mechanism.

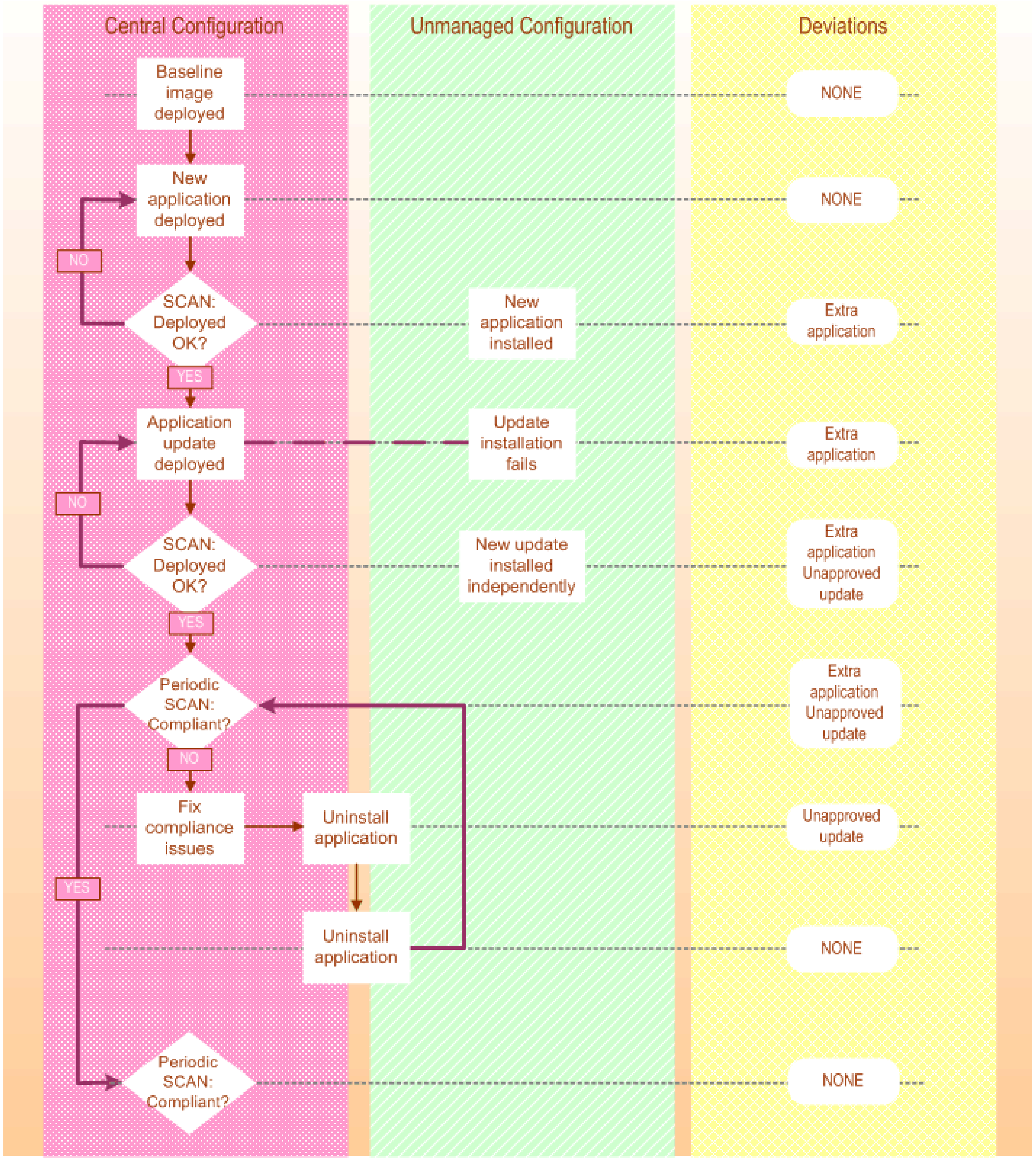

Figure 1.5: Continuous configuration management with feedback.

In this scenario, the user is still making unapproved configuration changes. However, the process incorporates a feedback mechanism. For example, an update that fails to deploy is detected and accommodated by a continuous re-deployment (and, one would assume, some reporting mechanism to alert administrators that the automated deployment was failing).

> This detection is *not* simply detecting that a Windows Installer package failed to run; the detection is actually checking for the files, configuration settings, and other items that make up a "fingerprint" for the update, physically verifying that the update is installed and properly configured.

A periodic scan allows the system to detect changes that weren't approved and to correct them. In this case, the correction involves uninstalling the unapproved application. By the time the computer has been in its life cycle for a while, it remains in compliance with the corporate standard. Figure 1.6 shows an even more compelling version of this life cycle, and assumes that the application installed by the user is in fact an approved application, for which two updates exist. The user independently discovers and installs one of these, while the configuration management process steps in to install the other—automatically. The computer is, at the end of the process, completely compliant and has the benefit of the extra flexibility provided by allowing that application to be installed exactly when the user needed it.

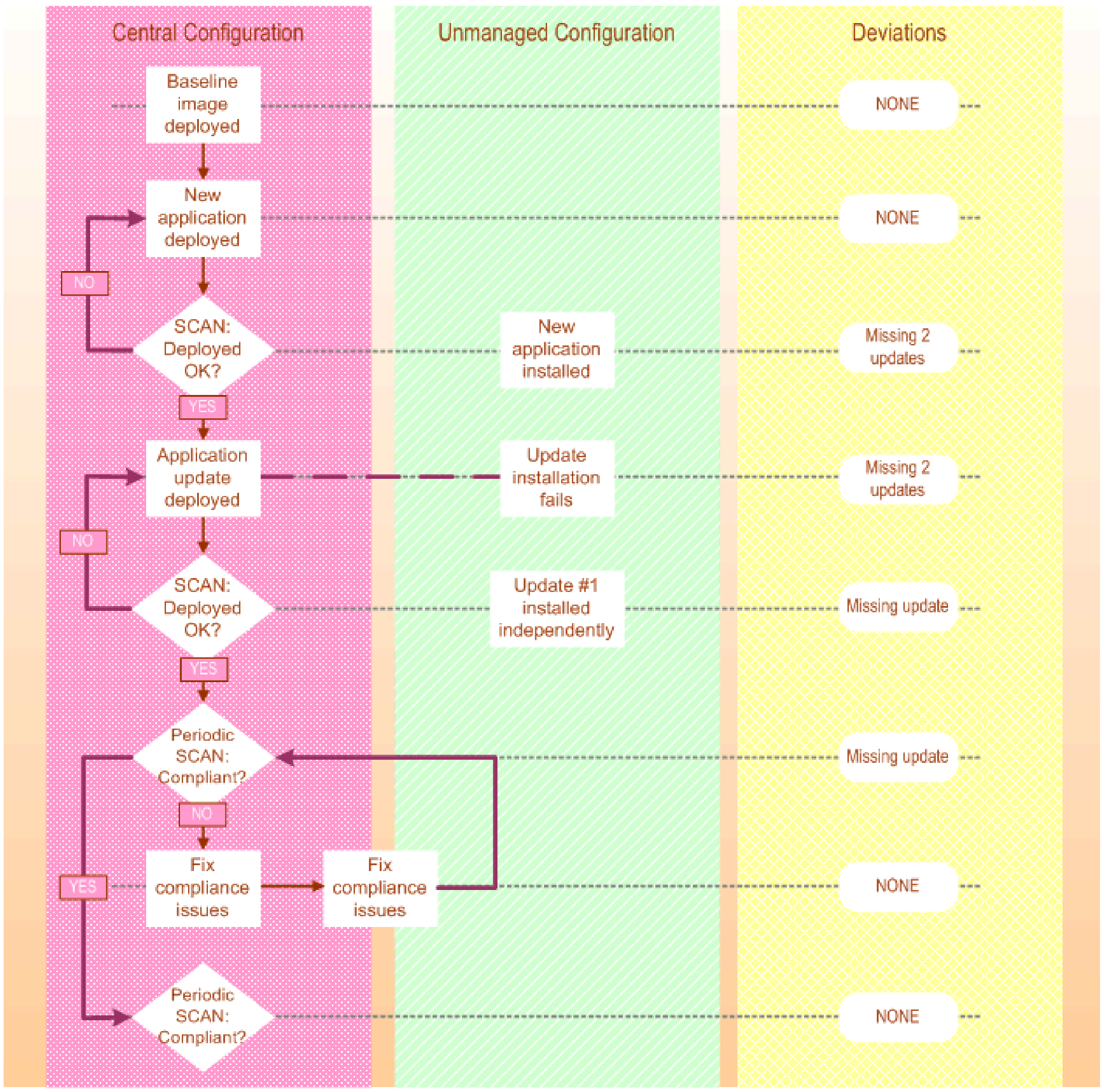

Figure 1.6: Continuous configuration management keeping a new application updated.

Case Study: Improving Business Agility with Configuration Management

As a closing note, I'll offer the story of a consulting client I worked with recently. This financial services firm has strict operational and security requirements imposed not only by company policy but also by regulations such as the Sarbanes-Oxley Act, which applies to most financial services institutions. The firm's initial approach to the Sarbanes-Oxley Act was to lock down its systems tighter than ever before (and this is a company that always believed in locking down systems) so that the company could deploy a known-compliant configuration to its systems and be absolutely assured that that configuration wouldn't change.

The company's IT department had much experience with locking down systems and was one of the first companies to implement a major Windows System Policies rollout when Windows 95 came out and Microsoft introduced its Zero Administration for Windows (ZAW) initiative for Windows 95 and Windows NT 4.0. Now, working with Win2K, Windows XP, and Group Policy, the company figured a more controlled and strict lockdown would be easy. And it was: within a few months, every computer on the network had the latest patches, was tightly configured, and was practically immune to changes that users might attempt.

And then the problems occurred. Application updates to line-of-business (LOB) applications became difficult to deploy because users didn't have permission to install updates on their systems. When new divisions of the company began offering insurance products, their computers weren't configured to support their business needs, and users were forced to work around a number of restrictions and problems until scrambling administrators could modify the computers' baseline configurations. In fact, the lack of flexibility in the IT infrastructure led to the new insurance products being offered 3 months later than expected, which cost the company millions of dollars in lost revenue for those 3 months and put them a month behind their biggest competitor in offering those products. *That* caught the attention of upper management.

Flexibility had never been voiced as a business need, and the current configuration management process reflected this shortcoming. Management now decided that flexibility was necessary, and directed IT to come up with a system that would:

- Allow changes to be made to systems more easily.
- Enforce specific Sarbanes-Oxley Act-sensitive configurations, but other types of changes would have to be allowed.
- Enable users to be able to install new software applications and updates, and IT would have to find a way to ensure that these installations remained configured appropriately.
- Facilitate users who need to move between departments more frequently to keep up with changing customer demand, meaning users' computers would need to be able to run a variety of software form different departments, and IT would need to ensure that each piece of software had the correct configuration and latest updates.

In short, IT could no longer rely on a single baseline, and they wouldn't be able to push out configuration changes and updates on a one-time basis. IT needed to find a way to enforce certain configurations and updates regardless. Several things were obvious:

- It would no longer be sufficient to use Group Policy to deploy software; while Group Policy could still be used to enforce certain registry-based configurations, Group Policy simply wasn't designed to analyze the target system and select appropriate software and updates from a library.
- Additional techniques would need to be used to enforce more esoteric computer configurations, especially those configurations that didn't involve the registry and were therefore more difficult for Group Policy to access.
- Some sort of interactive inventory-analyze-configure methodology would need to be used so that computers could be continuously analyzed for configuration compliance, even though the roles of those computers might change on a daily basis.
- The entire process had to be automated because the company supported more than 40,000 desktops—more than any manual process could hope to keep up with.

The goal was clear: continuous configuration management was needed. I'll continue this case study in Chapter 2, where you'll learn how the company tried to get a handle on the configuration information it needed to manage, and how the firm developed a system to meet management's new business requirements.

Summary

In this chapter, I've introduced you to the fundamental concepts of both traditional configuration management and continuous configuration management. Traditional configuration management relies on static baselines and centrally managed, point-in-time updates and deployments; continuous configuration management relies not on a baseline but on a set of definitive configuration standards. The continuous process constantly monitors systems and reconfigures them for compliance; traditional configuration management simply performs a one-time deployment or configuration and doesn't incorporate a feedback mechanism to allow for future corrective re-deployments or re-configurations. As such, continuous configuration management can provide for greater business flexibility, while traditional configuration management seeks to reduce or eliminate change—an often unnecessary restriction on business agility.

In the next chapter, I'll dive into continuous configuration management in more detail. I'll introduce you to the underlying technologies that make this management possible, and provide some insight into how the various aspects of the process—including provisioning, deployment, management, and enforcement—work in an enterprise environment.