Realtime publishers

The Definitive Guide™ *To*

Active Directory Troubleshooting, Auditing, and Best Practices

2011 Edition

Don Jones

Chapter 5: Active Directory Auditing 63

Goals of Native Auditing 63

Native Auditing Architecture 63

Common Business Goals for Auditing 71

Weaknesses of Native Auditing 72

Third-Party Auditing Capabilities 74

Coming Up Next 76

Download Additional eBooks from Realtime Nexus! 76

Copyright Statement

Chapter 5: Active Directory Auditing

The previous chapter was about Active Directory's (AD's) declarative security—that is, how you tell the directory who has permission to do what. We also had a look at how AD's security is designed and built, and how AD as an authentication mechanism interfaces with Windows' native authorization mechanisms. Those were the first two of the "three As," and the third one—*auditing* or *accounting*—is the focus of this chapter.

Goals of Native Auditing

Auditing has a fairly simply goal: Keep track of everything everyone is doing. Within the context of AD, that means keeping track of all uses of privilege, such as changing group memberships or unlocking user accounts. It also means keeping track of account activity, such as successful logons and failed logons. Extending that scope to Windows, auditing includes keeping track of file and folder access as well as changes to file permissions.

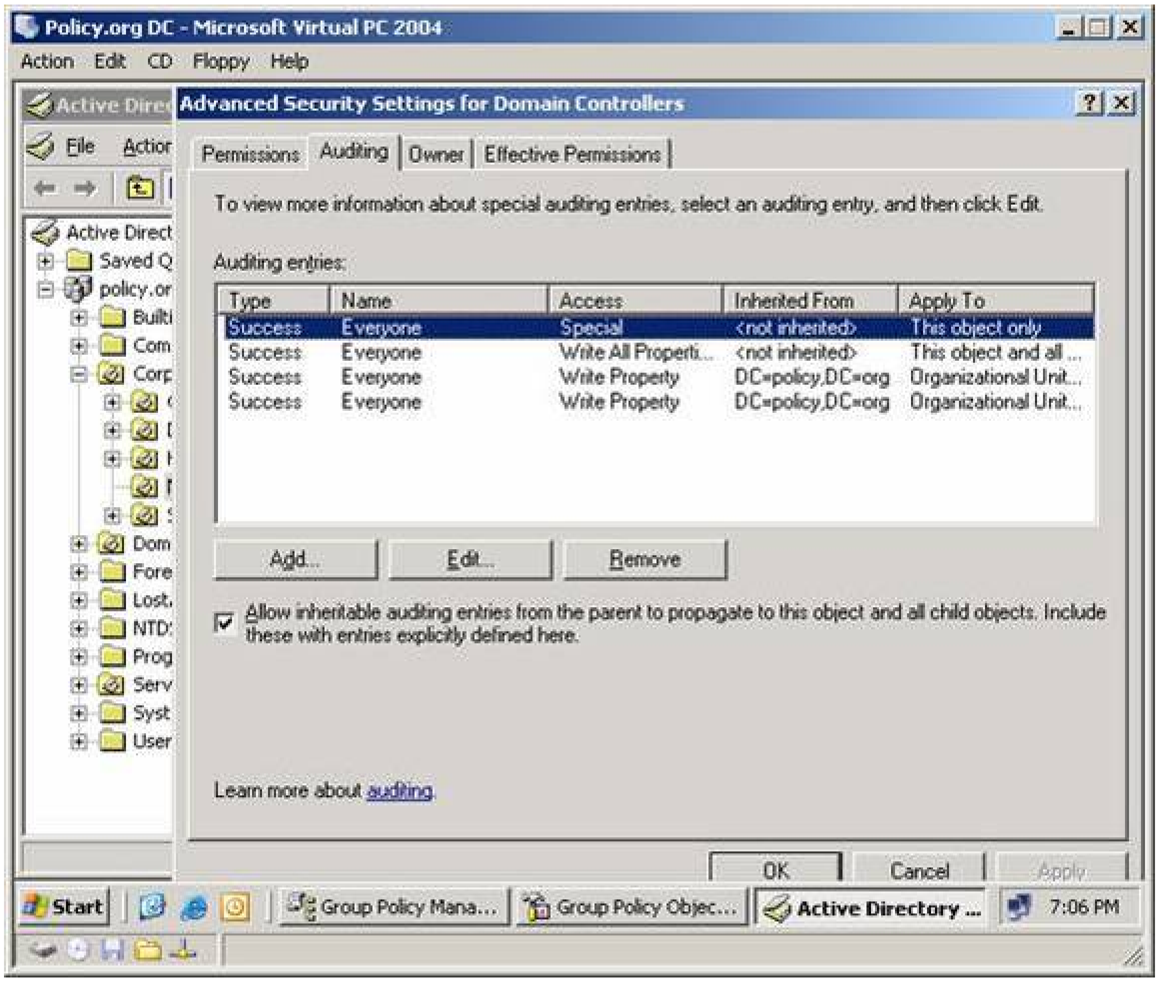
Your goals for auditing might differ somewhat from the goals of the operating system's (OS's) auditing architecture. Keep in mind that the auditing system used in Windows—including AD, which essentially just copied the architecture of the file system—dates back to the early 1990s when Windows NT was being designed and written. At that time, Microsoft couldn't have predicted organizations with thousands of file servers, dozens or hundreds of domain controllers, and thousands of other servers running Exchange, SQL Server, SharePoint, and other business platforms. The fact is that Windows' native auditing architecture doesn't always scale well to especially large environments, or even to some midsize ones—a fact we'll explore later in this chapter. So although *you* might want to audit *every single event* in your environment, actually doing so may create performance challenges, management challenges, and even logistical challenges. For right now, let's just assume your goal is indeed to audit everything that happens in your environment, and see where the architecture takes us.

Native Auditing Architecture

In the previous chapter, you learned that permissions are applied to a Discretionary Access Control List (DACL). Each DACL consists of one or more Access Control Entries (ACEs), and each ACE grants or denies a specific set of permissions to a single security principal—that is, a user or a group. The DACL is the *authorization* part of the AAA model: AD *authenticates* you, and gives you a security token containing a unique Security Identifier (SID). That SID is compared with the ACEs in a DACL to determine your permissions on a given resources.

Auditing works in much the same way. A Security Auditing Control List (SACL) consists of one or more entries. Each entry designates a specific auditing action for activities conducted by a single user or group. The SACL is attached to a resource, like a file or directory object, and whenever the specified security principal engages in the specified activity with that resource, the action is logged. Typically, you have the ability to log "success" and/or "failure" actions. That is, you can choose to log an entry when someone successfully exercises their permissions or when they attempt to do so and are denied.

Figure 5.1 shows a SACL configuration for AD. As you can see, this resource—the "Domain Controllers" organizational unit (OU)—is configured to log several success actions performed by the special Everyone group. That is, whenever anyone successful performs any of these actions, an audit entry will be generated.


Policy.org DC - Microsoft Virtual PC 2004
Action Edit CD Floppy Help
Advanced Security Settings for Domain Controllers
Permissions Auditing Owner Effective Permissions
Add... Edit... Remove
OK Cancel Apply
Start Group Policy Mana... Group Policy Objec... Active Directory ... 7:06 PM

To view more information about special auditing entries, select an auditing entry, and then click Edit.

Auditing entries:

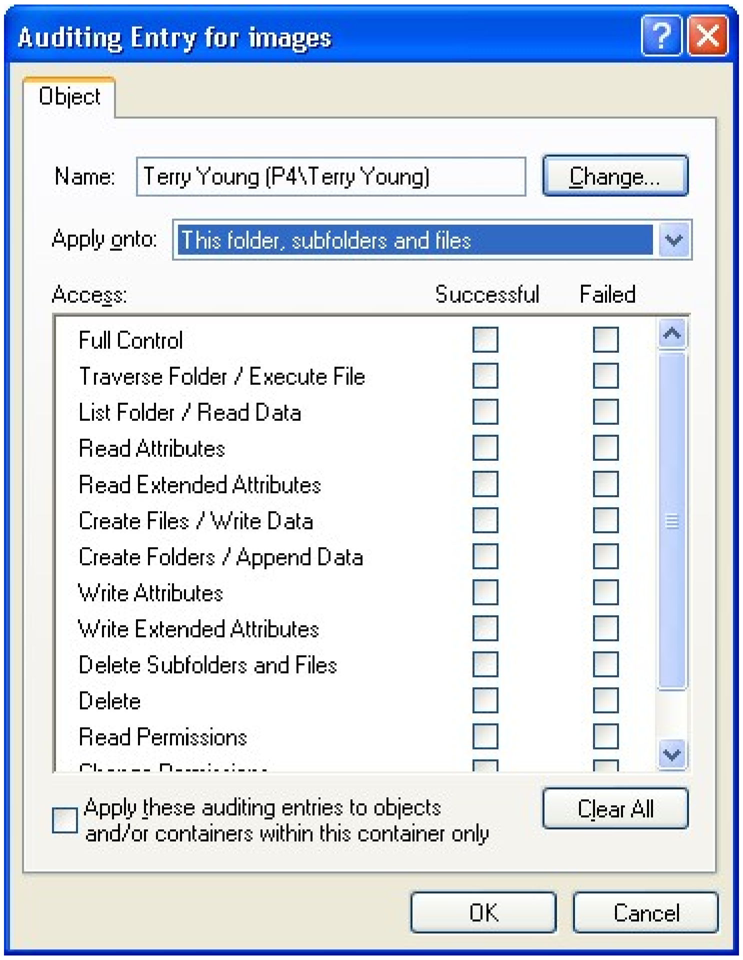
Type	Name	Access	Inherited From	Apply To
Success	Everyone	Special	<not inherited>	This object only
Success	Everyone	Write All Properti...	<not inherited>	This object and all ...
Success	Everyone	Write Property	DC=policy,DC=org	Organizational Unit...
Success	Everyone	Write Property	DC=policy,DC=org	Organizational Unit...

☑ Allow inheritable auditing entries from the parent to propagate to this object and all child objects. Include these with entries explicitly defined here.

Learn more about auditing.

Figure 5.1: SACL in AD.

Exactly what actions you can audit depends on what resource you're working with. For example, Figure 5.2 shows a file system SACL, and you can see that very different actions are available.



Name: Terry Young (P4\Terry Young)

Apply onto: This folder, subfolders and files

Access:	Successful	Failed
Full Control	☐	☐
Traverse Folder / Execute File	☐	☐
List Folder / Read Data	☐	☐
Read Attributes	☐	☐
Read Extended Attributes	☐	☐
Create Files / Write Data	☐	☐
Create Folders / Append Data	☐	☐
Write Attributes	☐	☐
Write Extended Attributes	☐	☐
Delete Subfolders and Files	☐	☐
Delete	☐	☐
Read Permissions	☐	☐

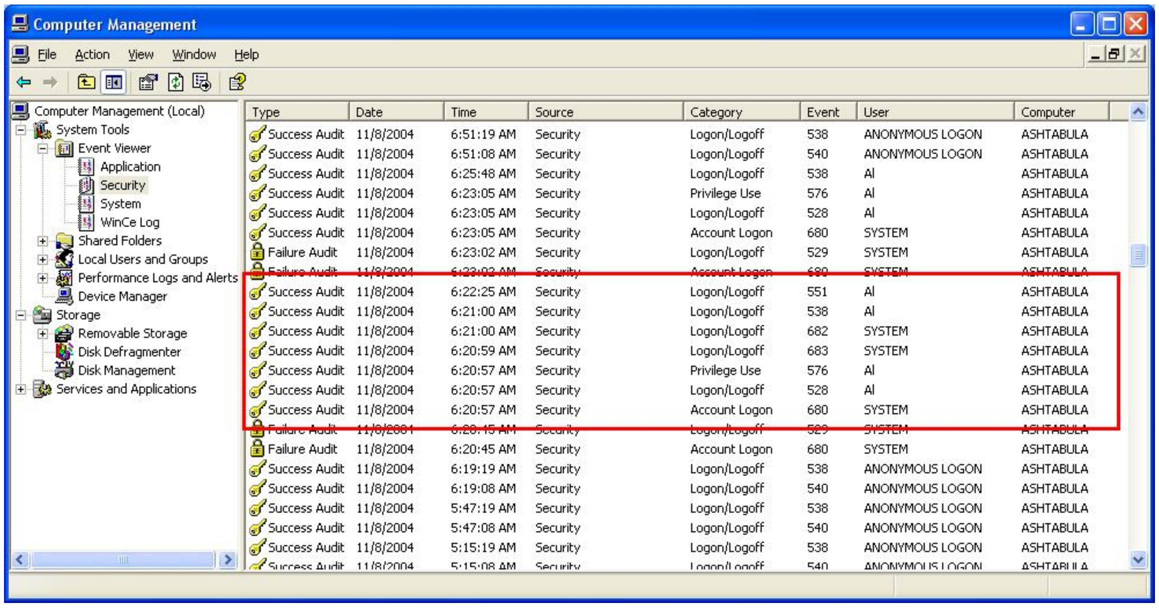
☐ Apply these auditing entries to objects and/or containers within this container only

Figure 5.2: A file system SACL.

Here, you can choose to audit things like creating folders, reading attributes, deleting files, and so on. Each resource, then, can have its own SACL. In practice, most of us assign SACLs at a fairly high level in the hierarchy and let those settings propagate to lower-level objects through inheritance. That way, we only have to manage SACLs in a relatively small number of places. But we still have to configure at least one top-level SACL per server, per major system. That is, each server will need a top-level SACL on at least the root of each logical drive, we'll need a separate SACL on the root of AD, and so on.

Other products may or may not follow this pattern. Exchange Server, for example, uses a similar structure for its auditing; SQL Server does not, nor does SharePoint. We'll stick with AD and the core Windows OS for the discussion in this chapter.

Once an auditable action occurs, Windows generates an audit entry. These are stored in the Security event log, which Figure 5.3 shows. A problem with this log is that *every* auditing event goes into it. Although it's nice to have everything in one big, central pile, it can make it tough to pull out specific entries. Again, this reflects Microsoft's relatively limited original vision for the auditing system.

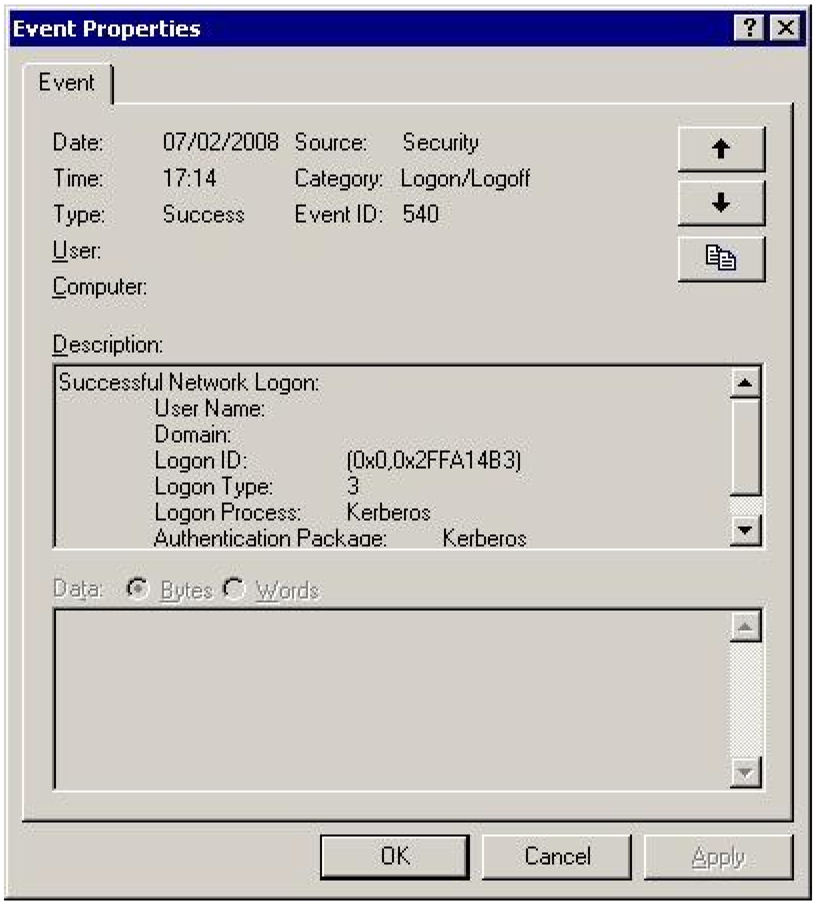


Type	Date	Time	Source	Category	Event	User	Computer
Success Audit	11/8/2004	6:51:19 AM	Security	Logon/Logoff	538	ANONYMOUS LOGON	ASHTABULA
Success Audit	11/8/2004	6:51:08 AM	Security	Logon/Logoff	540	ANONYMOUS LOGON	ASHTABULA
Success Audit	11/8/2004	6:25:48 AM	Security	Logon/Logoff	538	Al	ASHTABULA
Success Audit	11/8/2004	6:23:05 AM	Security	Privilege Use	576	Al	ASHTABULA
Success Audit	11/8/2004	6:23:05 AM	Security	Logon/Logoff	528	Al	ASHTABULA
Success Audit	11/8/2004	6:23:05 AM	Security	Account Logon	680	SYSTEM	ASHTABULA
Failure Audit	11/8/2004	6:23:02 AM	Security	Logon/Logoff	529	SYSTEM	ASHTABULA
Failure Audit	11/8/2004	6:23:02 AM	Security	Account Logon	680	SYSTEM	ASHTABULA
Success Audit	11/8/2004	6:22:25 AM	Security	Logon/Logoff	551	Al	ASHTABULA
Success Audit	11/8/2004	6:21:00 AM	Security	Logon/Logoff	538	Al	ASHTABULA
Success Audit	11/8/2004	6:21:00 AM	Security	Logon/Logoff	682	SYSTEM	ASHTABULA
Success Audit	11/8/2004	6:20:59 AM	Security	Logon/Logoff	683	SYSTEM	ASHTABULA
Success Audit	11/8/2004	6:20:57 AM	Security	Privilege Use	576	Al	ASHTABULA
Success Audit	11/8/2004	6:20:57 AM	Security	Logon/Logoff	528	Al	ASHTABULA
Success Audit	11/8/2004	6:20:57 AM	Security	Account Logon	680	SYSTEM	ASHTABULA
Failure Audit	11/8/2004	6:20:45 AM	Security	Logon/Logoff	529	SYSTEM	ASHTABULA
Failure Audit	11/8/2004	6:20:45 AM	Security	Account Logon	680	SYSTEM	ASHTABULA
Success Audit	11/8/2004	6:19:19 AM	Security	Logon/Logoff	538	ANONYMOUS LOGON	ASHTABULA
Success Audit	11/8/2004	6:19:08 AM	Security	Logon/Logoff	540	ANONYMOUS LOGON	ASHTABULA
Success Audit	11/8/2004	5:47:19 AM	Security	Logon/Logoff	538	ANONYMOUS LOGON	ASHTABULA
Success Audit	11/8/2004	5:47:08 AM	Security	Logon/Logoff	540	ANONYMOUS LOGON	ASHTABULA
Success Audit	11/8/2004	5:15:19 AM	Security	Logon/Logoff	538	ANONYMOUS LOGON	ASHTABULA
Success Audit	11/8/2004	5:15:08 AM	Security	Logon/Logoff	540	ANONYMOUS LOGON	ASHTABULA

Figure 5.3: The Security event log.

Each Windows server maintains its own individual Security event log—that includes domain controllers. Although AD's SACLs can be configured on any domain controller, and will replicate to all of them, only the domain controller that actually handles a given action will create an audit entry for it. The result is a centrally-configured auditing policy but a highly-distributed auditing log.

Figure 5.4 shows what these audit entries look like. They're fairly technical, and often include raw SIDs and other under-the-hood information. This example shows a successful domain logon, processed using the native Kerberos protocol. The user name and domain have been blanked out for this example but would normally be populated when a real user logs on.



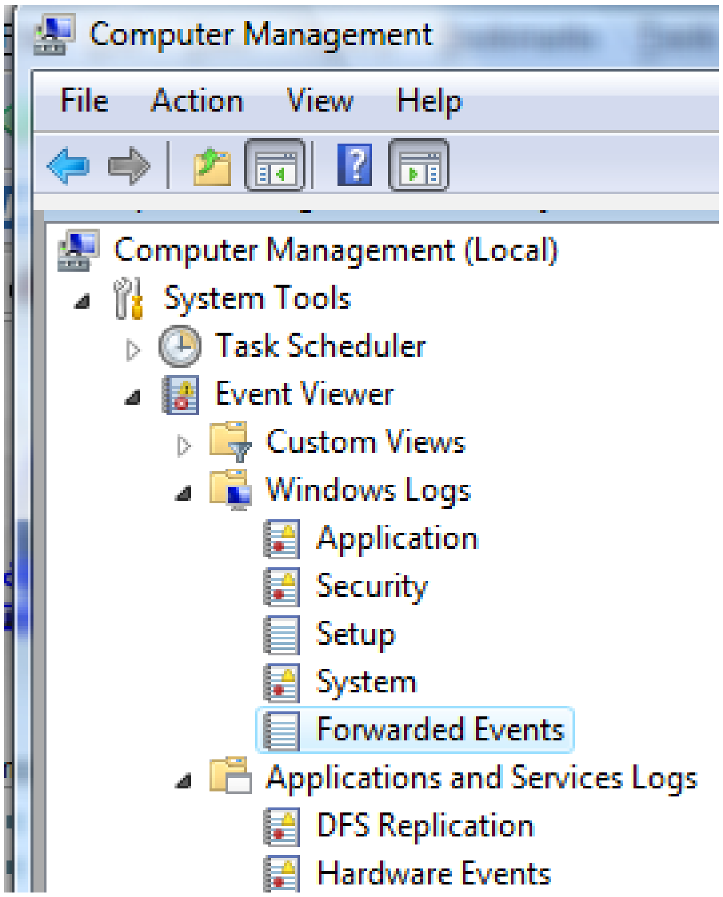
Date:	07/02/2008	Source:	Security
Time:	17:14	Category:	Logon/Logoff
Type:	Success	Event ID:	540
User:			
Computer:			

Description:

```
Successful Network Logon:
        User Name:
        Domain:
        Logon ID:         (0x0,0x2FFA14B3)
        Logon Type:       3
        Logon Process:    Kerberos
        Authentication Package:   Kerberos
```

Figure 5.4: An example audit entry.

Microsoft has already begun to address the issue of one log holding so much information. In Windows Vista and Windows Server 2008, Microsoft introduced a parallel event log architecture that makes it easier for each product or technology to maintain its own log. This was always possible—the original Application, System, and Security logs have long been supplemented by logs for Directory Services, for example. But this new architecture is more robust in several ways. Figure 5.5 shows some of the old and new-style logs.

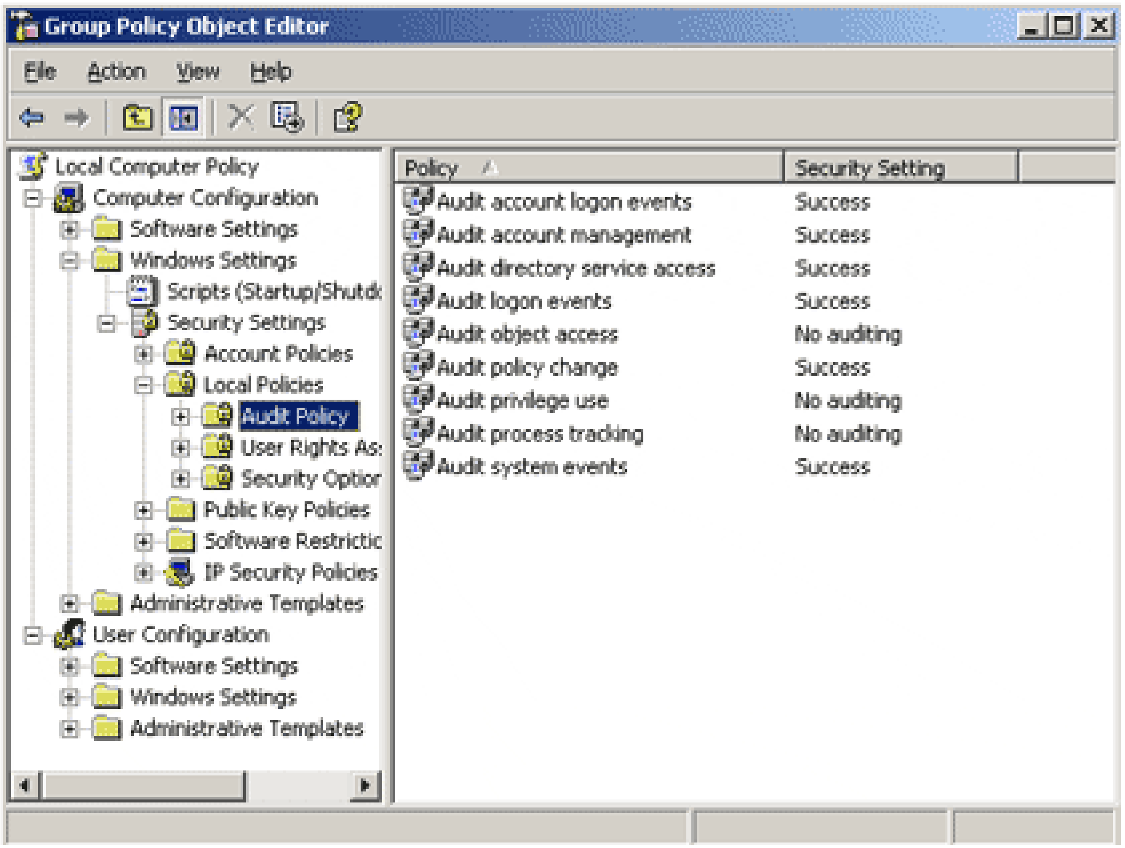


Computer Management (Local)
System Tools
Task Scheduler
Event Viewer
Custom Views
Windows Logs
Application
Security
Setup
System
Forwarded Events
Applications and Services Logs
DFS Replication
Hardware Events

Figure 5.5: New logs alongside the old logs.

Unlike DACLs, SACLs are not immediately utilized by the OS. SACLs simply designate what actions, what security principals, should be audited; the auditing system itself must also be turned on in order for events to be written to the logs. Figure 5.6 shows where that is usually configured in a Group Policy object (GPO).

Most organizations will configure auditing at a high-level GPO, such as one applied to all domain controllers, or even to all servers in the domain. The GPO pictured is specifically setting the audit policy, which includes turning on auditing of logon events, account management activity, access to AD, and so forth. The audit policy, as well as resource SACLs, must both be configured in order to generate the desired auditing events.

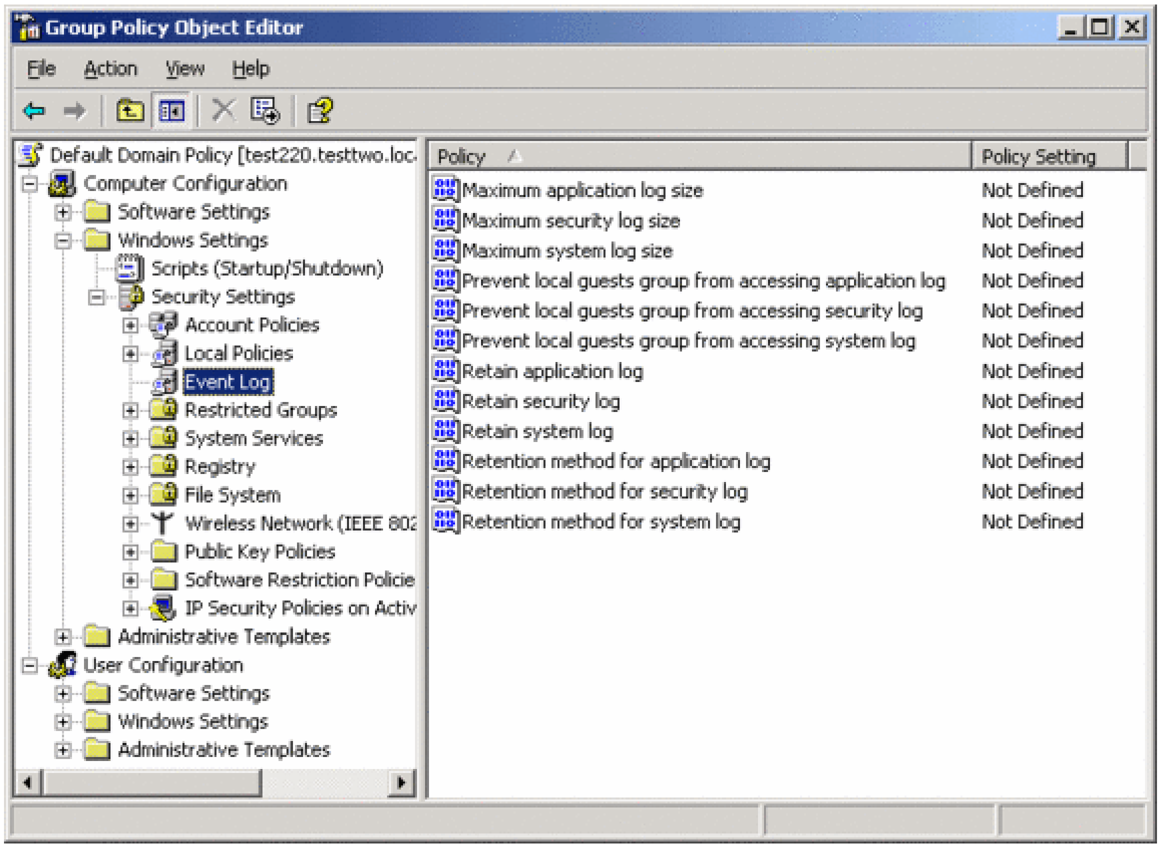


Policy	Security Setting
Audit account logon events	Success
Audit account management	Success
Audit directory service access	Success
Audit logon events	Success
Audit object access	No auditing
Audit policy change	Success
Audit privilege use	No auditing
Audit process tracking	No auditing
Audit system events	Success

Figure 5.6: Configuring auditing in a GPO.

This is where you have to use some caution. You don't want to turn on full-bore auditing without thinking about the consequences. A domain controller can generate thousands of logon events every minute during the busy morning logging-on rush, and generating all of those events requires computing power. If auditing all of those events is truly a requirement, then you're going to have to size your domain controllers accordingly to handle the load. The same goes for file servers: If a file server is expected to generate an event for every successful or failed file access attempt, it's going to need to have the computing power necessary to pull it off.

Generating that much log activity can also pound the actual event logs pretty hard. As Figure 5.7 shows, you'll want to pair your audit policy with a well-planned event log policy, setting event logs' sizes, rollover behavior, and other settings to accommodate the workload you plan for them to handle.



Policy	Policy Setting
Maximum application log size	Not Defined
Maximum security log size	Not Defined
Maximum system log size	Not Defined
Prevent local guests group from accessing application log	Not Defined
Prevent local guests group from accessing security log	Not Defined
Prevent local guests group from accessing system log	Not Defined
Retain application log	Not Defined
Retain security log	Not Defined
Retain system log	Not Defined
Retention method for application log	Not Defined
Retention method for security log	Not Defined
Retention method for system log	Not Defined

Figure 5.7: Configuring event log settings in a GPO.

The Security log—which is where auditing events are written—can be especially tricky. With the Application log, you might feel comfortable simply allowing it to overwrite itself when it gets full. For the Security log, you can't practically do that, or you'd open up the door for auditing information to be lost. Instead, you'll have to configure an appropriate log size, and implement maintenance procedures to archive and clear the log on a regular basis—perhaps as often as every evening, depending upon the load you're putting on that log.

A common criticism of Windows' native event logs is their highly-distributed nature. For example, an administrator could modify a group membership on one domain controller, connect to a second domain controller to use an account in that group, and connect to a third domain controller to reset the group membership. All three actions would be logged in three different Security event logs, making it difficult to correlate those independent events into a chain of activity.

Microsoft's initial solution to this problem, introduced in Windows Server 2008, is *event log forwarding.* Pictured in Figure 5.8, the idea is that individual servers can forward events to a central server, which collects all of the events in its own log.

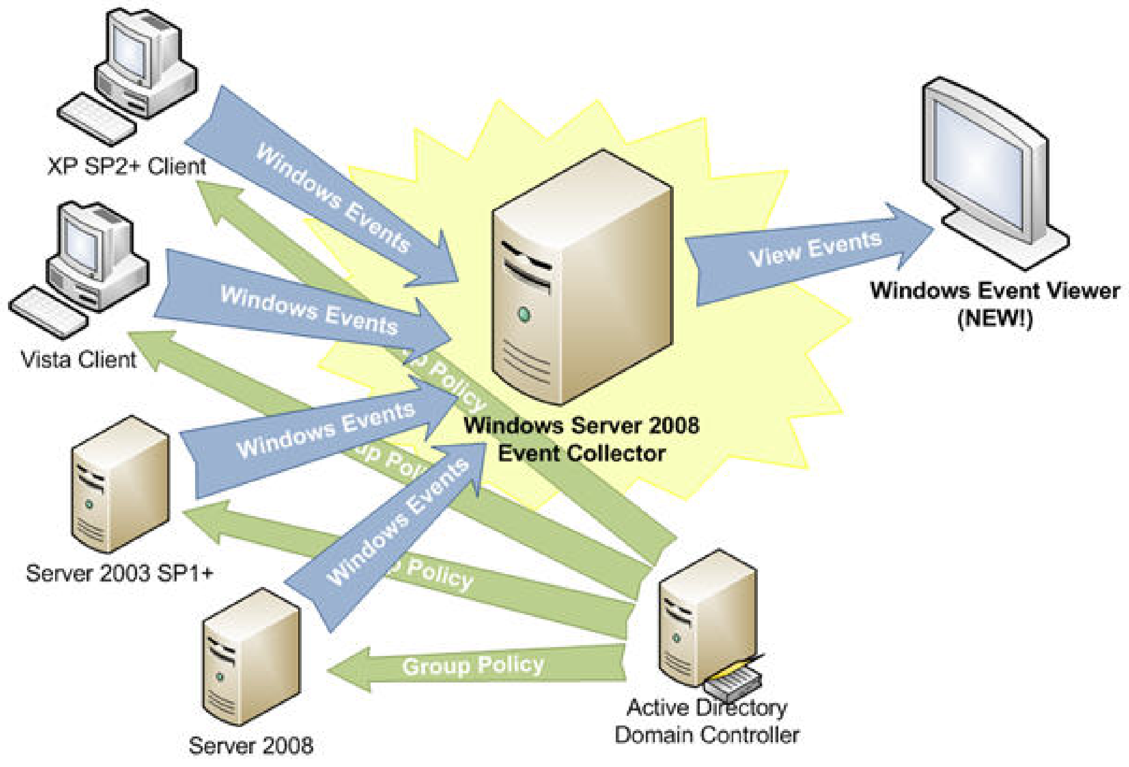


Figure 5.8: Event log forwarding.

As indicated, this feature can be configured with Group Policy, making it centrally controllable. The approach still has some significant drawbacks, however, which we'll discuss later in this chapter.

So that's how the native auditing system is built. Let's talk a bit more about how organizations want to use that system, and see where it might need enhancement.

Common Business Goals for Auditing

Unlike the 1990s when Windows NT was designed, most businesses today are subject to some kind of security policy. In many cases, that policy incorporates external requirements from industry rules or even legislation. Those requirements may include a need to audit every successful and failed action for pretty much *everything* in the environment—and that generates a *lot* of auditing traffic.

Another goal is for that auditing information to be tamperproof, or at least tamper-evident. In other words, the people being audited—including administrators—shouldn't be able to remove their own audit activity from the audit log. Organizations also want to be able to search, filter, and report on those events. For example, an auditor might want to see every audit entry that corresponds to a reconfiguration of AD's audit policy, then match each of those events to an approved action. That lets an auditor see that the only changes made to the directory were those that had been formally documented and approved.

Organizations also need to use these audit events for troubleshooting purposes. When something goes wrong in the environment, answering the question "What changed?" is usually the quickest way to solve the problem—and the audit logs should be able to answer that question quickly and effectively. So how does the native auditing system hold up?

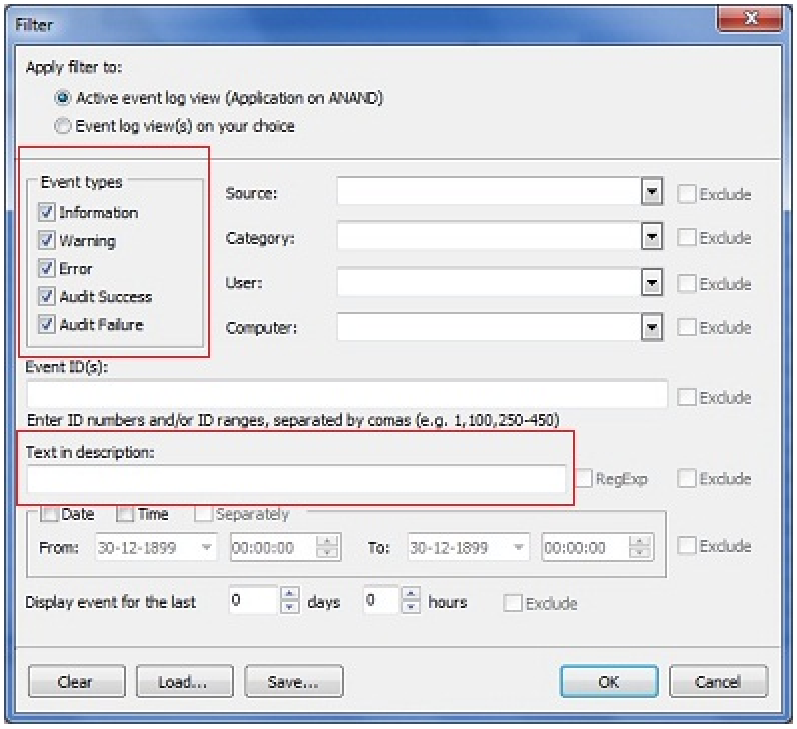
Weaknesses of Native Auditing

Unfortunately, the native auditing system does not always hold up well. I really don't regard this as a weakness on Microsoft's part—after all, their job isn't to anticipate every possible business need, but rather provide a platform on which other software can be deployed to meet specific, varying business needs. They've done that. The native auditing architecture is bare-bones, suitable for the smallest organizations that are less likely to be able to afford add-on software to meet specific business needs. The native system is also close to three decades old, and you can't always expect systems of that age to meet every possible modern requirement.

Goal one, being able to audit *everything,* is certainly possible within Windows—although you'll need to play log capacity and server performance around that goal. The native event log architecture isn't as performance-transparent as it perhaps could be, and asking a server to audit tens of thousands of events an hour *will* create an impact on that server.

Goal two—a tamper-evident log—is where the system really falls apart. Unfortunately, it's just not feasible to take away administrators' ability to clear the event log. You can do it, by carefully tweaking privileges, creating dedicated log-management user accounts, and so on—but it's complex, and many organizations find it impractical.

Even assuming you do so, meeting the next goal—centralized reporting, filtering, and alerting—isn't practical, either. Event log forwarding, even when used, doesn't occur in real time—there can be significant delays in events being forwarded. Even when you do rely on event forwarding, you're massing a *log* of log information into a single place, and relying on an extremely primitive event viewer for querying that log. Figure 5.9 shows the filtering capabilities of the native tool, and they're indeed primitive.



Filter

Apply filter to:

[✓ Active event log view (Application on ANAND)]
Event log view(s) on your choice

Event types
[✓] Information
[✓] Warning
[✓] Error
[✓] Audit Success
[✓] Audit Failure

Source: [ ] Exclude
Category: [ ] Exclude
User: [ ] Exclude
Computer: [ ] Exclude

Event ID(s): [ ] Exclude
Enter ID numbers and/or ID ranges, separated by comas (e.g. 1,100,250-450)

Text in description: [ ] RegExp [ ] Exclude

[ ] Date [ ] Time [ ] Separately
From: 30-12-1899 00:00:00 To: 30-12-1899 00:00:00 [ ] Exclude

Display event for the last 0 days 0 hours [ ] Exclude

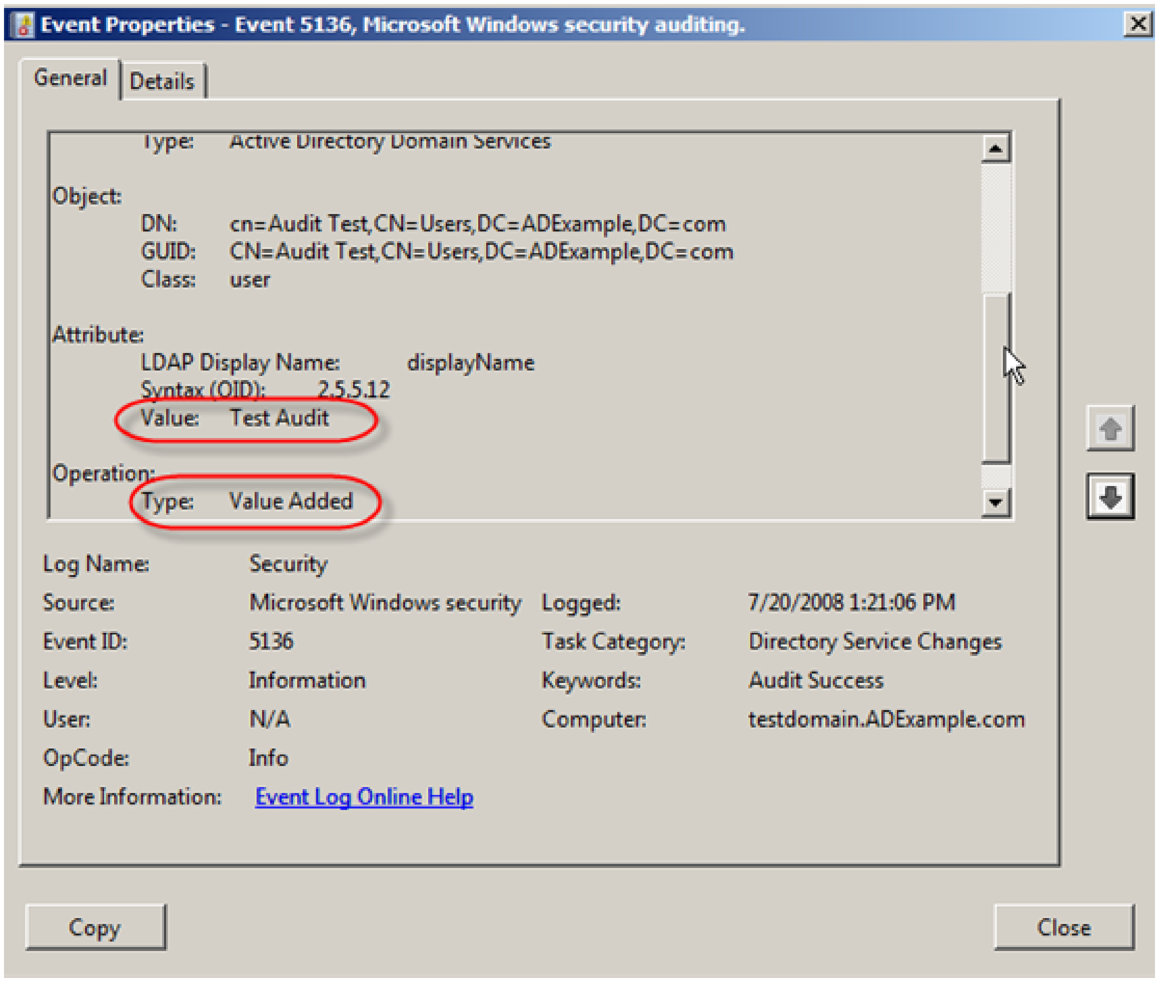
Clear | Load... | Save... | OK | Cancel

Figure 5.9: Native event log filtering.

As shown, you can filter for specific event types, and filter for specific text in the event description, as well as other criteria. But there's no way to correlate multiple related events in a chain of activity, and there's no reporting mechanism to speak of.

As for the final goal of using these events for troubleshooting—well, good luck. It's certainly possible, although it usually takes the form of, "see what's in the log, look up the event IDs to see what they mean, and figure out if that's relevant to the current problem." It's much harder to ask the native event viewer to give you, "all changes made to AD within the past 4 hours." Although there will be events related to those changes—provided your audit policy is capturing them—the event log isn't really designed to facilitate change management or change auditing. It isn't auditing the *change,* per se, it's auditing the *fact that someone made a change.*

As Figure 5.10 shows, Windows Server 2008 AD did start capturing "before and after" values in changes, making it a bit more usable for change auditing. However, the feature still isn't pervasive throughout all of AD, and finding the actual events in a massive log file can still be challenging.


Event Properties - Event 5136, Microsoft Windows security auditing.
General
Details
Copy
Close

```
        Type:    Active Directory Domain Services

Object:
        DN:      cn=Audit Test,CN=Users,DC=ADExample,DC=com
        GUID:    CN=Audit Test,CN=Users,DC=ADExample,DC=com
        Class:   user

Attribute:
        LDAP Display Name:      displayName
        Syntax (OID):       2.5.5.12
        Value:   Test Audit

Operation:
        Type:    Value Added
```

Log Name:	Security		
Source:	Microsoft Windows security	Logged:	7/20/2008 1:21:06 PM
Event ID:	5136	Task Category:	Directory Service Changes
Level:	Information	Keywords:	Audit Success
User:	N/A	Computer:	testdomain.ADExample.com
OpCode:	Info		
More Information:	Event Log Online Help		

Figure 5.10: Enhanced events in Windows Server 2008.

In most environments, a successful auditing program almost always involves third-party auditing supplements.

Third-Party Auditing Capabilities

Third-party auditing tools take several approaches to supplementing Windows' native capabilities. First, these tools may do a better (and faster) job of collecting events from multiple servers' logs into a central location. Often, that central location is a SQL Server database, although other tools will always forward events in real-time to an external logging mechanism, such as a syslog server—as Figure 5.11 illustrates.

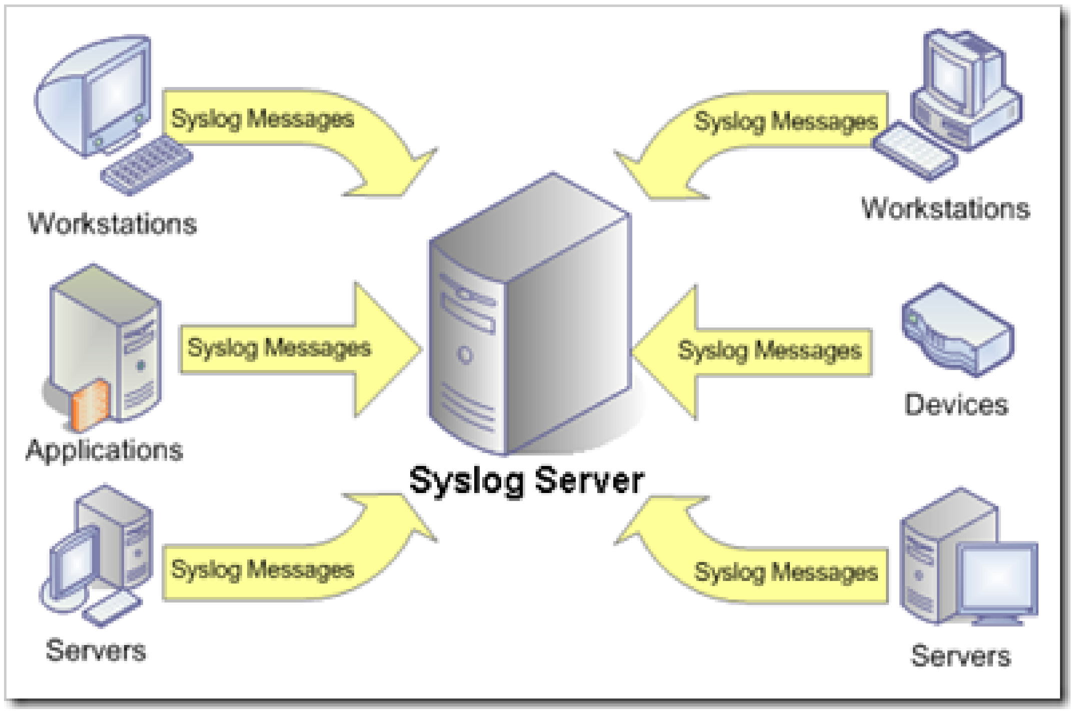


Figure 5.11: Forwarding events to a syslog server.

The idea is mainly to get the events *out of Windows* as quickly as possible, and into some separate system that can be secured differently from the environment's event logs. Databases are popular choices because they can be secured and they naturally lend themselves to complex queries, and thus, to reporting capabilities. In fact, many third-party auditing tools collect events in SQL Server mainly to leverage SQL Server Reporting Services as a reporting mechanism.

Third-party tools may also tap directly into native Application Programming Interfaces (APIs) to collect audit information—in addition to, or instead of, using the native event logs. These APIs often offer more detailed information, including better "before and after" details. In some cases, using the APIs may offer a better-performing way of collecting the information, reducing server load.

Once the event data is centrally located, third-party tools can kick in with real-time alerts, reporting, event archiving, analysis and collation, and much more. The trick is in getting the events into a single spot that can be queried quickly and effectively.

Coming Up Next

In the next chapter, I'll start summarizing many of the techniques and concepts from this and the preceding chapters, and presenting them to you as best practices. We'll start with a look at when, and why, you might want to reconsider your directory's design—as scary a concept as that might be! We'll also look at best practices for disaster recovery and business continuity, security, replication, FSMO placement, DNS design, and more. We'll wrap with a consideration of virtualization because that's all the rage these days, and discuss how suitable AD is, or isn't, for living inside a virtual machine. I'll also throw in some practices for ongoing AD care and feeding, to keep your directory healthy and happy.