realtimepublishers.com™

Tips and Tricks Guide™ To Active Directory Troubleshooting

NETPRO
The Directory Experts

Don Jones

Note to Reader: This book presents tips and tricks for Active Directory troubleshooting topics. For ease of use and for cross referencing, the questions are numbered.

Copyright Statement

Q.10: How does Kerberos work?

A: Kerberos is an industry-standard authentication protocol and part of the TCP/IP suite of internetworking protocols. Originally developed at MIT, Kerberos is now up to version 5, which is primarily defined in Internet Request for Comments (RFC) 1510.

> You can find RFC 1510 at http://www.ietf.org/rfc/rfc1510.txt. Microsoft has proposed several extensions to Kerberos that are used in Windows 2000 (Win2K) and later; review RFCs 3244 and 1964 at the same site for more information about Kerberos specifics and Microsoft extensions.

Kerberos provides a number of advantages over Microsoft's older authentication protocols:

- The burden of authentication is placed on the client. Older protocols placed the burden on the server, creating a less distributed and less scalable architecture.
- Authentication between clients and servers is mutual, meaning both are assured of the other's identity. Older protocols focused primarily on the server being assured of the client's identity.
- Kerberos uses strong encryption technologies and timestamps, making it very difficult to compromise. Older protocols relied primarily on well-known encryption hashes and simple obfuscation for protection, which resulted in the success of cracking utilities such as L0phtCrack.

Roles in Kerberos

Kerberos is named for the dog in Greek mythology that guarded the gates to the underworld. Like the three-headed mythical dog, the Kerberos protocol identifies three primary roles for authentication participants:

- The Key Distribution Center (KDC) is responsible for issuing authentication tickets to clients.
- Clients are computers that attempt to access resources located on a server.
- Servers are computers that contain resources being accessed by clients.

Any discussion of Kerberos in the Microsoft world can become confusing because computers often hold all three roles. For example, all domain controllers in an Active Directory (AD) domain are KDCs. They might also be file servers, making them Kerberos servers. You might also log on to a domain controller's console and attempt to access files on another server, in which case the domain controller becomes a Kerberos client. So in any discussion of Kerberos, it's important to consider what role each participant is playing in a single particular authentication transaction.

The Kerberos Process

Kerberos defines two basic processes, which are functionally very similar to one another. The first process occurs whenever a security principal, such as a user or computer, logs on to an AD domain. The second process occurs whenever a logged-on principal attempts to access a resource located on another computer. For information about how encryption relates to Kerberos, see the sidebar "Encryption and Kerberos.")

Remember that both users and computers have accounts in an AD domain, and that both are considered individual security principals. For example, the logon process occurs twice on every Windows XP Professional computer: Once when the computer starts, and again when a user logs on to the computer's console.

Encryption and Kerberos

Understanding how digital encryption works is a key to understanding Kerberos. By default, Kerberos uses *shared secret* encryption, which means that both the sender and recipient know a secret phrase or password that is used as an encryption key. Generally, this shared secret is the recipient's password, something that both the recipient—whether a computer or a user—and the KDC both know.

In Win2K and later, Microsoft extends the standard Kerberos protocol to support Public Key Infrastructure (PKI) and digital certificates. In this scenario, the security principals and the KDC don't share a secret. Instead, they use each others' public keys—obtained from a certificate authority (CA) server—to encrypt data, and their own private keys—stored in the computer or on a smart card—to decrypt data.

Logging on and Obtaining a TGT

When a security principal authenticates to a domain, the principal has to prove its identity to the domain controller. This is done by creating a special packet of data called an *authenticator,* then encrypting the packet and sending it to the KDC. A clear-text (unencrypted) version of the authenticator is also sent. How the authenticator is encrypted depends on the encryption scenario in use.

Usually, the principal's hashed password is used as the encryption key. For example, if the principal is a user, the user would type his or her user name and password into the logon dialog box on the client computer. The client computer hashes the password and uses the result to encrypt the authenticator. The domain controller receives the authenticator and looks up the user account in AD to retrieve its copy of the hashed password. If the hashed password from AD can be used to decrypt the authenticator, then the user must have typed the correct password. If the user hadn't, the password stored in AD wouldn't be able to decrypt the authenticator.

A *hash* is the result of a one-way encryption algorithm, meaning it is impossible to start with the encrypted result and ever arrive at the original unencrypted password. Domain controllers only store hashed passwords for principals; when passwords are changed, they are immediately re-hashed and stored.

In a PKI scenario, the user's client computer would use a smart card containing the user's private key to encrypt the authenticator. Smart cards generally require the user to provide a PIN or some other identifying factor (such as a thumbprint) to unlock the card. Once the domain controller received the authenticator, it would obtain the user's public certificate from AD or a CA, and try to use the public key to decrypt the authenticator. If the decryption works, the user's identity would be verified.

Technically, anyone could intercept the authenticator and use the user's public key to decrypt it. Because the authenticator doesn't contain any sensitive data, that's fine. An intruder could not modify and re-encrypt the authenticator without the user's private key, which is safely stored in the smart card.

The authenticator includes a timestamp and a unique identifying number. Domain controllers will reject any authenticators that are too old (5 minutes, by default) or that the domain controller has already processed. This behavior reduces the likelihood that an intruder could capture an authenticator off of the network and replay it against the domain controller at a later time (presumably after modifying it in some way).

Once the domain controller is convinced of the principal's identity, it creates a *ticket-granting* ticket. The TGT contains a unique, temporary encryption key that the domain controller makes up on the spot. By using this *session key* for future communications*,* the user's regular encryption key won't be overexposed on the network. One copy of the TGT is encrypted with the domain controller's private encryption key, and can be decrypted only by that domain controller. That copy of the TGT, along with an unencrypted copy, are encrypted using the principal's shared secret (or, in a PKI scenario, with the principal's public key), then sent back to the principal.

When the principal receives the TGT, the principal decrypts it—using either its shared secret or private key—and store it in memory. So now the principal has two important pieces of information:

- An unencrypted TGT containing the session key to be used for future KDC communications.
- An encrypted TGT that can only be decrypted by the KDC.

The TGT is also marked with an expiration time (8 hours, by default) and a maximum lifetime (7 days, by default). When the TGT expires, the principal can ask the KDC to renew the TGT for up to the maximum lifetime. Expiration times and maximum lifetimes ensure that session keys aren't used for such a long time that an intruder might be able to compromise them. Once the two copies of the TGT are in RAM, the logon process is complete.

TGTs are stored in a special non-swappable area of volatile RAM. In other words, the TGTs can't be written to disk under any circumstances and will be lost if the client computer loses power. User TGTs are also destroyed when the user logs off.

Accessing Server Resources

By now, the principal has a TGT from the KDC. That TGT is a key part of further authentication operations, such as attempting to access resources on a server. Note that Kerberos doesn't directly provide *authorization,* which determines *which* resources a principal may access. Kerberos simply provides *authentication*, which tells the server who the principal is. The server can then use other technologies—such as NTFS file access control lists (ACLs)—to determine what the principal is allowed to do.

When a client needs to access resources on a server, it contacts the KDC and requests a *ticket* for that server. The ticket request is accompanied by the KDC-encrypted copy of the TGT from the client's memory, and the request itself is encrypted with the session key from the unencrypted copy of the TGT. The result:

- The KDC can decrypt its half of the TGT to retrieve the session key, which ensures that the session key wasn't tampered with.

- The KDC uses the now-decrypted session key to decrypt the ticket request. This re-validates the client's identity; only this client had a copy of the session key, and it matches the one contained in the TGT.
- The KDC didn't have to persist any information about the client because the client is presenting all the necessary information—in the request and TGT—to revalidate the client's identity to the KDC.

Satisfied with the client's identity, the KDC generates a ticket. This ticket contains a new, unique session encryption key. One copy of the ticket is encrypted with the session key shared between the client and KDC. Another copy is encrypted with a secret shared between the KDC and the server to which the client is requesting access. Both copies are sent back to the client.

The client can decrypt one half of the ticket to obtain a session key, which it uses to encrypt an access request packet. The client sends this encrypted packet to the server, along with the other half of the server ticket—the half that only the server can decrypt.

When the server receives the request, it decrypts its half of the server ticket, revealing a session key. The server then uses the session key to decrypt the actual request. The result:

- The server is assured that a KDC validated the client's identity because only the KDC would have generated a valid server ticket that the server could decrypt.
- The client is assured of the server's identity because only the proper server would have been able to decrypt the server ticket.

The server can now process the access request and provide the client with the resources requested.

Advanced Kerberos Concepts

Kerberos has an additional advantage over Microsoft's older authentication protocols: delegation and proxying. These techniques essentially allow a background service or application to access resources using a user's security credentials, ensuring that the service or application can only access resources that the end users would normally be allowed to access on their own. However, both delegation and proxying require services and applications that have been specially programmed to deal with Kerberos' advanced features, making a discussion of how these features work beyond the scope of tip.

Q.11: How do I configure Kerberos?

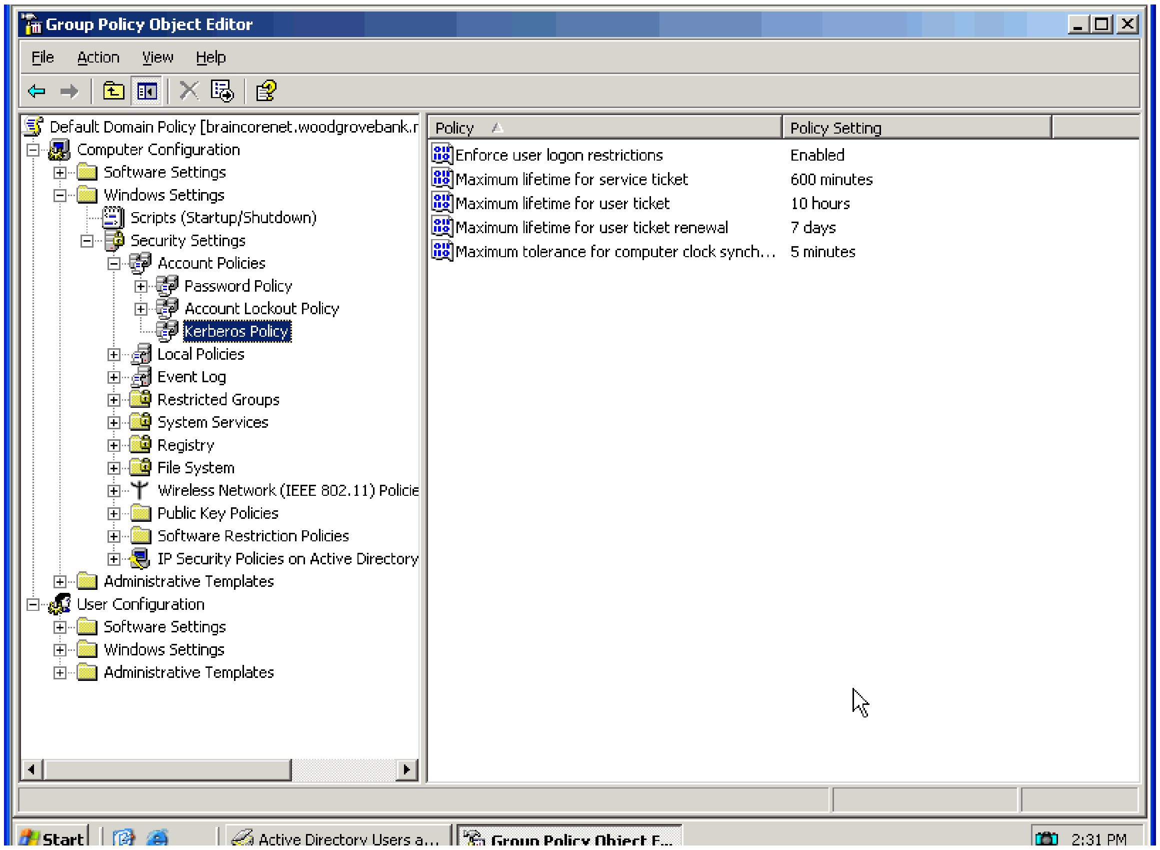
A: Kerberos is configured entirely within Active Directory (AD) and is a part of the Default Domain Policy Group Policy object in every domain. As an administrator, you can modify several Kerberos' key configuration settings.

☞ In general, you shouldn't need to modify the default Kerberos configuration settings. Microsoft came up with the default settings based on a typical work environment, and it's rare for them to be unsuitable or wrong. Be sure that you understand exactly what each setting does and how it impacts your domain before making any changes.

To modify the Kerberos policies for a domain, follow these steps:

> The screen shots that follow and the default setting values are from Windows Server 2003—the steps are identical in Windows 2000 (Win2K) domains.

1. Open Active Directory Users and Computers.
2. Right-click the domain, and select Properties.
3. Select the Group Policy tab.
4. Select Default Domain Policy from the list, and click Edit.
5. Locate the Kerberos settings, shown in Figure 11.1.



Policy	Policy Setting
Enforce user logon restrictions	Enabled
Maximum lifetime for service ticket	600 minutes
Maximum lifetime for user ticket	10 hours
Maximum lifetime for user ticket renewal	7 days
Maximum tolerance for computer clock synch...	5 minutes

Figure 11.1: Kerberos policies in the Default Domain Policy Group Policy object.

6. Modify the settings as appropriate.
7. Close the Group Policy Object Editor.
8. Close the Properties dialog box.

Your changes will be replicated throughout the domain.

Kerberos Settings

Windows contains only five Kerberos settings:

- **Enforce user logon restrictions**—Enabled by default, this setting determines whether the Key Distribution Center (KDC) validates every request for a session ticket against the user's rights. Enabling this policy provides better security because changes to user rights will be effective for the user's next ticket request. However, the validation step does require a bit of extra processing on the part of the KDC.
- **Maximum lifetime for service ticket**—Service tickets are those requested by Windows background services to act on behalf of a user. The default 600-minute lifetime is generally sufficient and prevents a service from being able to impersonate a user for an unreasonably long period of time. Shortening this value places increased load on domain controllers, as services will have to contact domain controllers more frequently to obtain new tickets.
- **Maximum lifetime for user ticket**—User tickets are those requested by users to access resources. The default setting, 10 hours, covers a typical workday and allows a user to utilize tickets for the entire day. Lengthening this value might overexpose the ticket and leave it more vulnerable to compromise; shortening this value might place an undue burden on domain controllers.
- **Maximum lifetime for user ticket renewal**—When a ticket's maximum lifetime expires, a client computer can renew the ticket. Renewals don't require a new session key, thereby saving a bit of processing on the KDC. The default setting for this value is 7 days, ensuring that ticket session keys don't last longer than a week.
- **Maximum tolerance for computer clock synchronization**—This value defaults to 5 minutes, meaning a client computer's clock can as many as 5 minutes ahead or behind the KDC's own clock. The default value should be fine for most environments. Lengthening this value could give intruders extra time to attempt to compromise a ticket request captured from the network; if your environment has network latencies or time sync problems that seem to necessitate lengthening this setting, you are better off correcting those time or latency problems than compromising Kerberos security.

For more information about how time affects Kerberos, see Question 10.

Q.12: How can I ensure that Kerberos is working?

A: Troubleshooting Kerberos can be difficult; Windows pretty much does everything behind the scenes, and doesn't give an administrator a terrific amount of control over what Kerberos is doing. There are, however, some steps you can take to see what Kerberos is up to.

You'll rely heavily on utilities included with the Windows resource kits, such as *The Microsoft Windows 2000 Server Resource Kit*. If you don't have these kits, you'll need to obtain a copy in order to follow along. Some of the utilities can be downloaded for free from Microsoft's Web site.

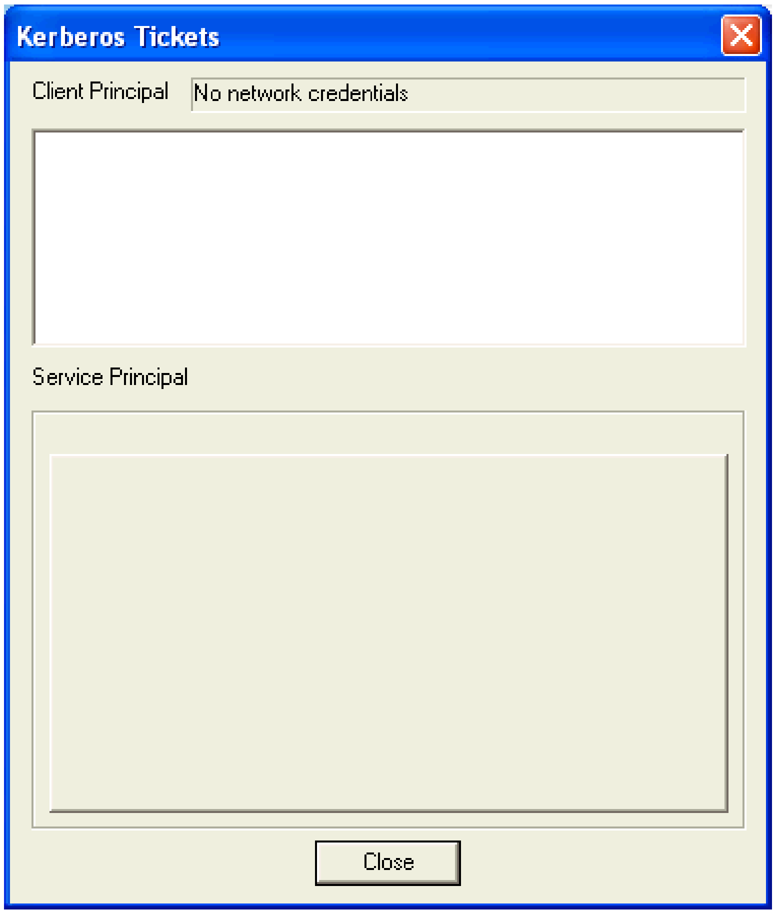
Checking for the TGT

After logging on to a workstation, first ensure that the computer obtained a ticket-granting ticket (TGT) from a Key Distribution Center (KDC—domain controller). I prefer to use KerbTray to check this.

For more information about how your computer obtains a TGT, refer to Question 10.

You can download KerbTray for free from http://www.Microsoft.com/windows2000/techinfo/reskit/tools/existing/kerbtray-o.asp.

After downloading and running KerbTray, you'll notice an additional icon in the system tray of the taskbar. Simply double-click the icon to see your computer's existing tickets, including the TGT. If the window appears empty, as Figure 12.1 shows, your computer isn't authenticating to an Active Directory (AD) domain (and might not, in fact, even be a member of a domain).



Client Principal: No network credentials

Figure 12.1: KerbTray won't display credentials if your computer isn't logged on to an AD domain.

Managing Tickets

Microsoft also provides a command-line utility, Klist.exe, in the resource kit (and as a download from http://www.Microsoft.com/windows2000/techinfo/reskit/tools/existing/klist-o.asp). You can use Klist not only to view tickets and your TGT (which KerbTray displays), but also to delete tickets.

> If you delete tickets, especially the TGT, you might not be able to authenticate to network resources until you log off and back on again.

Use Klist as follows:

- klist tgt—Displays information about the TGT in your system, including the domain that issued the TGT and how long it's good for. A system without a TGT can't access domain resources.
- klist tickets—Displays all cached tickets, including which server each ticket is for and how long each ticket is good. If you've been unsuccessfully attempting to access a particular resource and don't have a ticket for it, then you've found the problem.

When you check a TGT, you're looking for a few specific pieces of information in Klist's output:

- DomainName—The domain name should match the domain name that you logged on to.
- KeyExpirationTime—This time should be some date and time in the future.
- StartTime—This time should be some date and time in the past, even if it's the recent past.
- EndTime—Again a date and time in the future.
- TimeSkew—This date and time should be in the future, and should be greater than the EndTime.

If any of these values look wrong, then the KDC service on the domain controller that authenticated you might be having problems. Check the domain controller's event logs for error messages that provide additional clues.

Similarly, when checking a ticket, you're looking for:

- Server—This server should be the fully-qualified name of the resource you're trying to access.
- EndTime—Should be in the future by at least an hour or so.

If the EndTime value in particular is off, try deleting the ticket and letting Windows acquire a new one from the KDC.

Troubleshooting Steps

Having trouble accessing resources in the domain? Try these steps to narrow the problem:

1. Run Klist tgt to check your TGT. If your TGT is expired or not present, log off and back on again to repeat. If that doesn't fix it, your computer either isn't in the domain or its domain credentials need to be reset. See How-To: How to Reset a Computer's Domain Account for corrective steps.
2. Run Klist tickets to see if you have a ticket for the resource you're trying to access. If you don't, try to access another resource. If that works, then the resource giving you problems might not be in the domain. If you can't access any resources, but have a TGT, then the KDC service on the domain controller that issued your TGT might have failed. Check the service on the domain controller, then log off and back on again.
3. If you have a valid TGT and a valid ticket, use Klist purge to delete the ticket associated with the resource you're having problems with. Then try to access the resource again. Windows should obtain a new ticket from the KDC; if it doesn't, refer back to step 2. If you do get a new ticket (verify with Klist tickets) but still can't access the resource, you need to check the resource server for problems, such as a failure in the NetLogon service or an out-of-sync domain computer account.

Q.13: How are Relative Identifiers allocated?

A: Relative Identifiers (RIDs) are used to uniquely identify each object within a domain. In any Active Directory (AD) domain, each domain controller has the ability to create new objects—users, computers, groups, and so forth. Each of these new objects needs a unique ID number to avoid conflict with other new objects being created at any given time by other domain controllers in the domain.

The unique ID numbers given to each domain object are actually a combination of a domain ID and a RID; RIDs can be duplicated across domains because the combination of domain and RID will always be unique. The uniqueness of the domain ID is ensured by the forest-wide domain naming master.

The RID Master

In order to ensure that domain controllers don't duplicate ID numbers, AD includes a special Flexible Single Master Operations (FSMO) role in each domain, called the *RID master*. The RID master's job is to allocate each domain controller with a unique range of RIDs. Because all RIDs stem from this single source and the RID master doesn't issue overlapping pools to different domain controllers, each domain controller has a unique range of "spare" ID numbers to use when creating new objects.

As part of its role in ensuring uniqueness for each AD object, the RID master is also responsible for removing the entries for domain objects that are moved to another domain. However, you should note that the RID from the removed object is never reused in the domain.

For more information about the RID master FSMO role, see Question 1.

SID Construction

The unique number assigned to each domain object is called a Security Identifier (SID). A typical SID looks like this:

S-1-5-21-917267712-1342860078-1792151419-500

- S designates this identifier as a SID
- 1 indicates the revision level of the SID construction scheme
- 5 represents the identifier authority
- Everything else is the SID itself; the combination of domain ID and RID.

RID Management

You can't directly affect the allocation of RIDs except through a few documented workarounds to specific operating system (OS) problems. You can view certain RID attributes directly in AD.

> It is possible for a domain controller to use up its allocated RID pool more quickly than it can request a new one. For example, if you're migrating thousands of users to a domain controller that has poor connectivity to the RID master, the domain controller might run out of RIDs. For more information about this problem, see the Microsoft article "RID Pool Allocation and Sizing Changes in Windows 2000 SP4."

AD contains several attributes that contain information about RIDs; these attributes, in fact, are the sources that DisplayRID queries for its output. The major attributes are:

- FsmoRoleOwner—Contains the fully qualified domain name of the current holder of the RID master role.
- RidAvailablePool—Defines the number of security principals that the domain can contain (a fixed value currently just over 1 billion), and the number of RIDs that have been allocated already.
- RidAllocationPool—Defines the current pool for a domain controller, and its next pool.
- RidNextRid—The next RID that will be used on the domain controller.
- RidPreviousAllocationPool—The current pool of RIDs used to create new SIDs; this value includes the value of RidNextRid.
- RidUsedPool and NextRid—Unused attributes that are still defined in AD.

> The values of these attributes will differ from domain controller to domain controller.

Q.14: Why do deleted Active Directory objects sometimes reappear?

A: It's one of the strangest things that can happen in Active Directory (AD): You delete an object, such as a user, group, or organizational unit (OU), then a few minutes—or even days—later the object mysteriously reappears. Generally, deleting the object a second time makes it "stick," but reappearing objects—especially users and groups—can have serious security implications. After all, if a user account reappears, it can be used to access domain resources. So where do deleted objects come back from, and how can you prevent it from happening in the future?

AD Replication

The key to this puzzle lies within AD replication. Remember that every time an object is changed, its Update Sequence Number (USN) is incremented. When the object is replicated, it comes from the domain controller that has the highest USN for the object, and the object overwrites any older copies on other domain controllers.

Technically, object *attributes* have USNs; for more information about how AD replication operates, see Question 7.

When you delete an object from AD, it doesn't go away immediately. Instead, the object is *tombstoned,* meaning AD places a special marker on it indicating that the object is no longer active. The tombstone itself has a USN and is replicated to all domain controllers in the domain. 60 days after the tombstone, all domain controllers independently remove the object from their copy of the AD database.

There are a few situations in which the tombstone can be removed, and even situations in which a tombstoned and deleted object can mysteriously reappear. You'll need to review these possible situations to determine which occurred in your environment, and take steps to prevent it from happening again.

Offline Domain Controllers

Some companies decide to keep a spare domain controller off the network as a last-resort backup, in case all of their active domain controllers somehow become corrupted or fail in some fashion. Some companies also ship new domain controllers to new branch offices, and unforeseen circumstances keep the domain controller offline for longer than intended. Unfortunately, keeping an offline domain controller can cause deleted objects to return from the dead. Here's what happens:

- A domain controller is taken offline. Obviously, it is no longer replicating with the other domain controllers.
- An AD object is deleted by an administrator. This causes the object to be tombstoned for 60 days.
- If the offline domain controller is returned to the network, the results will depend on exactly when it is returned:

- If the domain controller is returned within the tombstone lifetime, then the domain controller will replicate the tombstone and the object will not reappear.
- If the domain controller is returned after the tombstone lifetime, then the formerly offline domain controller will have a copy of an object that doesn't exist on other domain controllers. Thus, most of the domain's domain controllers don't have the object, which means they don't have a USN for it. Any USN is better than none, so the formerly offline domain controller will replicate the object within AD and all other domain controllers will be happy to accept it. The object reappears.

The same scenario can occur if you restore an AD backup that's older than 60 days. For that reason, Windows' tools will try to prevent you from restoring a backup that is more than 60 days old. There are documented workarounds that can enable you to bypass this protective feature, but you shouldn't try.

How can you prevent this from happening? Take some basic precautions. First, make a backup of AD every single day. It only takes a few minutes and ensures that you have a recent backup to work from at all times. Second, never keep domain controllers offline for more than a few days. If you have to have a domain controller offline for a couple of weeks, ship a more recent AD backup that can be used to update the domain controller's database.

In Windows Server 2003, you can write the AD database backup to removable media, such as a CD-ROM. This media can be used to initialize a new domain controller's AD database. It's an ideal way to deploy domain controllers to remote offices: Ship a non-domain controller Windows Server 2003 computer to the office, use Dcpromo to install AD, and initialize AD from removable media. Doing so will prevent a heavy WAN load for initial replication while ensuring a recent copy of AD is placed onto the new domain controller.

Authoritative Restores

Here's another scenario: Imagine you've got an OU named Sales, which contains a user named Maria and a user named Pete. You make a backup of the AD database on Monday morning. On Tuesday morning, Pete leaves the company, so you delete his AD account. Tuesday afternoon, someone accidentally deletes Maria's user account. To correct the problem, you decide to perform an authoritative restore of the Sales OU. Maria's account comes back, but so does Pete's.

This mistake seems like a fairly obvious gaffe: You could have performed an authoritative restore of just Maria's user account. However, it's a common situation. When you perform an authoritative restore, AD increments the USNs of all restored objects by several thousand, making them the highest USNs in the domain. This forces AD to accept the restored objects—which in this case have no tombstone—as authoritative, and they are replicated throughout the domain.

Authoritative restores from backups older than 7 days can cause trust and computer account problems. The passwords for trusts and computer accounts are automatically changed about every 7 days.

Had you performed a non-authoritative restore, neither object would have reappeared. That's because the USN on the backup tape was older than the current USN held by the domain's domain controllers. When you restored the objects, they would briefly reappear on the domain controller that you restored them to, but that domain controller would quickly re-replicate the higher-USN version of the objects from other domain controllers, and the objects would again be tombstoned.

The moral is to restore as few objects as necessary when performing an authoritative restore. When restoring an entire OU, you might not realize that other objects contained within the OU will be restored, so you'll also need to perform a follow-up inspection after your restore to see what's back. For example, suppose someone had deleted the Sales OU on Tuesday afternoon, rather than deleting Maria's user account. Obviously, you'll need to perform an authoritative restore of the entire OU at this point, and Pete's account is going to come back whether you like it or not. A quick follow-up inspection of the OU after the restore will let you re-delete Pete's account and correct the problem.

Windows Server 2003 provides a means for restoring accidentally deleted objects, effectively removing their tombstone. At least one domain controller in the domain must be running Windows Server 2003 to enable this functionality. A new user right, *Reanimate Tombstone*, provides the necessary permissions to the Domain Administrators group by default. Unfortunately, there's currently no user interface for using this feature; you'll need a custom utility to take advantage of it for now.

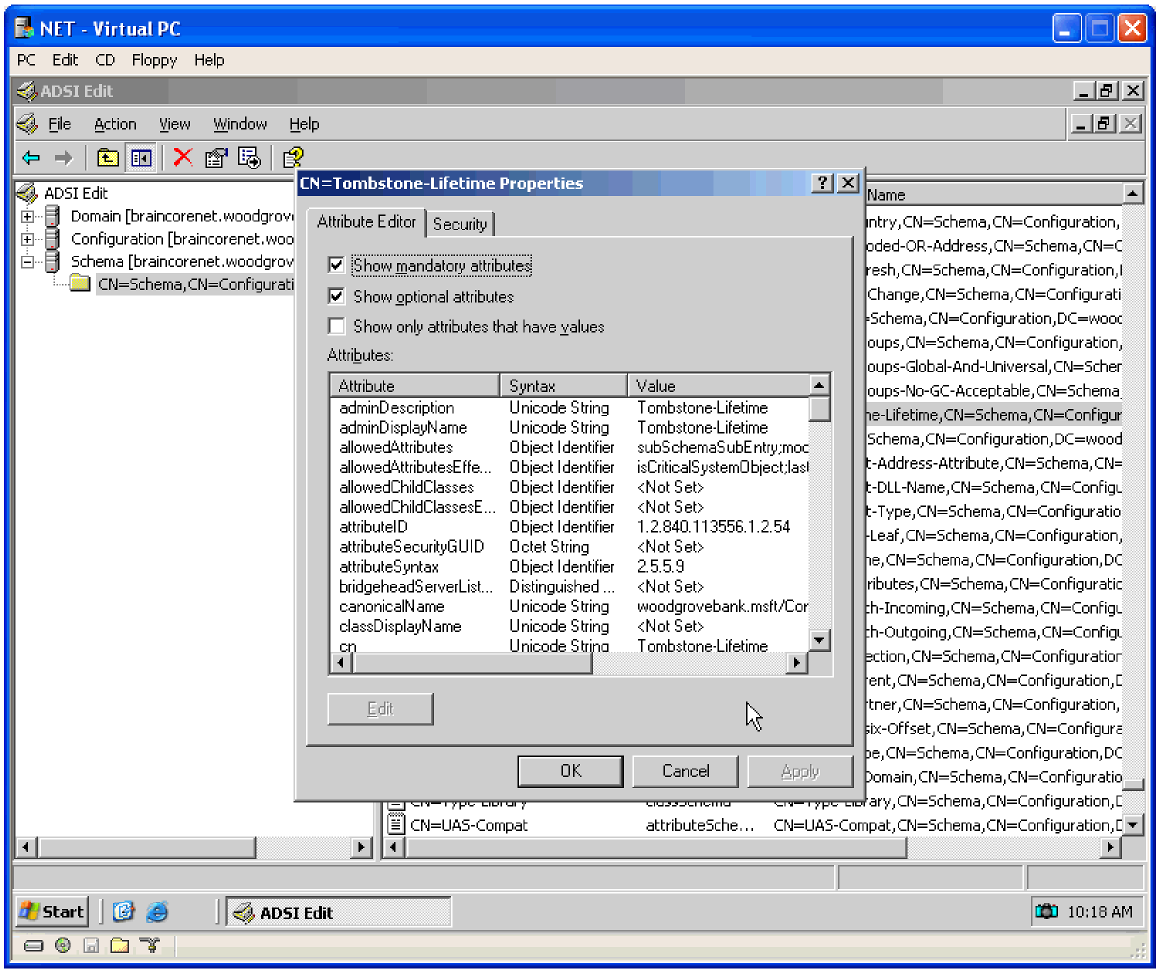
Other Tombstone Tricks

Windows 2000 (Win2K) Service Pack 3 (and some post-SP2 hotfixes) introduced a new feature known as Strict Replication Consistency. This feature can cause additional problems for out-of-date domain controllers, causing them to log event ID 1084 in the System event log.

The Microsoft article "Lingering Objects Prevent Active Directory Replication from Occurring" provides more information about this problem and describes how Strict Replication Consistency works.

You can also modify the tombstone lifetime in AD. The default 60-day value is designed to accommodate domain controllers that might be offline for an extended period of time and to ensure that backups are useful for a good, long while. However, you can reduce the value to as little as 2 days or even increase it. To do so, use the ADSI Edit snap-in, which Figure 14.1 shows, to edit the tombstoneLifetime attribute.

There's no default console that contains the ADSI Edit snap-in; you'll need to install the support tools from the Windows CD-ROM and run Adsiedit.msc.


NET - Virtual PC
PC Edit CD Floppy Help
ADSI Edit
File Action View Window Help
ADSI Edit
Domain [braincorenet.woodgrov
Configuration [braincorenet.woo
Schema [braincorenet.woodgrov
CN=Schema,CN=Configurati
Start
ADSI Edit
10:18 AM

CN=Tombstone-Lifetime Properties

Attribute Editor | Security

- [✓] Show mandatory attributes
- [✓] Show optional attributes
- [ ] Show only attributes that have values

Attributes:

Attribute	Syntax	Value
adminDescription	Unicode String	Tombstone-Lifetime
adminDisplayName	Unicode String	Tombstone-Lifetime
allowedAttributes	Object Identifier	subSchemaSubEntry;moc
allowedAttributesEffe...	Object Identifier	isCriticalSystemObject;last
allowedChildClasses	Object Identifier	<Not Set>
allowedChildClassesE...	Object Identifier	<Not Set>
attributeID	Object Identifier	1.2.840.113556.1.2.54
attributeSecurityGUID	Octet String	<Not Set>
attributeSyntax	Object Identifier	2.5.5.9
bridgeheadServerList...	Distinguished ...	<Not Set>
canonicalName	Unicode String	woodgrovebank.msft/Cor
classDisplayName	Unicode String	<Not Set>
cn	Unicode String	Tombstone-Lifetime

Edit

OK Cancel Apply

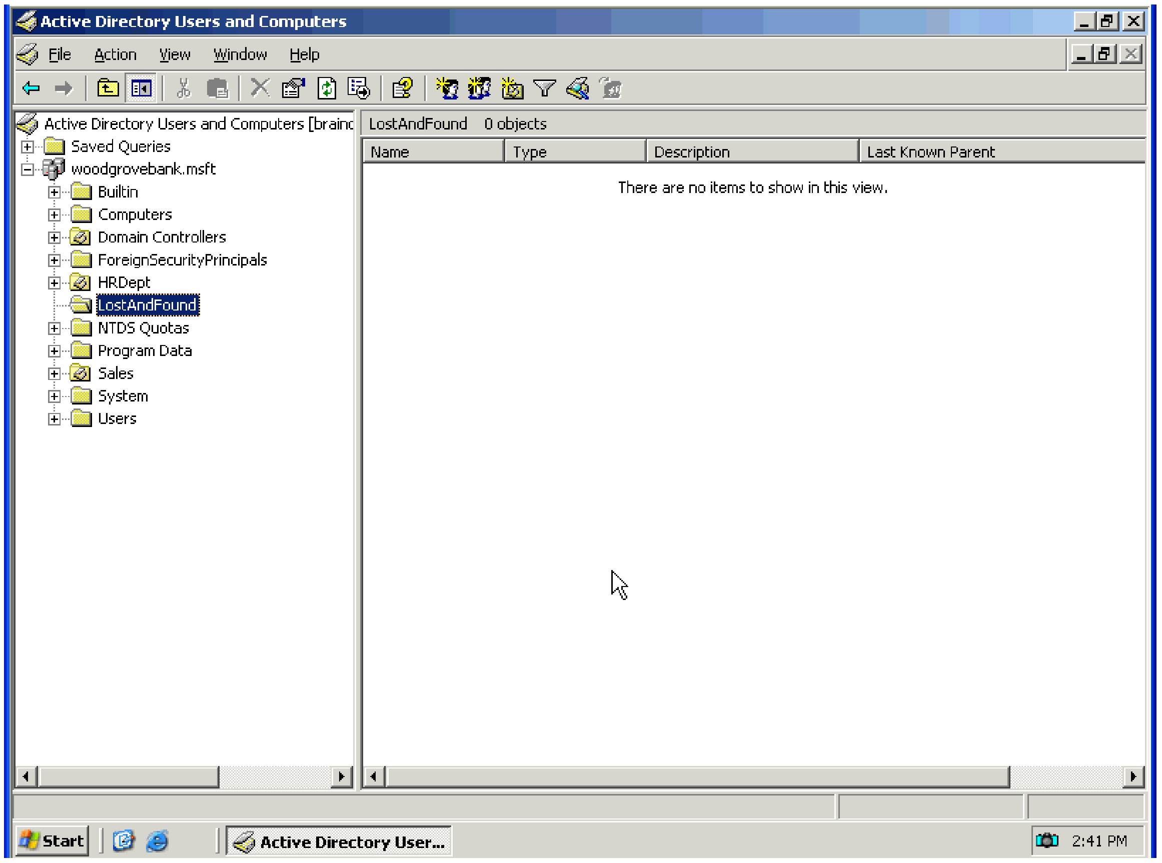
CN=UAS-Compat attributeSche... CN=UAS-Compat,CN=Schema,CN=Configuration,D

Figure 14.1: Using the ADSI Edit console to modify the tombstoneLifetime attribute.

Few administrators prefer to shorten the tombstone lifetime; many, however, lengthen it to accommodate longer domain controller absences or to extend the useful life of AD backups. In most environments, 60 days is more than sufficient. 30 days is a reasonable minimum if you specifically want deleted objects removed more quickly and aren't concerned about backup lifetime. Anything longer than 90 days is probably excessive, and will keep deleted objects on your domain controllers a lot longer than they need to be.

Q.15: Why do objects sometimes appear in the Lost and Found folder in Active Directory Users and Computers?

A: Many administrators don't even notice the Lost and Found folder. It's easy to miss, especially because Active Directory Users and Computers only displays it when you configure the snap-in to display advanced options. Open up your copy of Active Directory Users and Computers, and make sure that Advanced is selected in the View menu. Ideally, your Lost and Found folder will look like the one in Figure 15.1—empty. That means that all is well.



There are no items to show in this view.

Figure 15.1: An empty Lost and Found folder.

But if Lost and Found does contain objects, how did they get there? And what should you do about them?

A Home for Orphans

Essentially, the Lost and Found folder is a home for orphaned Active Directory (AD) objects. Objects usually become orphans through AD replication *convergence*. Every AD domain controller contains a complete read/write copy of the domain database. That means that it is possible for two administrators to make conflicting changes to AD at the same time. Eventually, AD replication will converge, resolving the discrepancy. The time it takes to do so is referred to as *replication latency*.

Most of the time, AD doesn't actually need to deal with what appears to be a conflicting change. For example, suppose one administrator changes a user's password, while another changes the user's name. AD replicates each attribute individually, so there's no conflict, even though two administrators made changes to the same user.

However, some types of conflicts can't be easily handled. For example, suppose that one administrator moved a user into the Sales organizational unit (OU), at the same time another administrator deleted the Sales OU on another domain controller. When convergence occurs, where does the user account wind up? In Lost and Found.

Handling Found Objects

When objects appear in Lost and Found, many administrators' first instinct is to delete them, thinking they're some kind of bogus object reference. Don't! They're real AD objects, and you simply need to move them into a regular container or OU.

Other Lost and Found Information

In some environments, administrators use copies of the Ntdsutil utility that they obtained from prerelease copies of Windows 2000 (Win2K). In the article "How to Troubleshoot an 'Internal Error' Error Message During the Replication Phase of Dcpromo," Microsoft documents a problem with prerelease copies of this utility in which using it to perform an authoritative restore would incorrectly increment the internal version number of the Lost and Found container, causing an internal Windows error. Obviously, stay away from prerelease utilities!

Lost and Found has one legitimate use that doesn't indicate a problem or a replication issue—moving objects between domains. When you use Microsoft's MoveTree utility to move objects between domains, the utility first moves objects into the Lost and Found folder, they're then copied to the destination domain and removed from Lost and Found. If MoveTree fails to work correctly, you might find objects still lingering in Lost and Found. Further, if you try to use MoveTree to move objects such as computer accounts, which it can't handle, they'll wind up in a subfolder under Lost and Found.