Active Directory Object Recovery Using the Recycle Bin

blog.netwrix.com/2021/11/30/active-directory-object-recovery-recycle-bin

Kevin Joyce

The Active Directory Recycle Bin enables users to recover deleted Active Directory objects without having to restore them from backup, restart Active Directory Domain Services or reboot domain controllers (DCs).

Let's explore exactly how object recovery works with the Recycle Bin and then discuss its limitations.
Handpicked related content:
How to Restore Active Directory Users

Active Directory Object Recovery with the AD Recycle Bin

If the AD Recycle Bin is enabled, when an object in deleted, the majority of its attributes are preserved for a period of time to facilitate restoring the object if needed. During this period, the object is in the **deleted object** state. (This time period is defined in the msDS-DeletedObjectLifetime attribute. By default, its value is the value of the tombstoneLifetime attribute. If the value of the msDS-deletedObjectLifetime attribute is null or the attribute simply doesn't exist, its value is interpreted to be the value of the tombstoneLifetime attribute. If there's also no tombstoneLifetime value, both values default to 60 days.)
Handpicked related content:

[On-demand Webinar] Active Directory 101: Install & Configure AD Domain Services

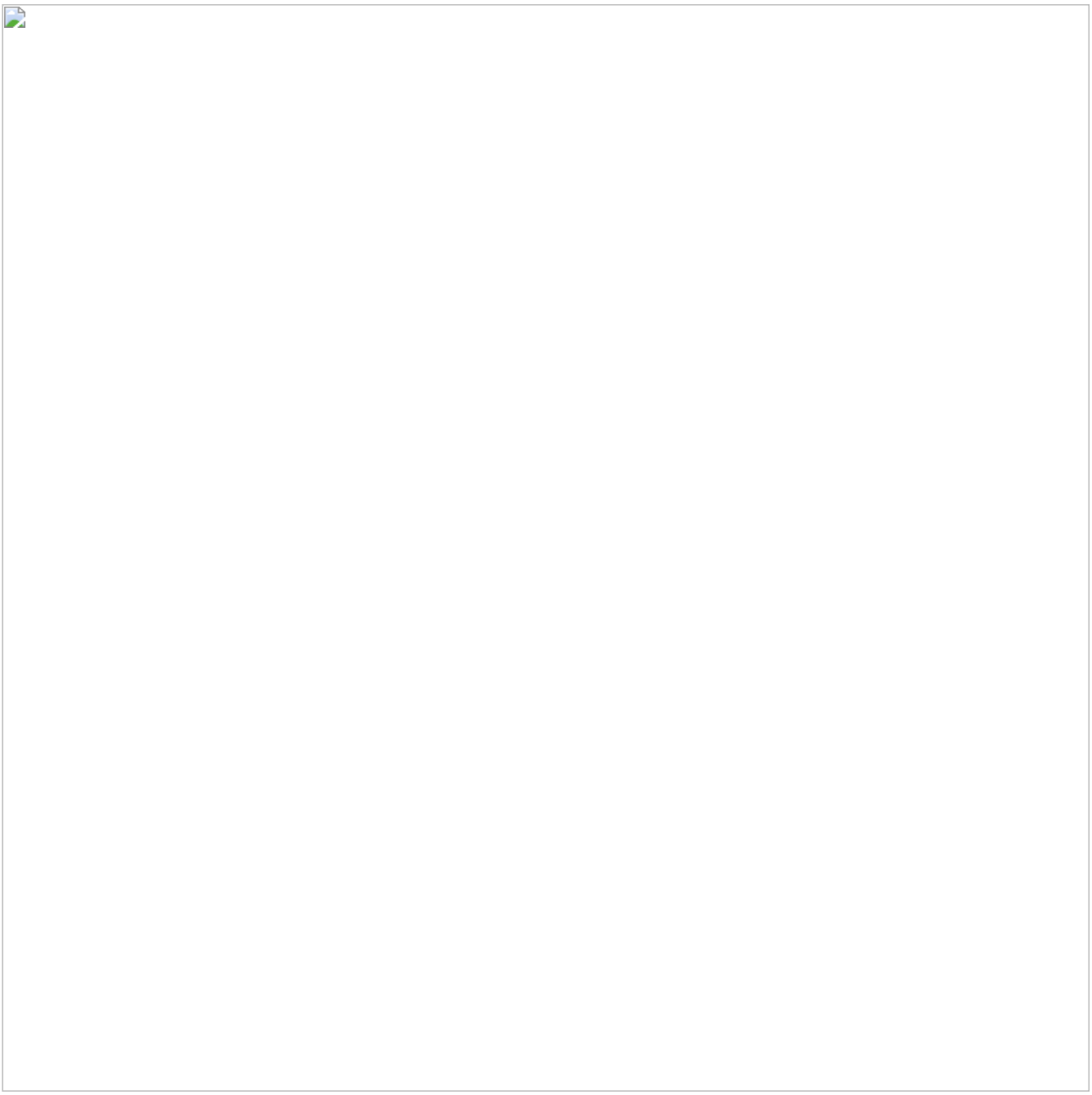
Once the object's time in a deleted object state is up, the object becomes a **recycled object**. A recycled object looks suspiciously like a tombstone with an isRecycled attribute slapped on and set to TRUE. Like a tombstone, the majority of its attributes are removed and it persists in Active Directory for the time period specified by the tombstoneLifetime attribute. Then it is cleaned up by Active Directory's garbage collection.

The lifecycle of an object deleted with the Recycle Bin enabled looks like this:

How an Object Changes when It Enters the Recycle Bin

While the Recycle Bin preserves more object attributes than a tombstone, a restored object is not identical to the original object. Let’s see how. Here is a user account that I am planning to delete:

Here is the object in the deleted object state in the Recycle Bin:



While the majority of the object's attributes are retained, there are some important differences:

- **The object has been moved.** The object has been moved into the partition's Deleted Objects container.
- **The object has been renamed.** The object's name has been updated using the *Common-Name* DEL:*Object-Guid.*
- **The object possesses some new attributes.** The **isDeleted** attribute has a value of TRUE and the **lastKnownParent** attribute is populated. A new attribute, **msDS-LastKnownRDN**, is populated with the object's last known relative distinguished name (this attribute allows the Recycle Bin to properly reset an object's RDN during its restoration, even if the object's renaming resulted in the truncation of the original RDN).
- **Two attributes have been removed.** Two attributes, **objectCategory** and **sAMAccountType,** are always removed from an object when it is deleted. If the object is recovered, the objectCategory value is automatically set to the most specific value in the object's **objectClass** attribute and the sAMAccountType value is calculated from the value of either the **userAccountControl** (for user objects) or **groupType** attribute (for group objects).

Keen-eyed readers might also notice that the **manager** and **memberOf** attributes are also missing from my screenshot. They're actually just hiding. Both these attributes are **link-valued** (i.e., they contain references to other objects) and tool I used (LDP) doesn't return deactivated links unless the cleverly-named Return Deactivated Links control has been set. If I had enabled that control, then the attributes and their values would have been visible in my screenshot, but I would have missed out on this teachable moment.

How to Recover an Object from the AD Recycle Bin

Prior to Windows Server 2012, restoring an object from the AD Recycle Bin required using an LDAP tool or PowerShell to list all deleted objects, sifting through a long list to find the desired object, and using another PowerShell command to restore it. It was a good thing the AD Recycle Bin was so useful because it was not exactly fun to use!

Now, Recycle Bin functionality is available in the Active Directory Administrative Center:

As you can see, you can quickly find the deleted object you're interested in by using the search filters.

To restore an object, simply click **Restore** in the Tasks list on the right side of the window. Here's what the restored object looks like:

Drawbacks to the Active Directory Recycle Bin

While the Recycle Bin dramatically simplifies object recovery, we have seen a couple of limitations: Objects are kept for only a fairly short period of time and some of their attributes are lost. There are a couple of additional drawbacks to the Recycle Bin:

- **Enabling the Active Directory Recycle Bin involves a schema change.** Therefore, once you turn the Recycle Bin on you can't turn it off without a full-forest recovery.
- **Active Directory is going to be a little bigger.** After enabling the AD Recycle Bin, deleted objects will retain far more of their attributes and persist longer than tombstones. As a result, Active Directory will likely use a little more space than it did before.
- **Enabling the Recycle Bin deletes all tombstones.** The most impactful consequence of enabling the Recycle Bin is that all tombstone objects in the forest will immediately cease to exist. Many admins have learned about this consequence the hard way.

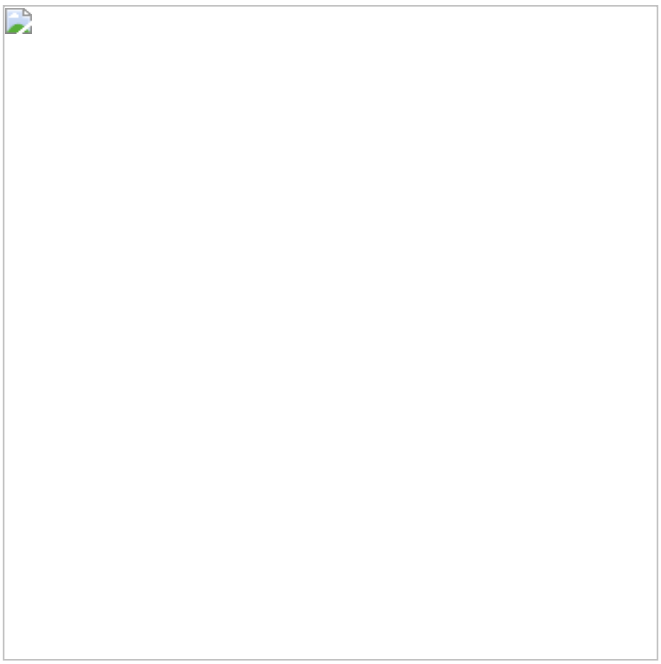
However, these issues do not outweigh the benefits of enabling the AD Recycle Bin.

Active Directory Object Recovery without the AD Recycle Bin

To illustrate the value of enabling the AD Recycle Bin, let's review what's involved in recovering an AD object when the AD Recycle Bin is not enabled.

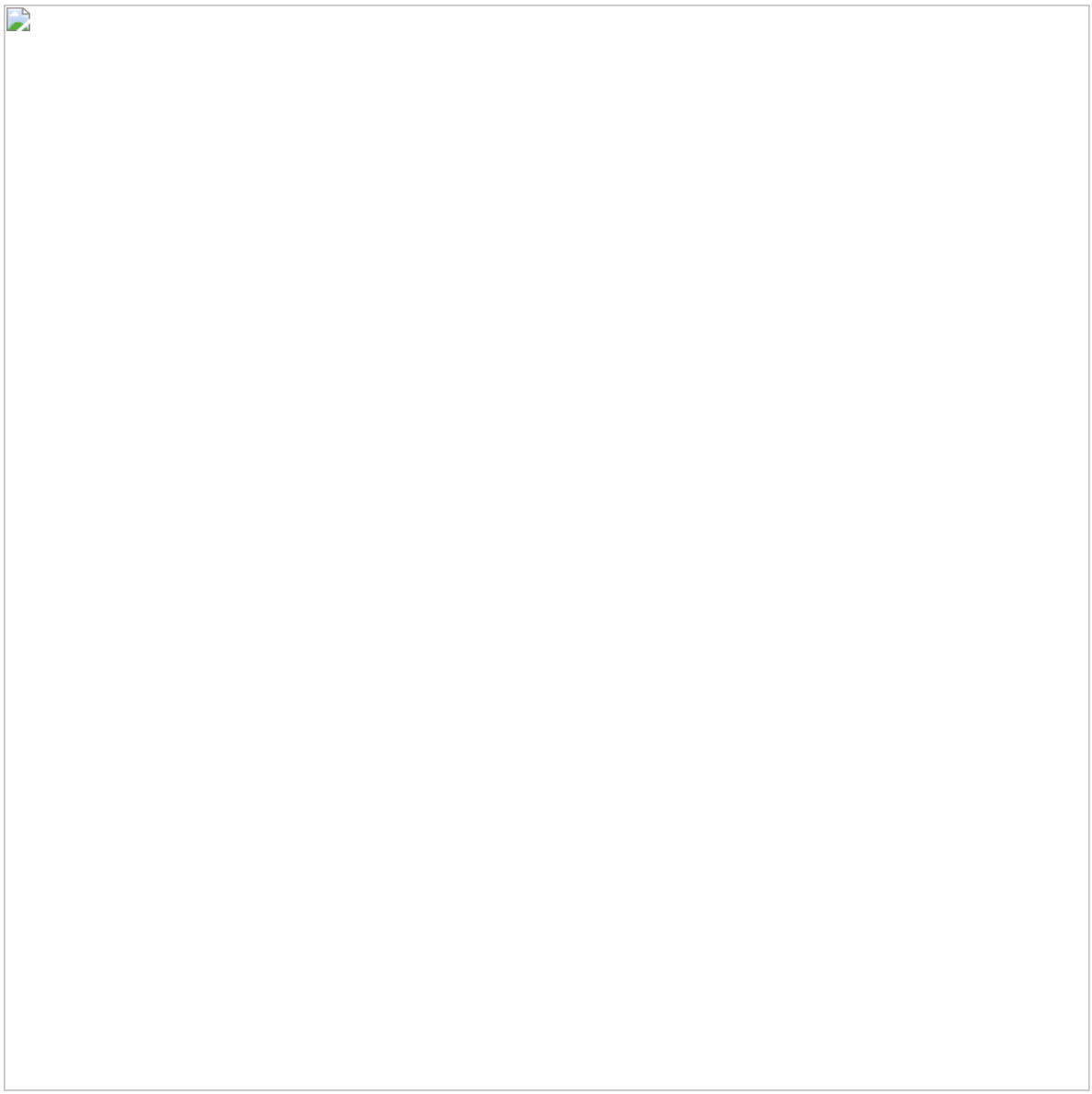
In a domain without the AD Recycle Bin enabled, when an Active Directory object is deleted, it becomes a **tombstone**. This object, stripped of the majority of its attributes, is kept in the partition's Deleted Objects container for the time period specified in the domain's **tombstoneLifetime**. During this period, the object is technically recoverable, but its lost attributes can be generally considered to be irrecoverable. Once the **tombstoneLifetime** value is reached, the object is garbage-collected into non-existence. This lifecycle is illustrated below:

Let's see how we can reanimate this tombstone using the Modify feature of the LDP utility:



1. Right-click the tombstone and select the **Modify** option.
2. In the **Edit Entry** section, enter the value “isDeleted” in the **Attribute** field, select the **Delete** radio button under **Operation**, and click the **Enter** button to add the entry to the Entry List.
3. In the **Edit Entry** section, enter the value “distinguishedName” in the **Attribute** field, enter distinguished name of object prior to its deletion in the **Values** field, select the **Replace** radio button under **Operation**, and click the **Enter** button to add the entry to the Entry List.
   Remember when I mentioned that the lastKnownParent attribute might end up being useful? Well, if you don’t know what the object’s dn was prior its deletion, you can try this trick: Take the current dn and replace the NULL terminated character (“A”) and everything to its right with the current value of the lastKnownParent attribute.
4. Select the **Extended** option at the bottom left of the panel.
5. Click the **Run** button.

Then we can find the reanimated object again and see what it looks like:



As you can see, we technically recovered the deleted user object. However, it's missing most of the information it possessed prior to its deletion.

You can (in theory) get around this problem by taking regular VSS snapshots of Active Directory. Then, if you need to recover a deleted object, you can "just" find a backup that was taken prior to the object's deletion, mount the snapshot using NTDSUTIL, connect to the mounted snapshot using an LDAP utility, locate the object, export it to… never mind.

But wait, it gets worse.

The Deleted Objects container isn't a forever kind of thing. The aptly named **tombstoneLifetime** property on the CN=Directory Service,CN=Windows NT,CN-Services,CN=Configuration,**ForestDistinguishedName** object defines the number of days before a deleted object will be permanently removed from Active Directory.

The value of **tombstoneLifetime** is based on the version of Windows Server that was involved in the creation of the domain's forest. It is set to 180 days (Microsoft's current recommended setting) by default in forests created using a version of Windows Server more recent than 2003. Older implementations default to 60 days. The behavior of the **tombstoneLifetime** property is actually worth paying attention to and is kinda cool. If the value exists, the tombstone lifetime is the value specified. Unless the value is less than 2; then the tombstone lifetime defaults to 60 days (Windows 2000 Server through Windows Server 2008) or 2 days (Windows Server 2008 R2 or later). If no value is specified, the value is 60 days.

If you're curious about the value of **tombstoneLifetime** in your environment, this PowerShell script will return it for you (it requires AD DS and AD LDS tools):

```
(Get-ADObject -Identity "CN=Directory Service,CN=Windows NT,CN=Services,$((Get-
ADRootDSE).configurationNamingContext)" -Properties *).tombstoneLifetime
```

Once an object has spent a tombstoneLifetime in the Deleted Objects container, it is logically deleted by Active Directory's garbage collection. At that point, it's gone and it's never coming back.

How Netwrix Can Help

The AD Recycle Bin is a useful tool for recovering recently deleted objects. For a more comprehensive solution, consider Netwrix Recovery for Active Directory. It enables you to restore backed-up objects that have exceeded their forest's mdDS-DeletedObjectLifetime and therefore are no longer recoverable using the AD Recycle Bin

Kevin Joyce
Senior Technical Product Manager at Netwrix. Kevin is passionate about cyber-security and holds a Bachelor of Science degree in Digital Forensics from Bloomsburg University of Pennsylvania.