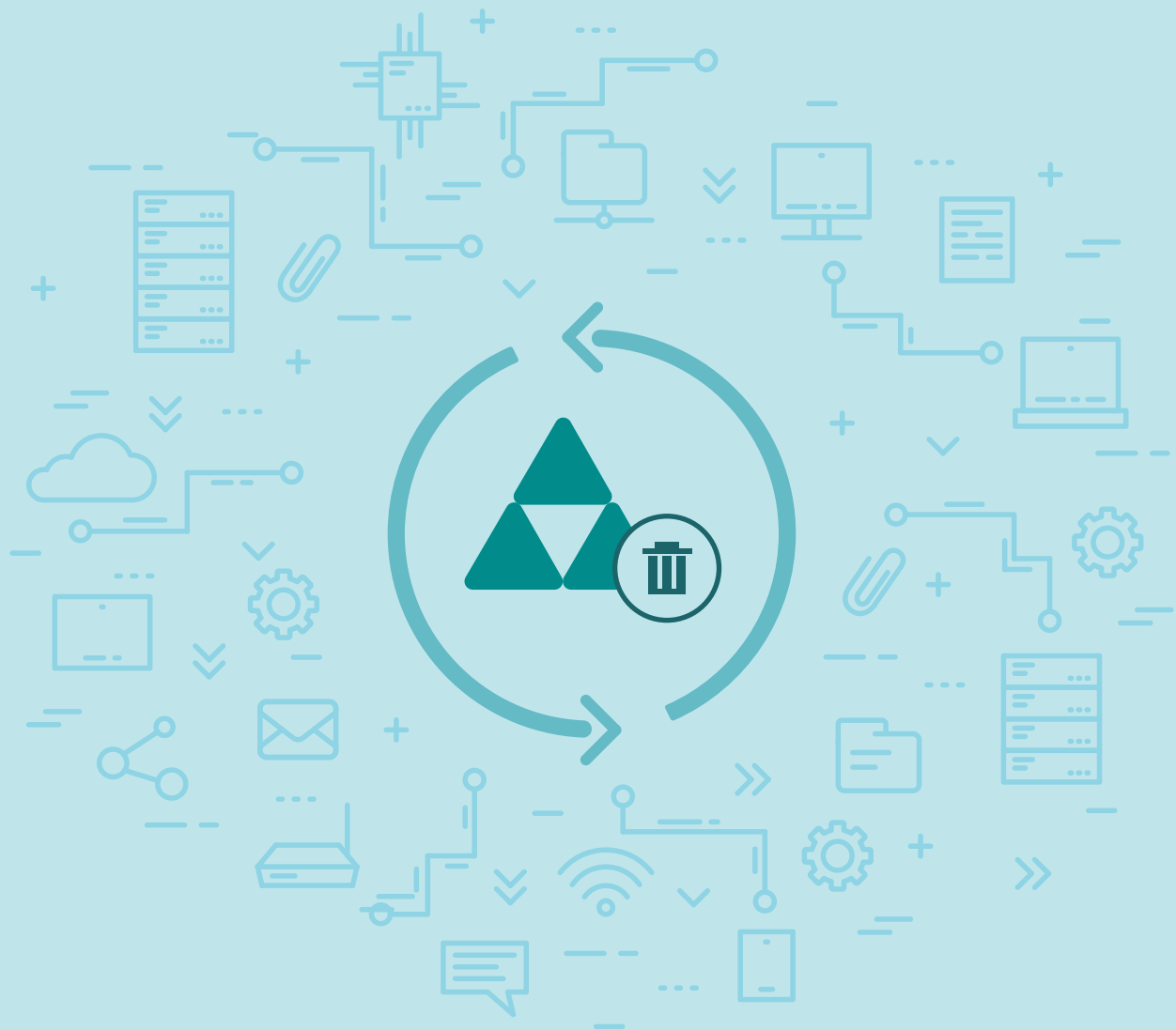


Recovering deleted Active Directory objects

RecoveryManager does
what PowerShell can't.



Introduction

Active Directory (AD) is the foundation of most critical organizational services. Case in point—any amount of extended downtime in AD will result in loss of productivity. Many administrators can relate to the frustration of accidental deletions causing issues for end users. This is why it's essential to have an effective disaster recovery plan in place to deal with accidental deletions in AD.

A comprehensive recovery plan for accidental deletion is critical for applications that continually run, like AD. Recognizing this need, Microsoft introduced an AD module for PowerShell in Windows Server 2008 R2. AD module-enabled PowerShell allows you to recover deleted objects and perform various other AD tasks.

This guide explores how to use PowerShell to restore deleted AD objects while highlighting some of its limitations, such as its lack of flexibility in recovering nested AD objects. Additionally, we'll discuss how RecoveryManager Plus, our comprehensive [AD backup solution](#), can not only perform these recovery tasks but also go beyond PowerShell's capabilities.

What can PowerShell do?

PowerShell is Microsoft's flagship command-line shell designed for system administrators. PowerShell has an interactive prompt and a scripting environment, both of which can be used independently or in combination to perform routine management tasks.

Initially, PowerShell was more of a server management tool as its AD integration was not fully complete. However, since Microsoft released its AD module for PowerShell, the ability to control nearly all aspects of AD and its objects has changed the playing field dramatically.

The Active Directory module from PowerShell allows administrators to perform the following functions:

- Restore a deleted AD object (e.g. user, group, OU, and GPO).
- Manually trigger a GPO backup.
- Recover a user's last set password upon restoration.
- Restore security permissions and authorizations provided to security groups.
- Perform deleted object restoration without having to restart the domain controllers.

A deeper look at PowerShell's capabilities

Never tried restoring a deleted AD object with PowerShell? Here's how it works.

To restore a deleted object, open PowerShell and type in the following command:

```
Restore-ADObject -Identity $dn
```

Here, \$dn is the distinguished name of the object to be restored. To find the distinguished name of the object, use the following script in PowerShell:

```
(Get-ADObject -SearchBase (get-addomain).deletedobjectscontainer -IncludeDeletedObjects -filter "samaccountname -eq '%OLD_NAME%' ")
```

To find the distinguished name of the object and to perform the restoration, use the following script in PowerShell:

```
(Get-ADObject -SearchBase (get-addomain).deletedobjectscontainer -IncludeDeletedObjects -filter "samaccountname -eq '%OLD_NAME%' ") | Restore-ADObject
```

Here, %OLD_NAME% is the name of the object before being deleted.

```

PS C:\Users\Administrator> (Get-ADObject -SearchBase (get-addomain).deletedobjectscontainer -IncludeDeletedObjects -filter "samaccountname -eq 'Brian'")

Deleted           : True
DistinguishedName : CN=Brian\0ADEL:15d5a7bd-7b97-4974-b93c-d5da666643bf,CN=Deleted Objects,DC=validate4,DC=com
Name              : Brian
ObjectClass       : user
ObjectGUID        : 15d5a7bd-7b97-4974-b93c-d5da666643bf

PS C:\Users\Administrator> (Get-ADObject -SearchBase (get-addomain).deletedobjectscontainer -IncludeDeletedObjects -filter "samaccountname -eq 'Brian'") | Restore-ADObject
PS C:\Users\Administrator>

```

To manually trigger a backup of a GPO, open PowerShell and type in the following command:

Backup-GPO -Name '\$GPOName' -Path \$path

\$GPOName is the name of the GPO to be backed up and ***\$path*** is the path where you want to store the backup.

```

Administrator: Windows PowerShell

PS C:\Users\Administrator> Backup-Gpo -Name TestGPO -Path C:\GpoBackups -Comment "Weekly Backup"

DisplayName      : TestGPO
GpoId            : bdbef146-3f8c-43cb-ae8-dfecd756546a
Id              : 2d0f9570-fee4-f1c-bc82-8bd3687d0b9a
BackupDirectory : C:\GpoBackups
CreationTime     : 8/11/2018 12:21:08 PM
DomainName      : validate4.com
Comment         : Weekly Backup

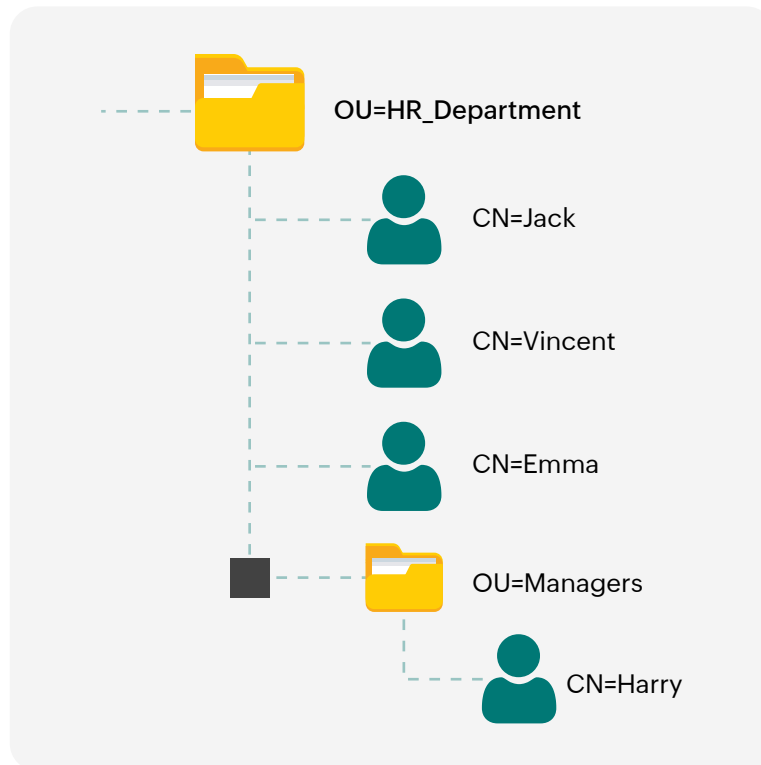
PS C:\Users\Administrator>

```

What makes PowerShell so hard to use?

There is no doubt about how powerful PowerShell is. However, as shown in Figures 1 and 2, PowerShell scripts can quickly become overly complex. Using PowerShell effectively requires a certain level of scripting expertise. Even for those skilled in scripting, restoring AD objects through PowerShell can be challenging—especially when dealing with multiple objects, where the complexity grows significantly.

Consider the following scenario: An administrator at example.com accidentally deletes a nested organizational unit (OU) called HR_Department, which contains user accounts of employees in the HR department. The deletion of the OU resulted in deletion of a nested OU called Managers, which contains user accounts of the managers who work in the HR department. Jack, Vincent, and Emma are user accounts in the HR_Department OU. Harry is a user account in the Managers OU. The following illustration shows the hierarchy of the HR_Department OU.



When the HR_Department OU is deleted, a total of six objects including the two OUs and the four users are moved to the Deleted Objects container with their distinguished names mangled. The Deleted Objects container displays all deleted objects in a flat hierarchy as its direct children, and the original hierarchy is lost. To restore the HR_Department OU, the administrator must identify that OU's original hierarchy. It's critical to restore objects from the highest level of the hierarchy because deleted objects can only be restored to a live parent.

If the administrator knows the original hierarchy of the deleted OU, they can use the **Restore-ADObject** cmdlet to retrieve the deleted objects one hierarchy level at a time. If the administrator is not familiar with the original hierarchy, the admin must first identify the hierarchy before starting the restoration process.

For example, if the administrator decides to find the hierarchy of the user account Emma, the PowerShell cmdlet must be constructed so that the lastKnownParent attribute of Emma is returned.

Get-ADObject -SearchBase "CN=Deleted Objects,DC=validate4,DC=com"
-ldapFilter:"(msDs-lastKnownRDN=Emma)" -IncludeDeletedObjects -Properties
lastKnownParent

```

Administrator: Windows PowerShell
PS C:\Users\Administrator> Get-ADObject -SearchBase "CN=Deleted Objects,DC=validate4,DC=com" -ldapFilter:"(msDs-lastKnownRDN=Emma)" -IncludeDeletedObjects -Properties lastKnownParent

Deleted : True
DistinguishedName : CN=Emma\OADEL:0e6c9355-a782-4731-a38c-13d76dbd14e5,CN=Deleted Objects,DC=validate4,DC=com
LastKnownParent  : OU=HR_Department\OADEL:d6625511-8637-4bde-b24e-f665bfa96e7b,CN=Deleted Objects,DC=validate4,DC=com
Name             : Emma
ObjectClass      : user
ObjectGUID       : 0e6c9355-a782-4731-a38c-13d76dbd14e5

PS C:\Users\Administrator>

```

In the output returned by this cmdlet, the administrator can notice that the value of *lastKnownParent* for Emma is HR_Department. The administrator can also notice that the distinguished name of the HR_Department OU is mangled, indicating the HR_Department OU object has been deleted.

Here's an example of a mangled distinguished name:

OU=HR_Department\OADEL:d6625511-4bde-b24e-f665bfa96e7b,CN=Deleted
 Objects,DC=validate4,DC=com

The administrator then has to search for all objects in the Deleted Objects container whose *lastKnownParent* value is HR_Department.

Get-ADObject -SearchBase "CN=Deleted Objects,DC=validate4,DC=com" -Filter
{lastKnownParent -eq 'OU=HR_Department\OADEL:d6625511-4bde-b24e
f665bfa96e7b,CN=Deleted Objects,DC=example,DC=com'} -IncludeDeleted
Objects -Properties lastKnownParent | ft

```

Administrator: Windows PowerShell
PS C:\Users\Administrator> Get-ADObject -SearchBase "CN=Deleted Objects,DC=validate4,DC=com" -Filter {lastKnownParent -eq 'OU=HR_Department\OADEL:d6625511-4bde-b24e-f665bfa96e7b,CN=Deleted Objects,DC=example,DC=com'} -IncludeDeletedObjects -Properties lastKnownParent | ft

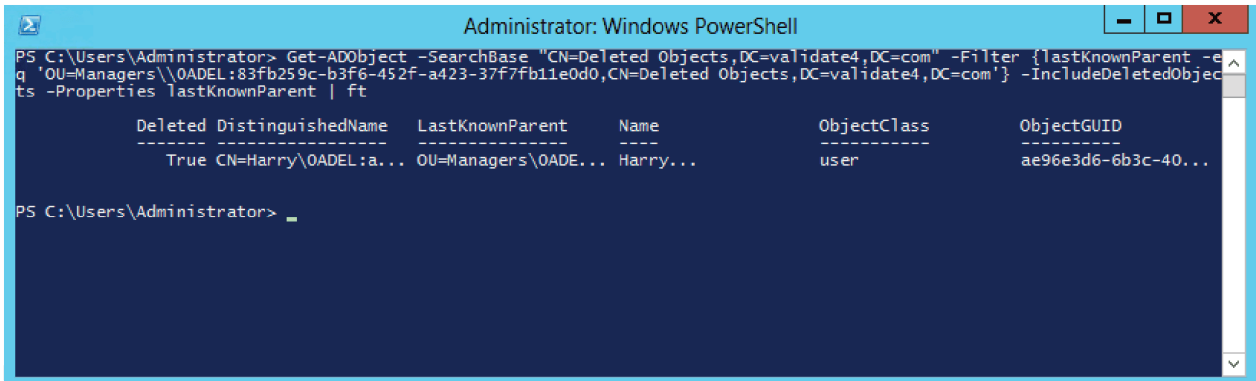
Deleted DistinguishedName LastKnownParent Name ObjectClass ObjectGUID
-----
True CN=Jack\OADEL:69... OU=HR_Department... Jack... user 699d5418-11b4-4b...
True CN=Vincent\OADEL... OU=HR_Department... Vincent... user 9278dbd0-912f-4d...
True CN=Emma\OADEL:0e... OU=HR_Department... Emma... user 0e6c9355-a782-47...
True OU=Managers\OADE... OU=HR_Department... Managers... organizationalUnit 83fb259c-b3f6-45...

PS C:\Users\Administrator>

```

In the output returned by this cmdlet, the administrator can notice that Managers is an OU. The administrator now has to search for all the deleted objects that were in the Managers OU. The objects in the Managers OU will contain a lastKnownParent attribute as Managers.

```
Get-ADObject -SearchBase "CN=Deleted Objects,DC=validate4,DC=com" -Filter  
{lastKnownParent -eq 'OU=Managers\0ADEL:83fb259c-b3f6-452f-a423-37f7fb11e0d0,  
CN=Deleted Objects,DC= validate4,DC=com'} -IncludeDeletedObjects -Properties  
lastKnownParent | ft
```



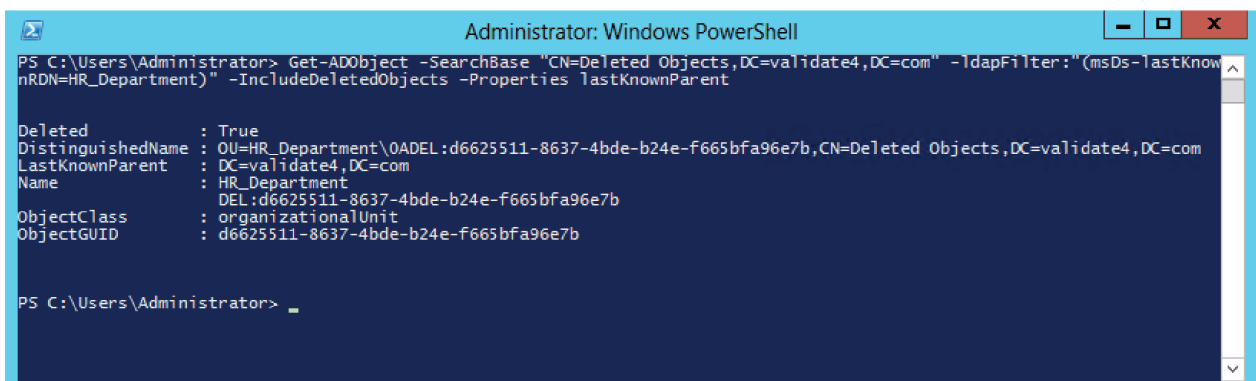
```
Administrator: Windows PowerShell
PS C:\Users\Administrator> Get-ADObject -SearchBase "CN=Deleted Objects,DC=validate4,DC=com" -Filter {lastKnownParent -eq 'OU=Managers\0ADEL:83fb259c-b3f6-452f-a423-37f7fb11e0d0,CN=Deleted Objects,DC=validate4,DC=com'} -IncludeDeletedObjects -Properties lastKnownParent | ft

Deleted DistinguishedName LastKnownParent Name ObjectClass ObjectGUID
-----
True CN=Harry\0ADEL:a... OU=Managers\0ADE... Harry... user ae96e3d6-6b3c-40...
```

In the output that this cmdlet displays, the administrator identifies only the user account Harry and no other objects within the OU. The administrator now has the list of all objects that were deleted and can start with the restoration.

Since the HR_Department OU is the object at the top of the hierarchy, it must be restored first. Because all previous investigation steps were performed using the lastKnownParent attribute—which points to the direct parent of the object and does not indicate whether the next parent object is also deleted—administrators can verify that the value of lastKnownParent for HR_Department is indeed a live OU by running the following command:

```
Get-ADObject -SearchBase "CN=Deleted Objects,DC= validate4,DC=com" -ldapFilter:"(msDs-lastKnownRDN=HR_Department)" -IncludeDeletedObjects -Properties lastKnownParent
```



```
Administrator: Windows PowerShell
PS C:\Users\Administrator> Get-ADObject -SearchBase "CN=Deleted Objects,DC=validate4,DC=com" -ldapFilter:"(msDs-lastKnownRDN=HR_Department)" -IncludeDeletedObjects -Properties lastKnownParent

Deleted : True
DistinguishedName : OU=HR_Department\0ADEL:d6625511-8637-4bde-b24e-f665bfa96e7b,CN=Deleted Objects,DC=validate4,DC=com
LastKnownParent : DC=validate4,DC=com
Name : HR_Department
ObjectClass : organizationalUnit
ObjectGUID : d6625511-8637-4bde-b24e-f665bfa96e7b

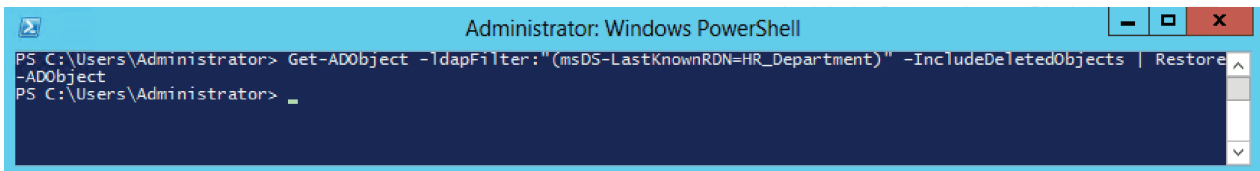
PS C:\Users\Administrator>
```

This concludes the investigation, and the administrator is ready to restore the HR_Department OU to its original hierarchy and state.

To restore the HR_Department OU using PowerShell, the administrator has to perform the following operations in the domain controller.

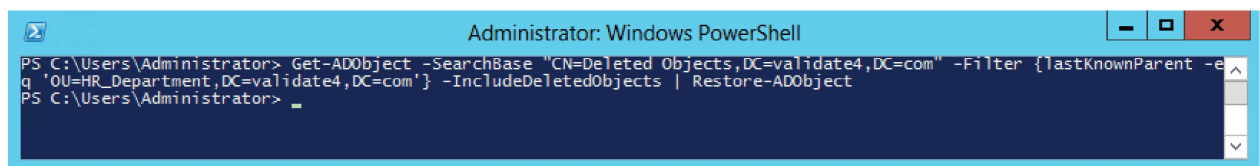
1. Click **Start**, and then click **Administrative Tools**. Right-click the **Active Directory Module for Windows PowerShell**, and then click **Run as administrator**.
2. Restore the HR_Department OU (the highest level of hierarchy of the objects to be restored) by running the following command:

```
Get-ADObject -SearchBase "CN=Deleted Objects,DC= validate4,DC=com"  
-ldapFilter:"(msDs-lastKnownRDN=HR_Department)" -IncludeDeletedObjects -Properties  
lastKnownParent
```



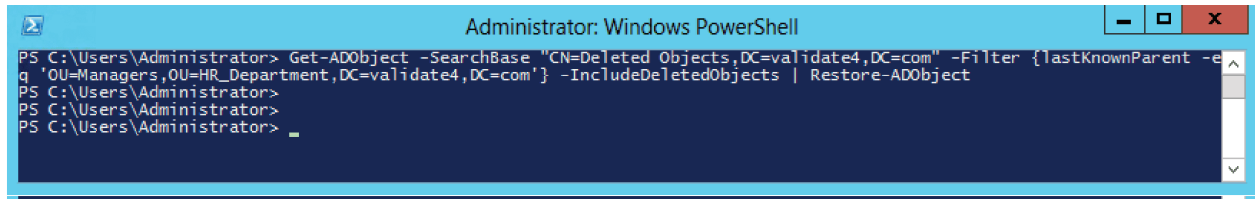
3. Restore the user accounts Jack, Vincent, and Emma and the Admins OU (the direct children of the HR_Department OU whose distinguished name was restored to OU=HR_Department,DC=validate4,DC=com in the previous step) by running the following command:

```
Get-ADObject -SearchBase "CN=Deleted Objects,DC= validate4,DC=com" -Filter  
{lastKnownParent -eq 'OU=HR_Department,DC= validate4,DC=com'} -IncludeDeletedObjects |  
Restore-ADObject
```



4. Restore the user account Harry (the direct child of the Admins OU whose distinguished name was restored to OU=Admins,OU=HR_Department,DC=validate4,DC=com in the previous step) by running the following command:

```
Get-ADObject -SearchBase "CN=Deleted Objects,DC= validate4,DC=com" -Filter  
{lastKnownParent -eq 'OU=HR_Department,DC= validate4,DC=com'} -IncludeDeletedObjects |  
Restore-ADObject
```

```
Administrator: Windows PowerShell
PS C:\Users\Administrator> Get-ADObject -SearchBase "CN=Deleted Objects,DC=validate4,DC=com" -Filter {lastKnownParent -eq 'OU=Managers,OU=HR_Department,DC=validate4,DC=com'} -IncludeDeletedObjects | Restore-ADObject
PS C:\Users\Administrator>
PS C:\Users\Administrator>
PS C:\Users\Administrator>
```

We can see from the above example that while PowerShell can be used to restore multiple deleted objects, the process is not always straightforward. Restoring multiple nested objects, in particular, can become increasingly complex and challenging.

Another important consideration when using PowerShell is the need to combine various tasks. For example, assume you need to find all deleted objects within a specific time period and restore only the user objects that were deleted during that period. You would need to run a script to generate a list of all objects deleted in the specified time period. Then find if the parent container of the deleted user objects is still live in AD. If not, you have to restore them as shown in the steps above before you can start restoring the user objects.

All of these tasks can be performed with PowerShell, but it requires too much time, effort, and knowledge to be considered a viable option. On the other hand, you have RecoveryManager Plus, which makes finding and restoring deleted user accounts quite simple.

What can RecoveryManager Plus do?

RecoveryManager Plus is an [AD backup and recovery tool](#) that helps administrators back up and restore all AD objects with ease. Its incremental backup functionality enables administrators to back up only the data that has changed since the previous backup cycle, significantly reducing the time needed for backup operations. This contrasts with the native backup tool, which supports only full backups of domain controllers and can be time-consuming.

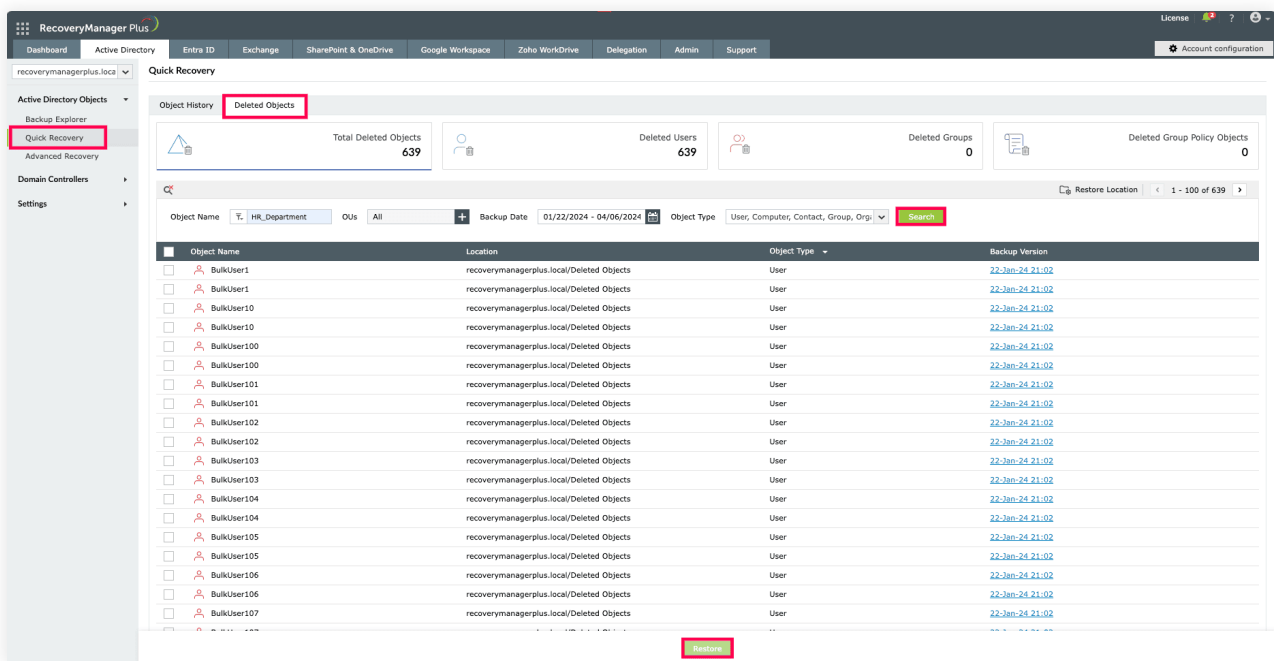
RecoveryManager Plus also allows you to restore deleted objects with all attributes intact, either to their last known location or to any new location as per your requirements.

Restoring deleted objects using RecoveryManager Plus

Considering the same use case as above, the following section explains how RecoveryManager Plus simplifies the process of restoring deleted objects.

Unlike PowerShell, that requires administrators to restore objects from the highest level of hierarchy, RecoveryManager Plus allows administrators to restore any object irrespective of its original hierarchy. When an object and its parent container are deleted, restoring the deleted object automatically restores the parent container too, eliminating the need to restore containers individually.

When the entire HR_Department OU is deleted, administrators can find the deleted OU in the *Deleted Objects* section in RecoveryManager Plus. Unlike the native AD Recycle Bin, which must be manually enabled, RecoveryManager Plus' recycle bin feature is active immediately after installation. All deleted AD objects are accessible here, and administrators can use built-in filters to refine search results by object type and deletion date.



If the administrator knows the original hierarchy of the deleted OU, the admin can select just the parent container and recycle it which will automatically restore all the constituent objects of the OU.

Alternatively, if the administrator doesn't know the parent container, administrators can select all the user objects that were deleted by checking the box next to them and can restore them with all their attributes intact. The users and the OUs, i.e. HR_Department and Managers, are also restored in the correct hierarchy.

Other key features of RecoveryManager Plus

Besides restoring deleted objects, RecoveryManager Plus is a multifaceted tool that has several capabilities that make it a must-have for administrators who want total control over the contents of their AD.

Features	PowerShell	RecoveryManager Plus
Restore live AD objects to any of their past versions	⊗	✓
AD rollback	⊗	✓
Granular GPO restoration	⊗	✓

[Learn more](#) about the various features that RecoveryManager Plus has to offer.

Summary

PowerShell is powerful. However, there's also no denying the fact that PowerShell is complex and difficult to use. PowerShell requires a deeper understanding of scripting to perform tasks that require mere clicks in RecoveryManager Plus. RecoveryManager Plus is designed with ease of use in mind. From recovering deleted users and restoring objects to any of their previous states to rolling back your entire AD to any past state, RecoveryManager Plus is the efficient AD backup solution your organization needs.

ManageEngine
RecoveryManager Plus

Our Products

AD360 | Log360 | ADManager Plus | ADAudit Plus
ADSelfService Plus | M365 Manager Plus

About RecoveryManager Plus

ManageEngine RecoveryManager Plus is a comprehensive backup and recovery solution for Active Directory, Entra ID, Microsoft 365, Google Workspace, on-premises Exchange and Zoho WorkDrive environments. With its incremental backups, flexible retention policies, backup immutability and multiple modes of restoration—such as domain controller recovery and object-, item- and attribute-level restoration—RecoveryManager Plus delivers a holistic solution for ensuring seamless business continuity by backing up all enterprise application data.

For more information, visit www.manageengine.com/ad-recovery-manager.

\$ Get Quote

↓ Download