Active Directory Disaster Recovery Template

Contents

[Active Directory DR Plan 4](#_Toc212009071)

[Introduction 4](#_Toc212009072)

[Purpose 4](#_Toc212009073)

[Summary 4](#_Toc212009074)

[Quick Reference Guide 5](#_Toc212009075)

[Emergency Contacts 5](#_Toc212009076)

[Critical Passwords 5](#_Toc212009077)

[Active Directory and Domain Controller Configuration Sheet 6](#_Toc212009078)

[General Information 6](#_Toc212009079)

[Network Configuration 6](#_Toc212009080)

[AD Role & Function 6](#_Toc212009081)

[Backup & Recovery 6](#_Toc212009082)

[Replication & Health 6](#_Toc212009083)

[Security & Credentials 7](#_Toc212009084)

[Hardware / Virtualization 7](#_Toc212009085)

[Logging & Monitoring 7](#_Toc212009086)

[Best Practices for Active Directory Backups 8](#_Toc212009087)

[Use Dedicated Backup Storage 8](#_Toc212009088)

[Use Dedicated Backup Storage 8](#_Toc212009089)

[Always Include the System State 8](#_Toc212009090)

[Back Up All FSMO Role Holders 8](#_Toc212009091)

[Retain Multiple Recovery Points 8](#_Toc212009092)

[Validate Backups Regularly 8](#_Toc212009093)

[Protect the DSRM Password 8](#_Toc212009094)

[Monitor and Log Backup Status 9](#_Toc212009095)

[Active Directory Pre-Recovery Steps 9](#_Toc212009096)

[Stabilize the Environment 9](#_Toc212009097)

[Capture and Document the Current State 9](#_Toc212009098)

[Check SYSVOL and DFSR Health 10](#_Toc212009099)

[Assess the Scope of the Problem 10](#_Toc212009100)

[Determine Recovery Scope 10](#_Toc212009101)

[Known Issue – Deleted Objects 11](#_Toc212009102)

[Unknown or Corruption Scenarios 11](#_Toc212009103)

[Communication and Change Control 11](#_Toc212009104)

[Proceed to Recovery (Once Verified) 11](#_Toc212009105)

[Active Directory Recycle Bin 13](#_Toc212009106)

[Overview 13](#_Toc212009107)

[Prerequisites 13](#_Toc212009108)

[Recoverable Objects 13](#_Toc212009109)

[Limitations and Considerations 13](#_Toc212009110)

[Active Directory Recovery Methods 16](#_Toc212009111)

[Non-Authoritative Restore (Standard Recovery) 16](#_Toc212009112)

[Prerequisites: 16](#_Toc212009113)

[Procedure: 16](#_Toc212009114)

[Authoritative Restore (Selective Object Recovery) 17](#_Toc212009115)

[Examples of Authoritative Restores 19](#_Toc212009116)

[FSMO Architecture Overview 22](#_Toc212009117)

[FSMO Role – Function, Criticality, and Seizure Impact 22](#_Toc212009118)

[FSMO Role Location: 22](#_Toc212009119)

[Schema Master 22](#_Toc212009120)

[Domain Naming Master 22](#_Toc212009121)

[Relative ID (RID) Master 23](#_Toc212009122)

[Primary Domain Controller (PDC) Emulator 23](#_Toc212009123)

[Infrastructure Master 24](#_Toc212009124)

[Global Catalog (GC) Interaction 24](#_Toc212009125)

[FSMO Placement Commands 25](#_Toc212009126)

[How to Seize the Operations Master Roles 25](#_Toc212009127)

[DNS Zone Recovery 27](#_Toc212009128)

[Backup Procedure 27](#_Toc212009129)

[Prepare backup directory 27](#_Toc212009130)

[DNS Server Restore 27](#_Toc212009131)

[Restore DNS zones 27](#_Toc212009132)

[GPO Recovery 28](#_Toc212009133)

[GPO Backup Overview 28](#_Toc212009134)

[GPO Backup Procedure 28](#_Toc212009135)

[Script Summary 28](#_Toc212009136)

[GPO Restoration Procedure 29](#_Toc212009137)

[SYSVOL DFSR Recovery 29](#_Toc212009138)

[D2 and D4 References by Microsoft 29](#_Toc212009139)

[Non-Authoritative Restore 29](#_Toc212009140)

[Authoritative Restore 30](#_Toc212009141)

[Non-Authoritative SYSVOL Restore (D2) 30](#_Toc212009142)

[Authoritative SYSVOL Restore (D4) 30](#_Toc212009143)

[Why Schema Rollbacks Are Not Possible 32](#_Toc212009144)

[USN (Update Sequence Number) by 100,000 32](#_Toc212009145)

[Tombstone period of 180 days 33](#_Toc212009146)

[Backup and Recovery Scripts 34](#_Toc212009147)

[AD Object Reference Script 34](#_Toc212009148)

[Restore GPO Link to OU 43](#_Toc212009149)

# Active Directory DR Plan

## Introduction

Active Directory is the backbone of a Microsoft network environment and underpins a wide range of critical services, including authentication (logon/logoff), email systems, access control for applications and data, and directory-based configurations. Given this central role, any disruption to Active Directory can have a cascading effect across multiple dependent systems. It is therefore essential to maintain a well-defined disaster recovery (DR) plan to address potential failures.

Active Directory infrastructure supports numerous business-critical applications and services. In the event that AD becomes unavailable, essential functions such as user authentication, policy enforcement, and access to core resources may be severely impacted.

## Purpose

The objective of this document is to define the specific disaster recovery scenarios applicable to Active Directory, assess the risks associated with each, and outline the recovery procedures necessary to restore service continuity. It provides a structured set of response actions to be executed immediately upon identification of a disaster event, ensuring that business operations can resume with minimal disruption.

## Summary

Active Directory is often the unseen foundation of an organization’s IT operations. The simple act of a user logging on, opening email, or launching an internal application relies entirely on Active Directory for authentication and authorization — yet these background processes are rarely considered until they fail. Every user login, computer account validation, and access request depends on Active Directory being healthy, synchronized, and available.

Active Directory supports critical services and functions, including:

* Users – Account data, privileges, profiles, and applied policies.
* Clients – Management settings, network configuration, and policies.
* Servers – Management profiles, network configuration, services, printers, shared files, and policies.
* DNS – Domain name resolution and service location records.
* Application Access – Authentication and authorization for SQL Server, web portals, and internal applications.
* Directory Lookups – Integration for Exchange and other directory-aware systems.
* Security – Centralized control using user and group objects for access and policy enforcement.

Given the reliance on these services, the resilience of the Active Directory infrastructure is critical. All authentication and authorization activity depends on it. In the event of hardware failure, data loss, or logical corruption within the AD database, tested and verified recovery procedures are essential to restore service continuity.

While deploying multiple domain controllers in separate locations provides redundancy against hardware or site failure, it does not protect against replicated logical corruption or deliberate deletions by a privileged administrator. For disaster recovery to be effective, clearly defined, documented, and well-understood recovery process are to be documented.

Administrators should be thoroughly familiar with these procedures and perform regular simulated recovery exercises in an isolated environment to ensure readiness, verify backup integrity, and maintain confidence in the organization’s ability to recover Active Directory when it matters most.

## Quick Reference Guide

### Emergency Contacts

AD Team Lead: [Name] - [Phone]

On-Call Admin: [Phone]

Change Management: [Phone]

Executive Approval: [Name] - [Phone]

### Critical Passwords

DSRM Password Location: [Secure location]

Service Account: DOMAIN\BackupServiceAccount ([Password vault])

## Active Directory and Domain Controller Configuration Sheet

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Check / Field | DC01 | DC02 | DC03 | Description / Notes |  |
| General Information |  |  |  |  |  |
| Hostname |  |  |  | FQDN of the domain controller |  |
| Site |  |  |  | AD site assignment |  |
| Domain |  |  |  | AD domain name |  |
| Forest |  |  |  | AD forest name |  |
| OS Version |  |  |  | Windows Server version/build |  |
| Installation Type |  |  |  | Full / Core / RODC |  |
| Network Configuration |  |  |  |  |  |
| IPv4 Address |  |  |  | Static IP configuration |  |
| Subnet |  |  |  | Associated subnet |  |
| Default Gateway |  |  |  | Network gateway |  |
| Preferred DNS |  |  |  | Primary resolver (another DC preferred) |  |
| Alternate DNS |  |  |  | Secondary resolver |  |
| Reverse DNS Record |  |  |  | Confirm PTR record |  |
| AD Role & Function |  |  |  |  |  |
| FSMO Roles |  |  |  | List roles held by DC |  |
| Global Catalog |  |  |  | GC enabled (Yes/No) |  |
| RODC |  |  |  | Read-only DC? |  |
| Time Source |  |  |  | PDCe or NTP reference |  |
| Recycle Bin Enabled |  |  |  | Forest feature status |  |
| Backup & Recovery |  |  |  |  |  |
| Windows Backup Scheduled |  |  |  | Frequency and tool |  |
| Last Backup Date |  |  |  | Date/time of last System State backup |  |
| Backup Verified |  |  |  | Date of last test restore |  |
| DNS, GPO and Object Reference backup script scheduled |  |  |  | Backup of objects as a reference and GPO and DNS data |  |
| Replication & Health |  |  |  |  |  |
| Replication Partners |  |  |  | Inbound/outbound DCs |  |
| Replication Summary |  |  |  | Output of repadmin /replsummary |  |
| Security & Credentials |  |  |  |  |  |
| DSRM Password Stored |  |  |  | Secure storage location (not actual password) |  |
| BitLocker Key Storage |  |  |  | Confirm recovery keys in AD |  |
| Hardware / Virtualization |  |  |  |  |  |
| Platform |  |  |  | Physical / VM |  |
| Hypervisor Host |  |  |  | Host or cluster name |  |
| CPU / RAM / Disk |  |  |  | Resource allocation |  |
| Snapshot Policy |  |  |  | Snapshots should be disabled |  |
| Logging & Monitoring |  |  |  |  |  |
| Event Log Retention |  |  |  | Size and overwrite policy |  |
| Monitoring Agent |  |  |  | Installed agent (e.g. SCOM, Zabbix) |  |

## Best Practices for Active Directory Backups

### The Obvious – Backup you fool

* The obvious, but not to all, enable BACKUPS using Windows Backup or another backup product.
* Enable the AD Recycle Bin to reduce the necessity of an Authoritative Restore.

### Use Dedicated Backup Storage

* Store backups on a dedicated local or network volume, not on the same disk as the operating system.
* If using a share, restrict access to a dedicated service account with write-only permissions.
* Don’t allow share or NTFS permissions to Users or Authenticated Users
  + Especially Full Control, you know who you are…. Muppet and you’re the apparent expert

### Always Include the System State

* Active Directory is stored within the NTDS.dit database, SYSVOL, the registry, COM+ class registration database, and critical boot files, all of which are part of the System State.
* Never rely on volume-level or file-based backups alone; they won’t capture the AD database metadata or replication data needed for restore.
* Backup SYSVOL as a file level backup.
* Export and Backup GPO’s to the file system.
* Export DNS and backup.

### Back Up All FSMO Role Holders

* Ensure every DC is protected, but at minimum always back up:
  + Schema Master
  + Domain Naming Master
  + RID Master
  + PDC Emulator
  + Infrastructure Master
* Losing an un-backed FSMO holder complicates recovery, role seizure should be the last resort

### Retain Multiple Recovery Points

Keep at least:

* 7 days of rolling backups on local or nearline storage.
* Weekly or monthly copies offsite or offline (for ransomware isolation).

### Validate Backups Regularly

* A backup that can’t restore isn’t a backup.
* Perform test restores to a non-production.

### Protect the DSRM Password

* The Directory Services Restore Mode (DSRM) password is required for any restore.
* Reset if unknown:
  + Open CMD as Domain Admin
  + Type ‘ntdsutil’
  + Type ‘set dsrm password’, press Enter
  + Type ‘reset password on server %serverName%’, press Enter
  + Type the new password \* 2, press Enter
  + Type q,q to quit ntdsutil
* Store the password securely (e.g., sealed envelope or password vault).  
  Without it, you can’t boot into DSRM to perform an authoritative or non-authoritative restore.

### Monitor and Log Backup Status

* Windows Server Backup logs to Event Viewer → Applications and Services Logs → Microsoft → Windows → Backup.  
  Set up alerting for:
* Event ID 4 – Backup completed successfully
* Event ID 5 – Backup failed
* Automate daily email summaries or forward logs to your monitoring system (SCOM, Wazuh, etc.).

## Active Directory Pre-Recovery Steps

Before initiating any recovery procedure, whether restoring a deleted object, performing a non-authoritative restore, or planning a full authoritative recovery, the priority is to stabilize, assess, and document the current state of the directory. These steps ensure you know exactly what you’re recovering, the scope of impact, and the health of replication and dependencies before taking irreversible actions.

### Stabilize the Environment

* Isolate affected Domain Controllers (DCs) if corruption or replication loops are suspected.
* Temporarily disconnect them from the network or disable replication partners in Active Directory Sites and Services.
* Confirm FSMO role holders:
* netdom query fsmo
* Record these, they are critical for any restore or role transfer decisions.
* Ensure at least one healthy DC remains operational and reachable before performing any recovery. Never restore multiple DCs simultaneously.

### Capture and Document the Current State

On the affected DC (or from a remote admin workstation):

* Capture domain health and configuration:
* Get-ADForest | Format-List
* Get-ADDomain | Format-List
* Get-ADDomainController | Format-List
* Check replication status:
* repadmin /replsummary
* repadmin /showrepl
* Verify DNS health:
* nslookup -type=all \_msdcs.domain.com
* nslookup domain.com
* nslookup \_msdcs.domain.com 127.0.0.1
* Document backup availability:
* wbadmin get versions

### Check SYSVOL and DFSR Health

Verify DFS Replication via Event Viewer:

* Open Applications and Services Logs → DFS Replication.
* Review for common issues:
  + Event ID 2213 – Replication stopped after dirty shutdown
  + Event ID 5002 / 5004 – Replication partner unreachable
  + Event ID 4102 / 4104 – Backlogs or replication failures
* Generate DFSR health report:
  + Get-DfsrBacklog -GroupName "Domain System Volume" -SourceComputer DC1 -DestinationComputer DC2
* Get-DfsrState
  + If backlog is 0, replication is up to date.
* Compare SYSVOL contents across DCs:
  + dir C:\Windows\SYSVOL\domain\Policies | Select Name, LastWriteTime
  + Compare output between DCs to ensure each GPO folder and its GPT.ini are identical.
* Validate GPO consistency between AD and SYSVOL:
* $ADGPOs = (Get-ADObject -SearchBase "CN=Policies,CN=System,DC=your,DC=domain" -Filter \*).Name
* $SYSVOLGPOs = Get-ChildItem "C:\Windows\SYSVOL\domain\Policies" | Select-Object -ExpandProperty Name
* Compare-Object $ADGPOs $SYSVOLGPOs
* No output = all GPOs are in sync.

### Assess the Scope of the Problem

Determine what’s broken and how far it’s spread:

* Missing or deleted AD objects?
* Replication errors?
* SYSVOL inconsistencies?
* Schema update or FSMO role issues?
* DNS or authentication failures?
* Run:
  + repadmin /replsummary
  + repadmin /showrepl
  + dcdiag /v /c /e /f:dcdiag.log
  + Check Event Viewer for Directory Service, DNS Server, and DFSR logs for signs of replication failure, database corruption, or version mismatches.

### Determine Recovery Scope

Identify precisely which components require recovery:

| * Scope | * Recovery Type |
| --- | --- |
| * Specific user, group, or computer objects | * Recycle Bin or Authoritative Restore (OU subtree) |
| * Single Domain Controller | * Non-Authoritative Restore |
| * Entire Organizational Unit | * Authoritative Restore (subtree) |
| * GPOs or SYSVOL | * DFSR / SYSVOL Recovery |
| * DNS Zones or Records | * Zone restore or AD-integrated DNS recovery |
| * Entire Domain or Forest | * Full Domain or Forest Recovery |

### Known Issue – Deleted Objects

If the problem is confirmed object deletion:

If Recycle Bin is enabled:

* Get-ADObject -Filter 'isDeleted -eq $true' -IncludeDeletedObjects
* Restore-ADObject -Identity <GUID>

If Recycle Bin is not enabled or deletion predates it:

* Identify the affected OU or partition.
* Use authoritative restore from a System State backup.
* Confirm replication health before reintroducing the restored DC to the network.

### Unknown or Corruption Scenarios

If the root cause is unclear (e.g., replication errors, missing data, authentication failures):

Run integrity checks:

* ntdsutil
* activate instance ntds
* files
* integrity

Review NTDS and replication logs for JET errors, USN rollback, or lingering object warnings.

Identify if corruption is isolated to a single DC or replicated across the forest.

* If isolated → Non-authoritative restore that DC.
* If replicated → Authoritative restore required.

### Communication and Change Control

Before executing any recovery:

* Notify all relevant teams (helpdesk, identity, network, and security).
* Record findings, replication states, and backup versions.
* Disable automation or scripts that modify AD during recovery to prevent race conditions.

### Proceed to Recovery (Once Verified)

After confirming:

* Backups are valid and current
* The scope is well defined
* Healthy replication partners exist
* You can safely proceed with the chosen recovery method, Recycle Bin, Non-Authoritative Restore, or Authoritative Restore, based on your analysis.
* Backup Procedures
* Backup of DNS and GPO
* DNS and GPO need to be backed up separately from the Windows Backup
* Create a Service account with Domain Admin, assign User Rights to both ‘Logon As A Service’ and ‘Logon as Batch’.
* Copy the script below to a DC and permission so only DA’s can amend.
* Schedule a nightly task

System State backup includes:

* Active Directory database (NTDS.dit)
* Active Directory log files
* SYSVOL folder structure and Group Policy objects
* Boot files and system files
* COM+ Class Registration Database
* System Registry

Creating System State Backup (Windows Server 2016+)

* Using Windows Server Backup (GUI)
* Launch: wbadmin.msc
* Click "Local Backup" in left pane
* Click "Backup Schedule" in Actions pane
* Select "Custom" backup option
* Click "Add Items" and select "System State", SYSVOL, NTDS and C:\ADBackups directories
* Configure backup destination eg local dedicated drive
* Set schedule
* Verify backup completes successfully
* Backup Documentation
* Maintain a backup log including:
* Backup date and time
* Backup location and storage device
* Backup verification status
* Domain controller backed up
* Forest and domain functional levels
* List of operations masters at backup time
* Known issues in domain at backup time

## Backup Configuration

### System State backup schedule configuration

### Service account setup

### Scheduled task creation for backup script

### Backup monitoring and alerting setup

### Backup retention policy

## Active Directory Recycle Bin

### Overview

The Active Directory Recycle Bin is a feature that provides the ability to restore deleted Active Directory objects without needing to perform an authoritative restore from backup media. Introduced in Windows Server 2008 R2, this feature significantly simplifies the recovery process for accidentally deleted AD objects and preserves all attributes during restoration.

How to Enable the Active Directory Recycle Bin

### Prerequisites

Before enabling the AD Recycle Bin, ensure the following requirements are met:

* Forest Functional Level: The forest must be at Windows Server 2008 R2 or higher functional level
* Schema Version: The AD schema must be updated to support the feature
* Permissions: You must be a member of the Enterprise Admins group
* Important Warning: Once enabled, the AD Recycle Bin cannot be disabled
* Enabling via PowerShell

The recommended method for enabling the AD Recycle Bin is through PowerShell:

* Enable-ADOptionalFeature –Identity 'Recycle Bin Feature' –Scope ForestOrConfigurationSet –Target 'yourdomain.com'
* After running this command, you'll receive a warning about the irreversible nature of this operation. Type 'Y' to confirm.

Enabling via Active Directory Administrative Center

* Open Active Directory Administrative Center (ADAC)
* Navigate to your domain in the left pane
* Click on "Enable Recycle Bin" in the Tasks pane on the right
* Confirm the warning dialog
* Refresh ADAC to see the "Deleted Objects" container
* The feature typically takes approximately 15 minutes to replicate across all domain controllers in the forest.

### Recoverable Objects

The AD Recycle Bin can recover virtually all types of Active Directory objects, including:

* User accounts: All user attributes, group memberships, and properties are preserved
* Computer accounts: Including domain-joined computer objects
* Security and distribution groups: With all group memberships intact
* Organizational Units (OUs): Including all objects within the OU structure
* Contacts: Mail-enabled and standard contacts
* Group Policy Objects (GPOs): Though GPO links may need to be manually restored
* Other AD objects: Such as printers, shared folders, and custom schema objects
* When an object is deleted, it moves to the Deleted Objects container and enters a "deleted" state where it remains for the duration of the deleted object lifetime (180 days by default). During this period, most attributes are preserved, making full restoration possible.

### Limitations and Considerations

#### Time-Based Limitations

Deleted Object Lifetime:

By default, deleted objects are retained for 180 days. After this period, objects enter a "recycled" state where most attributes are removed, leaving only a minimal subset. Once in the recycled state, full restoration is no longer possible.

Recycled Object Lifetime:

After the recycled object lifetime expires (an additional 180 days by default), objects are permanently removed through garbage collection and cannot be recovered.

#### Functional Limitations

Cannot Be Disabled:

Once enabled, the AD Recycle Bin feature cannot be turned off. This is a permanent forest-level change.

No Rollback:

There is no way to revert to the previous tombstone-based deletion model after enabling the feature.

Storage Overhead:

Deleted objects consume space in the Active Directory database until they are permanently removed through garbage collection. Organizations with frequent deletions may see increased database size.

Replication Considerations:

Deleted objects must replicate across all domain controllers, which can impact replication traffic in environments with many deletions.

#### Recovery Limitations

Container Objects:

When recovering an OU or container, you must first restore the container itself before restoring child objects. The hierarchy must be rebuilt from the top down.

Linked Attributes:

While most attributes are preserved, some complex linked attributes or dependencies may require manual verification after restoration.

No GUI for Recycled Objects:

Objects in the recycled state (after the deleted object lifetime expires) cannot be viewed or restored through the Active Directory Administrative Center interface; PowerShell must be used.

GPO Links:

While GPOs themselves can be restored, GPO links to OUs are not automatically restored and must be manually re-established.

#### Find deleted objects in recycle bin with PowerShell

List Delete Objects

* Get-ADObject -Filter {isDeleted -eq $true} -IncludeDeletedObjects | Format-List Name, DeletedDate

Find specific deleted user

* Get-ADObject -Filter {(samAccountName -eq "username") -and (isDeleted -eq $true)} -IncludeDeletedObjects

Restore deleted user to original location

* Restore-ADObject -Identity (Get-ADObject -Filter {(samAccountName -eq "username") -and (isDeleted -eq $true)} -IncludeDeletedObjects).ObjectGUID

Restore deleted user to specific OU

* Restore-ADObject -Identity $UID -TargetPath "OU=Users,DC=domain,DC=com"

Restore deleted organizational unit and all contents

* Get-ADObject -Filter {(Name -eq "DeletedOU") -and (isDeleted -eq $true)} -IncludeDeletedObjects | Restore-ADObject

Restore deleted group

* Restore-ADObject -Identity (Get-ADObject -Filter {(Name -eq "GroupName") -and (isDeleted -eq $true)} -IncludeDeletedObjects).ObjectGUID

Verify restored object properties

* Get-ADUser -Identity "username" | Format-List \*

## Active Directory Recovery Methods

### Non-Authoritative Restore (Standard Recovery)

Use non-authoritative restore when the domain controller hardware failed but the domain is otherwise healthy, this assumes that the DC is non-bootable.

Deploying a fresh Domain Controller is almost always preferable to performing a non-authoritative restore. A clean build ensures you’re running from a known-good, current installation without inheriting potential corruption or stale metadata from backups. It allows you to install the latest patches, security baselines, and configuration hardening from the outset. Once joined to the domain, the new DC will automatically replicate the directory data from healthy peers, cleanly re-establishing consistency across the forest.

FSMO roles can then be gracefully transferred or seized as needed, without the risk of reintroducing outdated or conflicting data.

### Prerequisites:

* Access to system state backup created before the failure
* At least one other healthy domain controller in the domain
* External backup media available and accessible

### Procedure:

Install Server OS

* Install a new server of the same OS version and patch to the latest or at least to the same version as the now dead DC
* Install Windows Backup Feature
* Set network settings and IP
* Type msconfig and select Boot > Safe Boot > AD Repair or Boot into DRMS with Shift and F8
* Reboot and logon as the local admin
* Mount volume or copy backup to local server
* Launch Windows Backup
* Click Recover…
* Select This Server
* Select Date of the restore point
* Select Recovery Type = System State
* Restore to the original location
  + There is a check box to Authoritatively restore of AD – Don’t check
* Reboot the server after the restore completes and ensure newer objects are replicated from another DC.

Restart domain controller in normal mode

* Restart-Computer -Force
* Post-Recovery Steps:
* Log back in as domain administrator after restart
* Run these commands on the recovered domain controller

Replication with domain controllers

* repadmin /syncall /d /P

Verify replication completes

* repadmin /replsummary
* repadmin /showrepl

Check for replication errors

* Get-ADReplicationFailure -target DC01 | Format-Table

Run diagnostic tests

* dcdiag /f:c:\dcdiag.log

Verify DNS registration

* ipconfig /registerdns
* nslookup DCNAME

Monitor Event Viewer for errors

* Directory Service logs
* Replication events
* DNS server logs

### Authoritative Restore (Selective Object Recovery)

#### When to user an Authoritative Restore and when to use the Recycle Bin

The Active Directory Recycle Bin is excellent for recovering deleted objects, but it doesn’t help when existing objects or entire directory partitions are corrupted, overwritten, or lost due to replication issues or human error. That’s where an authoritative restore still matters.

The Recycle Bin can only restore objects that were intentionally deleted and still exist in the deleted objects container. It can’t roll back changes—like group membership modifications, GPO edits, or accidental bulk updates—because those aren’t deletions; they’re legitimate writes replicated across all domain controllers. If an admin accidentally removes every user from a security group or a script wipes out configuration attributes, those changes replicate instantly, and the Recycle Bin won’t help.

An authoritative restore allows you to recover data from a known-good backup and mark that version of the directory as the master copy during replication. This forces other domain controllers to accept the restored version as authoritative, overwriting the bad data across the forest.

In short:

* Recycle Bin = undelete objects.
* Authoritative Restore = roll back to a previous state after corruption or unwanted changes.
* Both have their place, but they solve entirely different problems.

#### Authoritative Restore Procedure

Use authoritative restore to recover deleted Active Directory objects that should not be overwritten by replication from other domain controllers. The restored objects are marked as authoritative and will replicate outbound to other DCs.

#### Prerequisites

You must have a System State backup of the domain controller taken when the data was good.

You must know which partition (naming context) you’re restoring:

* Domain partition: DC=yourdomain,DC=com
* Configuration partition: CN=Configuration,DC=yourdomain,DC=com
* Application partition: e.g. DC=ForestDnsZones,DC=yourdomain,DC=com
* Ensure other domain controllers are healthy, you’ll be restoring to one DC and then replicating out.

#### Restore AD Partition Procedure

Boot into Directory Services Restore Mode (DSRM)

* Type msconfig and select Boot > Safe Boot > AD Repair or Boot into DRMS with Shift and F8
* Reboot
* Log in using the DSRM administrator password (set during DC promotion).
* Mount volume or copy backup to local server
* Launch Windows Backup
* Click Recover…
* Select This Server
* Select Date of the restore point
* Select Recovery Type = System State
* Restore to the original location
  + There is a check box to Authoritatively restore of AD – Don’t check
* Don’t Reboot the server
* This step restores the AD database (NTDS.dit) from backup, but at this stage, it’s considered non-authoritative, other domain controllers would overwrite it during replication if you booted normally.

#### Mark the Partition as Authoritative

* Open Command Prompt (still in DSRM).
* Open ntdsutil in CMD
* Type ‘activate instance ntds’
* Type ‘authoritative restore’
* Then specify the partition you want to make authoritative, for example:
* restore database
* or, to target a specific partition:
* restore subtree "DC=yourdomain,DC=com"
* restore subtree "CN=Configuration,DC=yourdomain,DC=com"
* When prompted, confirm.
* This process updates the Update Sequence Numbers (USNs) on all objects in that partition so that when replication resumes, other DCs treat this version as newer and replicate it outward.
* Type quit twice to exit ntdsutil.

Reboot Normally

* Restart the domain controller normally. Once it starts, it will advertise as healthy and begin replicating the restored (authoritative) data to the rest of the forest.

Validate Replication

* Run the following to verify replication and event logs:
* repadmin /showrepl
* repadmin /syncall /AdeP
* Check the Directory Service event log for replication success (Event ID 1109 and similar).

### Examples of Authoritative Restores

#### Authoritative Restore of a Single OU

Scenario: An administrator accidentally deletes the Sales OU, including all user and computer accounts, and the deletion has replicated to all DCs.

Goal: Restore only the Sales OU and its objects from a System State backup.

Steps:

* Boot the affected DC into Directory Services Restore Mode (DSRM).
* Restore the System State using Windows Server Backup:
* Do not reboot when prompted after the restore completes.
* Open a Command Prompt and run:
* ntdsutil
* activate instance ntds
* authoritative restore
* restore subtree "OU=Sales,DC=corp,DC=example,DC=com"
* quit, quit
* Restart the domain controller normally.

#### Authoritative Restore of an Entire Domain Partition

Scenario: A script corrupts large parts of the domain (e.g. user attributes overwritten or objects missing) across all domain controllers.

Goal: Restore the entire domain partition to its last known good state.

Steps:

* Boot into DSRM.
* Perform a System State restore as above.
* Before rebooting, run:
* ntdsutil
* activate instance ntds
* authoritative restore
* restore database
* quit, quit
* Reboot the DC normally.

#### Authoritative Restore of a Configuration Partition

Scenario: An improperly written schema extension or deleted configuration container (e.g. Exchange or DFS settings) breaks forest-wide functionality.

Goal: Restore the Configuration naming context to a working version from backup.

Steps:

* Boot into DSRM.
* Perform a System State restore as above.
* Before rebooting, run:
* ntdsutil
* activate instance ntds
* authoritative restore
* restore subtree "CN=Configuration,DC=corp,DC=example,DC=com"
* quit, quit

#### Authoritative Restore of a Deleted Group Policy Object (GPO)

Scenario: A key GPO (e.g. Default Domain Policy) was deleted and the Recycle Bin isn’t enabled.

Goal: Restore the deleted GPO from backup.

Steps:

* Boot into DSRM.
* Perform a System State restore as above.
* Before rebooting, run:
* ntdsutil
* activate instance ntds
* authoritative restore
* restore subtree "CN={6AC1786C-016F-11D2-945F-00C04FB984F9},CN=Policies,CN=System,DC=corp,DC=example,DC=com"
* quit, quit
* Restart the DC.
* Compare SYSVOL contents and re-link GPOs if needed.

#### Partial Object Restore (Single User or Group)

Scenario: A critical service account or group was deleted and is not recoverable via the Recycle Bin.

Goal: Restore only that object without affecting the rest of the directory.

Steps:

* Boot into DSRM.
* Perform a System State restore as above.
* Before rebooting, run:
* ntdsutil
* activate instance ntds
* authoritative restore
* restore object "CN=svc\_SQL,OU=Service Accounts,DC=corp,DC=example,DC=com"
* quit, quit

#### Authoritative Restore After a Forest-Wide Rollback

Scenario: A misconfiguration or replication storm corrupts all domain partitions, and restoring a single DC is not sufficient.

Goal: Perform a forest-level authoritative restore (usually from the forest root DC).

Steps:

* Boot into DSRM.
* Perform a System State restore as above.
* Before rebooting, run:
* ntdsutil
* activate instance ntds (restore all naming contexts:)
* authoritative restore
* restore database
* quit, quit
* Reboot, allow replication to push authoritative copies to all other DCs

## FSMO Architecture Overview

### FSMO Role – Function, Criticality, and Seizure Impact

FSMO (Flexible Single Master Operations) roles exist because certain Active Directory operations cannot be performed simultaneously across multiple domain controllers without creating logical inconsistencies, data corruption, or irreconcilable conflicts. These operations require a single authoritative source, a master role holder.

There are five FSMO, roles two forest-wide and three domain-wide, that ensure consistency in schema, domain naming, object creation, and directory replication.

### FSMO Role Location:

Forest-Wide Roles (1 per forest):

├── Schema Master

├── Domain Naming Master

└── (Typically both on same DC)

Domain-Wide Roles (1 per domain):

├── PDC Emulator

├── RID Master

└── Infrastructure Master

### Schema Master

Scope: Forest-wide  
Default location: First DC in the forest root domain

Purpose:  
Controls all changes to the Active Directory schema, the master definition of all object classes and attributes. When products such as Exchange, SCCM, or Azure AD Connect extend the schema, the changes are applied only via the Schema Master.

If unavailable:

* No schema updates or extensions can occur.
* Existing directory operations continue normally.

Improper seizure consequences:  
If the Schema Master role is seized and the original DC is later brought online, schema version conflicts can occur.

* Both DCs may believe they are authoritative for schema updates.
* Replication may fail with “schema mismatch” errors.
* Corrupted or divergent schema versions can render the forest unrecoverable without a rebuild.

Criticality: Low (short term), High (during schema updates)

### Domain Naming Master

Scope: Forest-wide  
Default location: First DC in the forest root domain

Purpose:  
Controls the creation and deletion of domains and application directory partitions in the forest. It ensures that all domain names are unique and updates the forest-wide naming context.

If unavailable:

* Cannot create or remove domains or application partitions.
* Daily operations and replication unaffected.

Improper seizure consequences:  
If the role is seized incorrectly or the original DC returns:

* The forest may contain duplicate domain or application partition records.
* “Orphaned” domain entries may remain in AD, causing replication or trust inconsistencies.
* Forest integrity checks (ntdsutil, repadmin /showrepl) may show naming collisions or metadata errors.

Criticality: Low (short term), High (during forest restructuring)

### Relative ID (RID) Master

Scope: Domain-wide

Purpose:  
The RID Master allocates RID pools to all domain controllers within its domain. Each new security principal (user, group, or computer) is assigned a unique Security Identifier (SID), which consists of the domain SID plus a Relative ID (RID). The RID Master ensures that each domain controller issues unique RIDs, preventing SID duplication across the domain.

If unavailable:

* DCs can continue creating new objects until their current RID pool runs out.
* Each domain controller receives a RID pool of 500 identifiers. When approximately half of the pool (250 RIDs) has been consumed, the DC automatically requests a new allocation from the RID Master.
* Once depleted, no new users, groups, or computers can be created.

Improper seizure consequences:  
If the RID Master role is seized while the original DC is still active or later reintroduced:

* Multiple DCs may issue overlapping RID pools, resulting in duplicate SIDs.
* SID duplication breaks access control and causes permanent security inconsistencies.
* AD replication may fail due to RID conflicts (event ID 16650 or event ID 16658).
* A forest rebuild may be required to safely recover.

Criticality: Medium (short term), High (if RID pool exhausted or duplicated)

### Primary Domain Controller (PDC) Emulator

Scope: Domain-wide

Purpose:  
The PDC Emulator is the operational backbone of the domain. It performs several time-sensitive and synchronization functions:

* Password authority: Receives immediate replication of password changes; resolves password mismatches during authentication.
* Time synchronization: Acts as the authoritative time source for the domain; the forest root PDC is the master clock for the forest.
* Group Policy master: Central point for GPO creation and editing (GPMC connects directly to it).
* Trust management: Maintains trust passwords and NT4 compatibility for legacy systems.

If unavailable:

* Password changes stop replicating promptly; recently changed passwords may fail to authenticate.
* Kerberos time skew may cause logon and service ticket failures.
* GPO edits fail or are inconsistent.
* External trusts and legacy systems may fail authentication.

Improper seizure consequences:  
If seized incorrectly or the old PDC Emulator is later reintroduced:

* Two DCs may both act as time authorities, creating time drift and Kerberos failures.
* Password replication loops or mismatches may occur, leading to logon failures.
* Group Policy edits may conflict or become corrupted.
* Trust relationships may intermittently fail.

Criticality: Very High, immediate recovery or transfer required

### Infrastructure Master

Scope: Domain-wide

Purpose:  
Maintains cross-domain object references by updating phantom objects, lightweight records that represent objects (users, groups, or computers) from other domains.  
When a cross-domain object is renamed, moved, or deleted, the Infrastructure Master updates these phantom references to keep directory data consistent.

If unavailable:

* Cross-domain group memberships or references display outdated or incorrect names.
* Access control continues to function (based on SIDs), but visible names become stale.
* In a single-domain forest, or where all DCs are Global Catalogs, this role has no functional impact.

Improper seizure consequences:  
If the Infrastructure Master is seized incorrectly or the original DC is brought back online:

* Phantom object records may become inconsistent between DCs.
* Cross-domain group memberships can appear corrupted or out of sync.
* Event logs may show phantom cleanup or reference errors (event ID 1412).
* Requires metadata cleanup and GC resync to correct.

Criticality: Low to Medium

Special note on GCs:  
In a multi-domain forest, the Infrastructure Master should not reside on a Global Catalog unless all DCs are GCs. Otherwise, it will never detect outdated phantom references.

### Global Catalog (GC) Interaction

Although not a FSMO role, the Global Catalog (GC) interacts closely with FSMO operations. It stores a partial replica of every object across the forest, enabling:

* Universal group membership resolution during logon.
* Forest-wide searches and object lookups.
* Phantom object validation for the Infrastructure Master.

Best practice: host at least one GC per site for redundancy and authentication performance.

### FSMO Placement Commands

CMD

* netdom query fsmo

PowerShell

* Get-ADForest | Select-Object -Property SchemaRoleOwner, DomainNamingMasterRole
* Get-ADDomain | Select-Object -Property InfrastructureRoleMaster, RidRoleOwner, PDCEmulator

### How to Seize the Operations Master Roles

FSMO roles are normally transferred gracefully using standard Active Directory management tools such as Active Directory Users and Computers, Domains and Trusts, or the Schema Management snap-in. These tools require the current FSMO holder to be online and reachable. However, in the event of a permanent failure or loss of connectivity to the role holder, the role must be seized by another domain controller.

Role seizure is a last-resort operation and should only be performed if the existing FSMO host cannot be recovered within a reasonable time frame. Improper or premature seizure can cause directory inconsistencies and replication issues.

Potential Risks of Role Seizure

* Data Loss or Inconsistent Directory State  
  When a FSMO role is seized, the new holder begins operating based on its current AD replication state. If replication was incomplete before the original FSMO server failed, the new role holder may be missing recent updates.  
  To reduce this risk, wait for at least one full replication cycle to complete across all domain controllers before seizing the role, ensuring the target DC has the latest data.
* Duplicate Role Holders  
  Because the original FSMO server is offline during the seizure, it does not receive notification that its role has been reassigned. If that server is later brought back online—such as after repair or restoration from backup—it may still consider itself the valid FSMO holder.  
  Having two domain controllers performing the same FSMO role can lead to severe directory corruption or conflicting updates.

To prevent this, once a FSMO role has been seized, the original role holder must never be brought back online. If the hardware is restored, it should be reinstalled and rejoined as a new domain controller rather than reintroduced with its previous Active Directory database.

Role seizure is a powerful but dangerous tool, use it only when the failure is permanent and recovery is impossible.

Steps:

Open an elevated Command Prompt on the target domain controller.

* Type:
* ntdsutil
* Enter FSMO maintenance mode:
* roles
* Connect to the target DC:
* connections
* connect to server <NewDCName>
* quit
* *(Replace <NewDCName> with the hostname of the DC taking over the roles.)*
* (Optional) View current FSMO role assignments:
* select operation target
* list roles for connected server
* quit
* Seize the required FSMO roles using the appropriate commands:
  + seize schema master
  + seize naming master
  + seize rid master
  + seize pdc
  + seize infrastructure master
* *(Only run the commands for the roles that need to be seized.)*
* Exit NTDSUTIL:
* quit, quit

After Seizing the Roles:

Verify new FSMO assignments:

* netdom query fsmo

## DNS Zone Recovery

This procedure defines how to back up and restore the DNS Server Zones. Although its possible to backup DNS Server configurations its only possible to restore manually.

### Backup Procedure

The DNS backup [script](#_AD_Object_Reference) performs a full export of DNS zones and server configuration each day of the week.

### Prepare backup directory

* Ensure $backupPath is defined in the parent backup script (e.g. C:\Backups\ADBackup.ps1).
* The script automatically creates a daily subfolder under DNS\<DayOfWeek>.
* Schedule with a Service Account with Domain Admins
* User Rights Assignments for Logon As A Service and Logon as Batch are required.

### DNS Server Restore

Restore the DNS Server

If restoring to a new server:

* Install the DNS Server role
* Install-WindowsFeature DNS -IncludeManagementTools.
* Allow AD synchronization to restore DNS records to local DC.

### Restore DNS zones

Restore the DNS Zone

If a DNS zone becomes corrupted, loses records, or is accidentally deleted, it must be restored from a recent backup

* Copy the C:\ADBackup\DNS\<DayofWeek>\Zone.name to C:\Windows\System32\DNS.
* From CMD Type the following
* Dnscmd /zoneadd zone.name /primary /file zone.name
* Open the DNS Snap-in and for each Zone
* Click Type > Change and enable Store the zone in Active Directory
* Select Secure on the Dynamic Updates drop-down

## GPO Recovery

Starting with Windows Server 2008 R2, Microsoft moved away from FRS replication system to Distributed File System Replication (DFSR) service for handling SYSVOL replication between domain controllers.

When planning disaster recovery, you need to account for both the AD portion (which gets handled through normal DC backup and recovery procedures) and the SYSVOL files themselves.

A corrupted or out-of-sync SYSVOL can render your entire GPO infrastructure useless even if Active Directory is perfectly healthy, which is why regular backups of the SYSVOL

### GPO Backup Overview

GPO data exists in two parts:

* Active Directory component — stored in CN=Policies,CN=System,<domain>
* SYSVOL file component — stored in \\<domain>\SYSVOL\<domain>\Policies\<GPO\_GUID>

To preserve both components, GPOs must be backed up using the Backup-GPO PowerShell cmdlet. This command exports GPO settings into a portable XML structure containing both AD and SYSVOL data.

### GPO Backup Procedure

Script Description

The backup script automates the daily export of all GPOs to a designated path (C:\ADBackups\GPO\dayofweek) creating a clean snapshot of every policy in the domain.

Executes a full backup of all GPOs using:

* Scheduled daily to retain a rolling weeks worth of GPO backups.
* Backup-GPO -All -Path $gpoBackupPath
* Records backup metadata (name, GUID, creation time) into GPOBackupReference.csv for audit and recovery reference.

### Script Summary

try {

# Clean existing backup directory

if (Test-Path -Path $gpoBackupPath) {

Get-ChildItem -Path $gpoBackupPath -Recurse | Remove-Item -Recurse -Force -ErrorAction SilentlyContinue

}

# Perform full GPO backup

$gpoBackupResult = Backup-GPO -All -Path $gpoBackupPath -ErrorAction Stop

$gpoCount = ($gpoBackupResult | Measure-Object).Count

# Export reference file

$gpoBackupResult | Select-Object DisplayName, GpoId, BackupDirectory, CreationTime |

Export-Csv -Path "$gpoBackupPath\GPOBackupReference.csv" -NoTypeInformation

Write-Host "GPO backup complete: $gpoCount GPOs backed up" -ForegroundColor Green

}

catch {

Write-Error "Failed to backup GPOs: $\_"

}

### GPO Restoration Procedure

If you suspect a GPO is corrupted, deleted or made unauthorised changes, remove the GPO from the Group Policy Management Console to prevent further policy replication or application issues.

Process:

* Open GPO Management and navigate to Group Policy Objects
* Make a note of the Links locations if available.
* Make a note of
* Delete the GPO
* Right click Group Policy Objects > Manage Backups…
* Browse to C:\ADBackups\GPO\dayofweek and the last known good for that GPO
* Select GPO and Restore
* Restore Links using the [Object Reference script](#_AD_Object_Reference) or use [GPO Restore Links script](#_Restore_GPO_Link)

## SYSVOL DFSR Recovery

SYSVOL replication issues can prevent Group Policy from applying correctly across the domain, leading to inconsistent user and computer configurations. When SYSVOL becomes corrupted, out of sync, or replication halts entirely, domain controllers may hold differing versions of policy data, some with outdated or missing GPOs. This can manifest as Group Policy processing errors, missing logon scripts, or inconsistent security settings.

In such cases, a restore may be required.

A non-authoritative restore allows a domain controller to resynchronize its SYSVOL from a healthy partner, effectively discarding its local copy.

An authoritative restore, on the other hand, designates a specific DC’s SYSVOL as the master source, forcing other DCs to replicate from it. Selecting the right approach depends on where the corruption originated and which copy of SYSVOL is known to be correct.

### D2 and D4 References by Microsoft

Microsoft still insists on using the old *D2* and *D4* terminology from the FRS era when describing DFSR SYSVOL recovery. The actual flags haven’t existed since Server 2008, but apparently Redmond can’t quite let go of the good old days of File Replication Service.

### Non-Authoritative Restore

Use when a single DC's SYSVOL is corrupted but other DCs have good copies.

* The restored DC will receive updated SYSVOL data from replication partners.
* SYSVOL contents are overwritten by authoritative sources.
* Prefer the PDC Emulator as the authoritative source if it has the most up-to-date SYSVOL contents.
* All ADSIEdit attribute edits must replicate via AD replication to all DCs before starting DFSR on those DCs. Use repadmin /syncall /AdeP to force replication.
* Monitor the DFSR event log (Applications and Services Logs → DFS Replication) for the event IDs mentioned below.

### Authoritative Restore

Use when all DCs have corrupted SYSVOL or you need to force specific content across the domain.

* The restored DC becomes the authoritative source
* Other DCs receive SYSVOL data from the authoritative DC

### Non-Authoritative SYSVOL Restore (D2)

Use this when a single DC’s SYSVOL is bad and you want it to rebuild from peers.

Steps (per DC you want to repair):

* Open ADSIEdit.msc and connect to the default naming context.
* Navigate to the SYSVOL subscription object for the target DC:
* CN=SYSVOL Subscription,CN=Domain System Volume,CN=DFSR-LocalSettings,
* CN=<ServerName>,OU=Domain Controllers,DC=<yourdomain>,DC=<tld>
  + Set the attribute msDFSR-Enabled = FALSE on that DN.
* Force AD replication to propagate that change to other DCs:
  + repadmin /syncall /AdeP
* On the target DC, run:
* dfsrdiag pollad
  + You should see Event ID 4114 in the DFSR log (sysvol replication stopped).
* On the same DN in ADSIEdit, set msDFSR-Enabled = TRUE.
* Force AD replication again:
* repadmin /syncall /AdeP
* On the target DC, run:
* dfsrdiag pollad
  + You should see Event ID 4614 and 4604 indicating SYSVOL has been initialized and a D2 (non-authoritative sync) occurred.
* Verify SYSVOL shares, GPO contents and DFSR health (Get-DfsrBacklog, Get-DfsrState, repadmin /showrepl).

### Authoritative SYSVOL Restore (D4)

Use this when other DCs are corrupt or missing SYSVOL and you have a single known-good copy.

High level: stop/start DFSR, mark authoritative in AD on one DC (msDFSR-Options=1), mark others non-authoritative, force AD replication, start DFSR on authoritative server, flip flags and start others.

Steps:

* Stop DFSR service on all DCs in the domain and set Startup Type to Manual (prevents races):
* net stop dfsr
* sc config dfsr start= demand
* On the DC you want to be authoritative (preferably PDCe), in ADSIEdit edit the SYSVOL subscription DN:
* CN=SYSVOL Subscription,CN=Domain System Volume,CN=DFSR-LocalSettings,
* CN=<AuthServer>,OU=Domain Controllers,DC=<yourdomain>,DC=<tld>
  + Set msDFSR-Enabled = FALSE
  + Set msDFSR-Options = 1 *(this marks it authoritative — the D4 flag)*
* On every other DC in the domain, edit the same DN and set:
  + msDFSR-Enabled = FALSE
* Force AD replication and validate it succeeded across all DCs:
* repadmin /syncall /AdeP
* repadmin /showrepl
  + - wait until replication has propagated to all DCs.
* Start DFSR on the authoritative DC:
* net start dfsr
* dfsrdiag pollad
  + Expect Event ID 4114 (replication stopped — normal after the stop), then later Event ID 4602 when the authoritative initialization completes.
* On the authoritative DC set msDFSR-Enabled = TRUE (ADSIEdit) and force AD replication again:
* repadmin /syncall /AdeP
* dfsrdiag pollad
  + Event ID 4602 should appear indicating initialization (D4 completed).
* Start DFSR on each non-authoritative DC. On each, you will see Event ID 4114 indicating replication paused (expected). Then on each non-authoritative DC, set msDFSR-Enabled = TRUE in ADSIEdit (or you may perform this step before starting the service depending on your ordering), force AD replication, and run:
* dfsrdiag pollad
  + These DCs will now replicate the authoritative SYSVOL contents from the D4 source.
* When all DCs report successful sync and Event IDs indicating initialization complete, return DFSR service startup type to Automatic on all DCs:
* sc config dfsr start= auto
* Verify:
  + repadmin /replsummary and repadmin /showrepl show healthy replication.
  + Get-DfsrBacklog -GroupName "Domain System Volume" -SourceComputer <AuthDC> -DestinationComputer <DC> shows no backlogs.
  + Confirm \\domain\SYSVOL content and GPOs match across DCs.
  + Monitor DFSR Event Log for absence of 2213 (dirty shutdown) or persistent 5002/5004 errors.

Important cautions & troubleshooting tips

* AD replication is mandatory between the ADSIEdit edits and any DFSR start/poll—if the AD attribute changes don’t replicate, the process fails. Use repadmin /showrepl to confirm propagation.
* Prefer the PDC Emulator as the authoritative source since it typically has the most recent SYSVOL contents.
* If you encounter Event ID 2213 (dirty shutdown), follow Microsoft guidance for resuming replication (backup DFSR database, ResumeReplication as documented).
* If more than one DC is affected, expand the steps to include all affected DCs (the KB assumes a 2-DC example).
* Do not skip the dfsrdiag pollad and repadmin /syncall /AdeP cycles—these force DCs to pick up the ADSI edits.
* Always test in an isolated lab before running in production if possible.

## Why Schema Rollbacks Are Not Possible

Schema Version Numbers Are Monotonic

The Active Directory schema uses version numbers that only increment forward. Each schema modification increases the schema version number, and this version is replicated across all domain controllers in the forest.

When third-party applications or in-house solutions extend the Active Directory schema, they must define new attributes and classes using unique Object Identifiers (OIDs). OIDs are globally unique numerical identifiers arranged in a hierarchical tree, with the Internet Assigned Numbers Authority (IANA) assigning root branches to organizations. Microsoft’s own products, such as Exchange Server and Configuration Manager, use OIDs officially registered within Microsoft’s allocated namespace.

Problems arise when vendors, or internal developers, ignore proper registration and instead pick arbitrary numbers, reuse examples from documentation, or worse, borrow OIDs from Microsoft’s reserved ranges. This kind of shortcut creates a silent but serious landmine.

When the organization later tries to deploy a product like Exchange, SCCM, or a Windows Server feature pack that legitimately requires those same OID values, a schema conflict occurs. Active Directory will reject the schema update because the OID already exists but points to a different definition. The schema, being a forest-wide, append-only structure, doesn’t allow deletions or OID reuse, even if the offending attribute or class is deactivated.

At that point, the organization is trapped: they can’t install the required Microsoft product, and they can’t remove the conflicting schema objects to fix it. In less severe cases, a limited workaround might be possible by renaming or isolating the custom extensions. But when the conflicting OIDs are deeply embedded in production systems, the only practical fix is a complete forest rebuild, creating a new forest, migrating all accounts, groups, and data, and re-establishing trust relationships.

## USN (Update Sequence Number) by 100,000

It’s no longer necessary to arbitrarily increase the USN (Update Sequence Number) by 100,000 during a disaster recovery in modern Active Directory.

#### Purpose of USN

The USN is a per-DC counter that tracks changes in the Active Directory database. When an object is modified, its USN increments. During replication, DCs compare USNs to determine which changes are newer and need to be replicated.

Authoritative Restore Behaviour

When performing an authoritative restore:

* ntdsutil automatically marks restored objects as authoritative, which internally updates the USNs and version numbers for the restored objects.
* Replication partners treat these objects as newer than existing copies, so the restored data propagates across the forest.
* There is no manual need to “jump” USNs by a large number, the system handles this for you.

#### Risks of Arbitrary USN Modification

* Manually increasing USNs outside of supported methods can break replication logic, cause update collisions, or confuse replication partners.
* Microsoft documentation explicitly recommends using ntdsutil’s authoritative restore, which safely manages USN and version stamping.

Historical Context

* Older guides sometimes mention incrementing USNs by a large number (like 100,000) in very early versions of Windows Server (2000/2003).
* In modern versions (2008 R2 and later), ntdsutil handles it automatically; manual USN manipulation is unnecessary and unsafe.

## Tombstone period of 180 days

Active Directory tombstones are deleted objects retained in the directory database for a default of 180 days (previously 60 days in older Windows Server versions). When an object, such as a user, group, or computer, is deleted, AD doesn’t immediately remove it. Instead, it marks the object as tombstoned by clearing most attributes and setting the isDeleted flag. Tombstones allow the deletion event to replicate to all domain controllers, ensuring consistency across the forest.

After the 180-day tombstone lifetime, these objects are permanently purged during garbage collection. The retention period is controlled by the tombstoneLifetime attribute located under:  
CN=Directory Service,CN=Windows NT,CN=Services,CN=Configuration,DC=<domain>.

Once purged, the only recovery option is via a system state backup taken before the deletion occurred.

If the Active Directory Recycle Bin is enabled (Windows Server 2008 R2 and later), deleted objects first enter a Deleted Objects container in a logically deleted state, retaining all linked attributes. They can then be easily restored without requiring an authoritative restore or system state recovery, provided the tombstone lifetime has not expired.

## Backup and Recovery Scripts

### AD Object Reference Script

<#

.SYNOPSIS

Daily Active Directory backup script for DNS zones, GPOs, and AD objects.

.DESCRIPTION

Creates daily backups organized by day of week for DNS zones, DNS server configuration,

Group Policy Objects, and Active Directory object inventory including users, computers, groups, and OUs.

.NOTES

Run this script on a Domain Controller with appropriate permissions.

Requires Enterprise/Domain Admin privileges for full backup capability.

Version: 0.2

#>

[CmdletBinding()]

param()

# Configuration

$backupPath = "C:\ADBackups"

$dayOfWeek = (Get-Date).DayOfWeek

$services = @("DNS", "GPO", "ADObjects")

$dnsSystemPath = "$env:SystemRoot\System32\dns"

# Initialize backup directory structure

foreach ($service in $services) {

$targetPath = Join-Path -Path $backupPath -ChildPath "$service\$dayOfWeek"

if (-not (Test-Path -Path $targetPath)) {

try {

New-Item -Path $targetPath -ItemType Directory -Force -ErrorAction Stop | Out-Null

Write-Verbose "Created directory: $targetPath"

}

catch {

Write-Error "Failed to create backup directory $targetPath : $\_"

exit 1

}

}

}

# DNS Zone and Configuration Backup

Write-Host "Starting DNS backup..." -ForegroundColor Cyan

$dnsBackupPath = Join-Path -Path $backupPath -ChildPath "DNS\$dayOfWeek"

$dnsConfigFile = Join-Path -Path $dnsBackupPath -ChildPath "DNSServerConfig.txt"

try {

# Initialize DNS configuration output

$dnsConfig = @()

$dnsConfig += "=" \* 80

$dnsConfig += "DNS Server Configuration Backup"

$dnsConfig += "Server: $env:COMPUTERNAME"

$dnsConfig += "Backup Date: $(Get-Date -Format 'yyyy-MM-dd HH:mm:ss')"

$dnsConfig += "=" \* 80

$dnsConfig += ""

# Export DNS server settings using dnscmd

Write-Host " Exporting DNS server configuration..." -ForegroundColor Yellow

# Server Info

$dnsConfig += "#" \* 80

$dnsConfig += "DNS SERVER INFORMATION"

$dnsConfig += "#" \* 80

$serverInfo = dnscmd . /info 2>&1

$dnsConfig += $serverInfo

$dnsConfig += ""

# Forwarders

$dnsConfig += "#" \* 80

$dnsConfig += "FORWARDERS"

$dnsConfig += "#" \* 80

try {

$forwarders = Get-DnsServerForwarder -ErrorAction Stop

$dnsConfig += "Forwarder IP Addresses:"

foreach ($fwd in $forwarders.IPAddress) {

$dnsConfig += " $fwd"

}

$dnsConfig += "Use Root Hints: $($forwarders.UseRootHint)"

$dnsConfig += "Timeout: $($forwarders.Timeout) seconds"

$dnsConfig += "Enable Reordering: $($forwarders.EnableReordering)"

}

catch {

$dnsConfig += "Could not retrieve forwarders: $\_"

}

$dnsConfig += ""

# Alternative forwarders using dnscmd

$dnsConfig += "FORWARDERS (via dnscmd):"

$forwardersCmd = dnscmd . /info /Forwarders 2>&1

$dnsConfig += $forwardersCmd

$dnsConfig += ""

# Root Hints

$dnsConfig += "#" \* 80

$dnsConfig += "ROOT HINTS"

$dnsConfig += "#" \* 80

try {

$rootHints = Get-DnsServerRootHint -ErrorAction Stop

$dnsConfig += "Root Hint Servers:"

foreach ($hint in $rootHints) {

$dnsConfig += " Name Server: $($hint.NameServer.RecordData.NameServer)"

foreach ($ip in $hint.IPAddress) {

$dnsConfig += " IP: $($ip.RecordData.IPv4Address)$($ip.RecordData.IPv6Address)"

}

}

}

catch {

$dnsConfig += "Could not retrieve root hints: $\_"

}

$dnsConfig += ""

# Alternative root hints using dnscmd

$dnsConfig += "ROOT HINTS (via dnscmd):"

$rootHintsCmd = dnscmd . /info /RootHints 2>&1

$dnsConfig += $rootHintsCmd

$dnsConfig += ""

# Recursion Settings

$dnsConfig += "#" \* 80

$dnsConfig += "RECURSION SETTINGS"

$dnsConfig += "#" \* 80

try {

$recursion = Get-DnsServerRecursion -ErrorAction Stop

$dnsConfig += "Recursion Enabled: $($recursion.Enable)"

$dnsConfig += "Additional Timeout: $($recursion.AdditionalTimeout) seconds"

$dnsConfig += "Retry Interval: $($recursion.RetryInterval) seconds"

$dnsConfig += "Timeout: $($recursion.Timeout) seconds"

$dnsConfig += "Secure Response: $($recursion.SecureResponse)"

}

catch {

$dnsConfig += "Could not retrieve recursion settings: $\_"

}

$dnsConfig += ""

# Alternative recursion using dnscmd

$dnsConfig += "RECURSION (via dnscmd):"

$recursionCmd = dnscmd . /info /Recursion 2>&1

$dnsConfig += $recursionCmd

$dnsConfig += ""

# Scavenging Configuration

$dnsConfig += "#" \* 80

$dnsConfig += "SCAVENGING CONFIGURATION"

$dnsConfig += "#" \* 80

try {

$scavenging = Get-DnsServerScavenging -ErrorAction Stop

$dnsConfig += "Scavenging Enabled: $($scavenging.ScavengingState)"

$dnsConfig += "Scavenging Interval: $($scavenging.ScavengingInterval)"

$dnsConfig += "No-Refresh Interval: $($scavenging.NoRefreshInterval)"

$dnsConfig += "Refresh Interval: $($scavenging.RefreshInterval)"

$dnsConfig += "Last Scavenge Time: $($scavenging.LastScavengeTime)"

}

catch {

$dnsConfig += "Could not retrieve scavenging settings: $\_"

}

$dnsConfig += ""

# Alternative scavenging using dnscmd

$dnsConfig += "SCAVENGING (via dnscmd):"

$scavengingCmd = dnscmd . /info /ScavengingInterval 2>&1

$dnsConfig += $scavengingCmd

$dnsConfig += ""

# Server-Level Settings

$dnsConfig += "#" \* 80

$dnsConfig += "SERVER-LEVEL SETTINGS"

$dnsConfig += "#" \* 80

try {

$server = Get-DnsServer -ErrorAction Stop

$dnsConfig += "Server Name: $($server.ServerSetting.ComputerName)"

$dnsConfig += "Listen Addresses: $($server.ServerSetting.ListenAddresses -join ', ')"

$dnsConfig += "Round Robin: $($server.ServerSetting.RoundRobin)"

$dnsConfig += "Local Net Priority: $($server.ServerSetting.LocalNetPriority)"

$dnsConfig += "Bind Secondaries: $($server.ServerSetting.BindSecondaries)"

$dnsConfig += "Strict File Parsing: $($server.ServerSetting.StrictFileParsing)"

$dnsConfig += "Enable DNSSEC: $($server.ServerSetting.EnableDnsSec)"

$dnsConfig += "Enable EDNS Probes: $($server.ServerSetting.EnableEDnsProbes)"

$dnsConfig += "Forwarder Timeout: $($server.ServerSetting.ForwardingTimeout)"

$dnsConfig += "Cache Pollution Protection: $($server.ServerSetting.EnablePollutionProtection)"

$dnsConfig += "Default Aging State: $($server.ServerSetting.DefaultAgingState)"

$dnsConfig += "Default Refresh Interval: $($server.ServerSetting.DefaultRefreshInterval)"

$dnsConfig += "Default No-Refresh Interval: $($server.ServerSetting.DefaultNoRefreshInterval)"

}

catch {

$dnsConfig += "Could not retrieve server settings: $\_"

}

$dnsConfig += ""

# Advanced Server Settings via dnscmd

$dnsConfig += "#" \* 80

$dnsConfig += "ADVANCED SERVER SETTINGS"

$dnsConfig += "#" \* 80

$advancedSettings = @(

"/BootMethod",

"/EnableGlobalQueryBlockList",

"/GlobalQueryBlockList",

"/EventLogLevel",

"/LogLevel",

"/LogFilePath",

"/MaxCacheTTL",

"/MaxNegativeCacheTTL",

"/SendPort",

"/WriteAuthorityNS",

"/SecureResponses",

"/RpcProtocol",

"/NameCheckFlag"

)

foreach ($setting in $advancedSettings) {

$dnsConfig += ""

$dnsConfig += "Setting: $setting"

$settingValue = dnscmd . /info $setting 2>&1

$dnsConfig += $settingValue

}

$dnsConfig += ""

# Zone List with Details

$dnsConfig += "#" \* 80

$dnsConfig += "ZONE LIST WITH CONFIGURATION"

$dnsConfig += "#" \* 80

$zones = Get-DnsServerZone -ErrorAction Stop

foreach ($zone in $zones) {

$dnsConfig += ""

$dnsConfig += "Zone: $($zone.ZoneName)"

$dnsConfig += " Type: $($zone.ZoneType)"

$dnsConfig += " Dynamic Update: $($zone.DynamicUpdate)"

$dnsConfig += " Replication Scope: $($zone.ReplicationScope)"

$dnsConfig += " DS Integrated: $($zone.IsDsIntegrated)"

$dnsConfig += " Auto Created: $($zone.IsAutoCreated)"

$dnsConfig += " Paused: $($zone.IsPaused)"

$dnsConfig += " Reverse Lookup: $($zone.IsReverseLookupZone)"

$dnsConfig += " Signed: $($zone.IsSigned)"

$dnsConfig += " Secure Secondaries: $($zone.SecureSecondaries)"

if ($zone.NotifyServers) {

$dnsConfig += " Notify Servers: $($zone.NotifyServers -join ', ')"

}

if ($zone.MasterServers) {

$dnsConfig += " Master Servers: $($zone.MasterServers.IPAddressToString -join ', ')"

}

}

$dnsConfig += ""

# Save DNS configuration to file

$dnsConfig | Out-File -FilePath $dnsConfigFile -Encoding UTF8 -Force

Write-Host " DNS server configuration saved to: DNSServerConfig.txt" -ForegroundColor Green

# Backup DNS Zone Files

Write-Host " Backing up DNS zone files..." -ForegroundColor Yellow

$dnsZones = Get-DnsServerZone -ErrorAction Stop | Where-Object { -not $\_.IsAutoCreated }

$zoneCount = 0

foreach ($zone in $dnsZones) {

$zoneName = $zone.ZoneName

try {

# Export zone to file

Export-DnsServerZone -Name $zoneName -FileName $zoneName -ErrorAction Stop

$zoneCount++

Write-Verbose "Exported DNS zone: $zoneName"

}

catch {

Write-Warning "Failed to export DNS zone '$zoneName': $\_"

}

}

# Copy all DNS files to backup location

try {

Copy-Item -Path "$dnsSystemPath\\*" -Destination $dnsBackupPath -Force -ErrorAction Stop

Write-Host " DNS zone files copied to backup" -ForegroundColor Green

}

catch {

Write-Warning "Failed to copy some DNS files: $\_"

}

Write-Host "DNS backup complete: $zoneCount zones + server configuration backed up" -ForegroundColor Green

}

catch {

Write-Error "Failed to backup DNS: $\_"

}

# Group Policy Backup

Write-Host "Starting GPO backup..." -ForegroundColor Cyan

$gpoBackupPath = Join-Path -Path $backupPath -ChildPath "GPO\$dayOfWeek"

try {

# Clean previous GPO backups for this day

if (Test-Path -Path $gpoBackupPath) {

Get-ChildItem -Path $gpoBackupPath -Recurse | Remove-Item -Recurse -Force -ErrorAction SilentlyContinue

}

# Backup all GPOs

$gpoBackupResult = Backup-GPO -All -Path $gpoBackupPath -ErrorAction Stop

$gpoCount = ($gpoBackupResult | Measure-Object).Count

# Save GPO backup reference

$gpoBackupResult | Select-Object DisplayName, GpoId, BackupDirectory, CreationTime |

Export-Csv -Path "$gpoBackupPath\GPOBackupReference.csv" -NoTypeInformation

Write-Host "GPO backup complete: $gpoCount GPOs backed up" -ForegroundColor Green

}

catch {

Write-Error "Failed to backup GPOs: $\_"

}

# Active Directory Objects Inventory

Write-Host "Starting AD objects inventory..." -ForegroundColor Cyan

$adObjectsPath = Join-Path -Path $backupPath -ChildPath "ADObjects\$dayOfWeek"

$adInventoryFile = Join-Path -Path $adObjectsPath -ChildPath "ADObjects.txt"

try {

# Initialize output file

$output = @()

$output += "=" \* 80

$output += "Active Directory Objects Inventory - $(Get-Date -Format 'yyyy-MM-dd HH:mm:ss')"

$output += "Domain: $((Get-ADDomain).DNSRoot)"

$output += "=" \* 80

$output += ""

# Computers

$output += "#" \* 80

$output += "COMPUTERS"

$output += "#" \* 80

$computers = Get-ADComputer -Filter \* -Properties Enabled, OperatingSystem, LastLogonDate |

Select-Object Name, DistinguishedName, Enabled, OperatingSystem, LastLogonDate

$output += $computers | Format-Table -AutoSize | Out-String

$output += "Total Computers: $(($computers | Measure-Object).Count)"

$output += ""

# Users

$output += "#" \* 80

$output += "USERS"

$output += "#" \* 80

$users = Get-ADUser -Filter \* -Properties Enabled, EmailAddress, LastLogonDate |

Select-Object Name, DistinguishedName, Enabled, EmailAddress, LastLogonDate

$output += $users | Format-Table -AutoSize | Out-String

$output += "Total Users: $(($users | Measure-Object).Count)"

$output += ""

# Groups

$output += "#" \* 80

$output += "GROUPS"

$output += "#" \* 80

$groups = Get-ADGroup -Filter \* -Properties GroupScope, GroupCategory |

Select-Object Name, DistinguishedName, GroupScope, GroupCategory

$output += $groups | Format-Table -AutoSize | Out-String

$output += "Total Groups: $(($groups | Measure-Object).Count)"

$output += ""

# Group Memberships

$output += "#" \* 80

$output += "GROUP MEMBERSHIPS"

$output += "#" \* 80

foreach ($group in $groups) {

try {

$members = Get-ADGroupMember -Identity $group.DistinguishedName -ErrorAction SilentlyContinue |

Select-Object Name, DistinguishedName, ObjectClass

if ($members) {

$output += "`nGroup: $($group.Name)"

$output += "-" \* 80

$output += $members | Format-Table -AutoSize | Out-String

}

}

catch {

Write-Verbose "Could not retrieve members for group: $($group.Name)"

}

}

$output += ""

# Organizational Units

$output += "#" \* 80

$output += "ORGANIZATIONAL UNITS"

$output += "#" \* 80

$ous = Get-ADOrganizationalUnit -Filter \* -Properties Description |

Select-Object Name, DistinguishedName, Description

$output += $ous | Format-Table -AutoSize | Out-String

$output += "Total OUs: $(($ous | Measure-Object).Count)"

$output += ""

# GPO Information

$output += "#" \* 80

$output += "GROUP POLICY OBJECTS"

$output += "#" \* 80

$gpos = Get-GPO -All | Select-Object DisplayName, Id, CreationTime, ModificationTime

$output += $gpos | Format-Table -AutoSize | Out-String

$output += "Total GPOs: $(($gpos | Measure-Object).Count)"

$output += ""

# GPO Permissions (Apply Rights)

Write-Host " Capturing GPO permissions..." -ForegroundColor Yellow

$output += "#" \* 80

$output += "GPO PERMISSIONS (APPLY RIGHTS)"

$output += "#" \* 80

$gpoPermissions = @()

foreach ($gpo in $gpos) {

try {

$applyPermissions = Get-GPPermissions -Name $gpo.DisplayName -All -ErrorAction SilentlyContinue |

Where-Object { $\_.Permission -match "GpoApply" }

if ($applyPermissions) {

foreach ($perm in $applyPermissions) {

$gpoPermissions += [PSCustomObject]@{

GPOName = $gpo.DisplayName

Trustee = $perm.Trustee.Name

TrusteeType = $perm.Trustee.SidType

Permission = $perm.Permission

Inherited = $perm.Inherited

}

}

}

}

catch {

Write-Verbose "Could not retrieve permissions for GPO: $($gpo.DisplayName)"

}

}

if ($gpoPermissions.Count -gt 0) {

$output += $gpoPermissions | Format-Table -AutoSize | Out-String

$output += "Total GPO Apply Permissions: $(($gpoPermissions | Measure-Object).Count)"

}

else {

$output += "No GPO apply permissions found or permissions could not be retrieved"

}

$output += ""

# GPO Links

$output += "#" \* 80

$output += "GPO LINKS BY OU"

$output += "#" \* 80

foreach ($ou in $ous) {

try {

$gpoLinks = Get-GPInheritance -Target $ou.DistinguishedName -ErrorAction SilentlyContinue

if ($gpoLinks.GpoLinks) {

$output += "`nOU: $($ou.DistinguishedName)"

$output += "-" \* 80

$output += $gpoLinks.GpoLinks | Select-Object DisplayName, Enabled, Enforced, Order |

Format-Table -AutoSize | Out-String

}

}

catch {

Write-Verbose "Could not retrieve GPO links for OU: $($ou.DistinguishedName)"

}

}

# Write to file

$output | Out-File -FilePath $adInventoryFile -Encoding UTF8 -Force

# Export structured data as CSV for easier parsing

$computers | Export-Csv -Path "$adObjectsPath\Computers.csv" -NoTypeInformation

$users | Export-Csv -Path "$adObjectsPath\Users.csv" -NoTypeInformation

$groups | Export-Csv -Path "$adObjectsPath\Groups.csv" -NoTypeInformation

$ous | Export-Csv -Path "$adObjectsPath\OUs.csv" -NoTypeInformation

$gpos | Export-Csv -Path "$adObjectsPath\GPOs.csv" -NoTypeInformation

$gpoPermissions | Export-Csv -Path "$adObjectsPath\GPOPermissions.csv" -NoTypeInformation

Write-Host "AD inventory complete: Data saved to $adObjectsPath" -ForegroundColor Green

}

catch {

Write-Error "Failed to create AD objects inventory: $\_"

}

# Summary

Write-Host ("=" \* 80) -ForegroundColor Yellow

Write-Host "Backup Summary - $dayOfWeek" -ForegroundColor Yellow

Write-Host ("=" \* 80) -ForegroundColor Yellow

Write-Host "Backup Location: $backupPath\\*\$dayOfWeek" -ForegroundColor White

Write-Host "DNS Zones: $zoneCount backed up" -ForegroundColor White

Write-Host "DNS Server Config: Complete (forwarders, recursion, scavenging, server settings)" -ForegroundColor White

Write-Host "GPOs: $gpoCount backed up" -ForegroundColor White

Write-Host "GPO Permissions: $(($gpoPermissions | Measure-Object).Count) apply rights captured" -ForegroundColor White

Write-Host "AD Objects: Inventory completed" -ForegroundColor White

Write-Host ("=" \* 80) -ForegroundColor Yellow

### Restore GPO Link to OU

#Requires -Version 5.1

#Requires -Modules ActiveDirectory, GroupPolicy

<#

.SYNOPSIS

Restore GPO links based on backup inventory.

.DESCRIPTION

Takes a GPO name and relinks it to all OUs that had it linked in the backup.

.PARAMETER BackupPath

Path to the AD backups directory. Default: C:\ADBackups

.PARAMETER GPOName

Name of the GPO to restore links for.

.EXAMPLE

# Restore all links for a specific GPO

.\Restore-GPOLinks.ps1 -GPOName "Security Policy"

# Use a different backup location

.\Restore-GPOLinks.ps1 -GPOName "Desktop Settings" -BackupPath "D:\Backups"

#>

[CmdletBinding()]

param(

[Parameter()]

[string]$BackupPath = "C:\ADBackups",

[Parameter(Mandatory)]

[string]$GPOName

)

# Find the most recent backup

$daysOfWeek = @("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")

$currentDay = [int](Get-Date).DayOfWeek

$backupFile = $null

for ($i = 0; $i -le 6; $i++) {

$dayIndex = ($currentDay - $i) % 7

if ($dayIndex -lt 0) { $dayIndex += 7 }

$checkDay = $daysOfWeek[$dayIndex]

$testPath = "$BackupPath\ADObjects\$checkDay\ADObjects.txt"

if (Test-Path $testPath) {

$backupFile = $testPath

Write-Host "Using backup from: $checkDay`n" -ForegroundColor Green

break

}

}

if (-not $backupFile) {

Write-Error "No backup files found in $BackupPath"

exit 1

}

# Read the backup file

$content = Get-Content $backupFile -Raw

# Find the GPO Links section

$gpoLinkSection = ($content -split "GPO LINKS BY OU")[1]

if (-not $gpoLinkSection) {

Write-Error "Could not find GPO Links section in backup file"

exit 1

}

# Parse OUs that had this GPO linked

$ouSections = $gpoLinkSection -split "`nOU:" | Where-Object { $\_ -match $GPOName }

$ousToLink = @()

foreach ($section in $ouSections) {

$lines = $section -split "`n"

$ouDN = $lines[0].Trim()

# Verify this section actually contains our GPO

if ($section -match $GPOName) {

$ousToLink += $ouDN

}

}

if ($ousToLink.Count -eq 0) {

Write-Warning "GPO '$GPOName' was not linked to any OUs in the backup"

exit 0

}

# Display what will be linked

Write-Host "Found GPO '$GPOName' linked to $($ousToLink.Count) OU(s):`n" -ForegroundColor Cyan

foreach ($ou in $ousToLink) {

Write-Host " - $ou" -ForegroundColor Gray

}

Write-Host ""

# Confirm

$confirm = Read-Host "Proceed with relinking? (Y/N)"

if ($confirm -notmatch '^[Yy]') {

Write-Host "Cancelled" -ForegroundColor Yellow

exit 0

}

# Verify GPO exists

try {

$gpo = Get-GPO -Name $GPOName -ErrorAction Stop

Write-Host "`nGPO found: $($gpo.DisplayName)" -ForegroundColor Green

}

catch {

Write-Error "GPO '$GPOName' does not exist in the domain"

exit 1

}

# Link the GPO to each OU

Write-Host "`nRelinking GPO..." -ForegroundColor Cyan

$successCount = 0

$failCount = 0

foreach ($ouDN in $ousToLink) {

# Verify OU exists

try {

$ou = Get-ADOrganizationalUnit -Identity $ouDN -ErrorAction Stop

}

catch {

Write-Warning "OU not found: $ouDN - Skipping"

$failCount++

continue

}

# Check if link already exists

try {

$existingLinks = Get-GPInheritance -Target $ouDN -ErrorAction Stop

$linkExists = $existingLinks.GpoLinks | Where-Object { $\_.DisplayName -eq $GPOName }

if ($linkExists) {

Write-Host " Already linked: $ouDN" -ForegroundColor Yellow

$successCount++

continue

}

}

catch {

Write-Warning "Could not check existing links for: $ouDN"

}

# Create the link

try {

New-GPLink -Name $GPOName -Target $ouDN -LinkEnabled Yes -ErrorAction Stop | Out-Null

Write-Host " Linked: $ouDN" -ForegroundColor Green

$successCount++

}

catch {

Write-Warning "Failed to link to: $ouDN - $\_"

$failCount++

}

}

# Summary

Write-Host "`n"

Write-Host ("=" \* 80) -ForegroundColor Cyan

Write-Host "Summary" -ForegroundColor Cyan

Write-Host ("=" \* 80) -ForegroundColor Cyan

Write-Host "GPO: $GPOName" -ForegroundColor White

Write-Host "OUs Processed: $($ousToLink.Count)" -ForegroundColor White

Write-Host "Successfully Linked: $successCount" -ForegroundColor Green

Write-Host "Failed: $failCount" -ForegroundColor $(if ($failCount -gt 0) { "Red" } else { "White" })

Write-Host ("=" \* 80) -ForegroundColor Cyan