ADMIN:DOCS

Windows Server 2025 Core: DNS-Server mit PowerShell einrichten und verwalten

Eigenständigen DNS-Server unter Windows Server 2025 Core ohne Active Directory aufsetzen: Rolle installieren, dateibasierte Zonen und Records per PowerShell pflegen und systematisch prüfen.

Kategorie: Windows Server **Veröffentlicht:** 12.11.2026

Steht ein frisch bereitgestellter Windows Server 2025 Core erst einmal im Netz, taucht im laufenden Betrieb schnell eine praktische Hürde auf: Interne Dienste, Test-VMs oder Speicherziele sollen nicht mehr über wechselnde oder schwer merkbare IP-Adressen angesprochen werden, sondern verlässlich über eigene Hostnamen. In einer reinen Workgroup-Umgebung ohne Active Directory existiert dafür zunächst keine zentrale Instanz – Namensauflösung lässt sich dort entweder mühsam über lokale `hosts`-Dateien auf jedem einzelnen System pflegen oder sauber über einen eigenen DNS-Dienst im Netz abbilden.

Auf Basis des im Leitfaden Windows Server 2025 Core: Installation und Administration über SSH und PowerShell eingerichteten Systems (`WIN-CORE01`) übernimmt der Server genau diese Aufgabe. Da Server Core bewusst auf eine lokale grafische Oberfläche und Werkzeuge wie den DNS-Manager verzichtet, erfolgt die komplette Einrichtung und Pflege per SSH direkt in der Windows PowerShell.

Dafür wird zunächst die DNS-Serverrolle von Windows Server samt den dazugehörigen Verwaltungswerkzeugen installiert. Sie stellt den eigentlichen `DNS`-Systemdienst sowie das PowerShell-Modul `DnsServer` bereit. Erst über dieses Modul richtest du dateibasierte primäre Zonen ein, verwaltest Resource Records wie `A`, `AAAA`, `CNAME` oder `PTR` und trennst autoritative Antworten sauber von der Weiterleitung externer Anfragen über Forwarder. Entscheidend für den Betrieb ist die anschließende Überprüfung: Jede Änderung am Server muss sich über

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

gezielte Abfragen direkt am Zielport verifizieren lassen, bevor andere Systeme im Netz den Dienst produktiv nutzen.

Zielgruppe und Vorkenntnisse

Dieser Leitfaden richtet sich an Administratoren mit soliden Grundkenntnissen im Systembetrieb, die einen eigenständigen Windows-DNS-Server ohne Active Directory einrichten und über die Kommandozeile betreiben möchten.

Vorausgesetzt werden:

- Grundlagen von IPv4, IPv6, Subnetzen und TCP/IP-Kommunikation
- Konzeptionelles Verständnis von DNS-Namen, Zonen und Resource Records
- Grundkenntnisse in Windows Server Core sowie eine lauffähige Basisinstallation (wie `WIN-CORE01` aus dem vorausgehenden Leitfaden)
- Sicherer Umgang mit der Kommandozeile und bestehender administrativer OpenSSH-Zugang zum Zielsystem

Erfahrungen mit Linux und dem OpenSSH-Client erleichtern den Einstieg bei der Fernverwaltung, sind jedoch nicht zwingend erforderlich. Spezifische Vorkenntnisse mit dem Windows-DNS-Dienst oder dem PowerShell-Modul `DnsServer` werden nicht vorausgesetzt; sämtliche Cmdlets, Zonentypen und Abfragemechanismen werden schrittweise hergeleitet, im Kontext der dateibasierten Architektur eingeordnet und anhand nachvollziehbarer Prüfungen verifiziert.

> □□ **Abgrenzung der Umgebung:** Dieser Guide behandelt ausschließlich einen eigenständigen, dateibasierten DNS-Server in einer Workgroup-Umgebung. Active Directory Domain Services (AD DS), AD-integrierte Zonen, DNSSEC, DNS-over-HTTPS/TLS sowie Clustering oder Secondary-DNS-Replikation sind nicht Bestandteil dieser Anleitung.

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Windows-DNS-Server verstehen

Bevor du die Rolle installierst und erste Zonen anlegst, lohnt sich ein präziser Blick auf das Zusammenspiel aus Dienst, Verwaltungswerkzeugen, Zonenspeicherung und Antwortlogik. Wer diese Ebenen sauber trennt, erkennt später bei Fehlern sofort, ob ein Problem an der Zone, an der Rekursion oder am Netzwerk liegt.

DNS-Serverrolle, Systemdienst und PowerShell-Modul

Unter Windows Server 2025 trennt das Betriebssystem sauber zwischen der installierbaren Serverrolle (`DNS`), dem laufenden Hintergrunddienst und den Verwaltungswerkzeugen:

- **DNS-Serverrolle (`DNS`):** Stellt die Serverkomponenten bereit, die DNS-Anfragen auf Port 53 (UDP und TCP) entgegennehmen, Zonen im Speicher halten und Anfragen beantworten oder weiterleiten.
- **Systemdienst (`DNS`):** Der Windows-Dienst `DNS` (Prozess `dns.exe`) lädt beim Start die konfigurierten Zonen und Servereinstellungen. Läuft dieser Dienst nicht, ist keine Namensauflösung über den Host möglich – selbst wenn die Rolle installiert ist.
- **PowerShell-Modul `DnsServer`:** Über das Modul DnsServer steuerst du unter Server Core sämtliche Einstellungen des Servers, Zonen und Resource Records per Kommandozeile. Es wird bei der Rolleninstallation mit `-IncludeManagementTools` automatisch mitinstalliert.

Im laufenden Betrieb hält `dns.exe` sämtliche aktiven Zonen im Arbeitsspeicher und beantwortet Abfragen direkt aus dem RAM. Änderungen über PowerShell werden sofort im Speicher wirksam und erst zeitversetzt, beim Dienst-Stopp oder über ein explizites Zurückschreiben auf den Datenträger gespeichert. Wer eine `.dns`-Zonendatei im Editor bearbeitet, während der `DNS`-Dienst läuft, riskiert daher, dass der Dienst die manuelle Änderung beim nächsten Schreiben wieder überschreibt.

Primäre Zonen und dateibasierte Speicherung

Eine DNS-Zone bildet einen zusammenhängenden Teilbereich des DNS-Namensraums ab, für den der Server verantwortlich ist. Eine **primäre Zone** ist dabei stets die beschreibbare Originalquelle (Master) für diesen Bereich – nur hier lassen sich Resource Records direkt anlegen, ändern oder löschen.

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Da auf `WIN-CORE01` kein Active Directory existiert, speichert der Windows-DNS-Server jede primäre Zone als reguläre Textdatei im lokalen Verzeichnis `%SystemRoot%\System32\dns\` (standardmäßig unter `<Zonenname>.dns`). Im Gegensatz zu Active-Directory-integrierten Zonen gibt es hier keine automatische Multi-Master-Replikation über Domänencontroller; die Datenhaltung liegt vollständig auf diesem einen Server, wie Microsoft in der Dokumentation zu Verwalten von DNS-Zonen mithilfe des DNS-Servers in Windows Server beschreibt.

Dateibasierte primäre Zone

Speicherort
: Lokale Zonendatei (`%SystemRoot%\System32\dns\`)

Voraussetzung
: Keine Domäne nötig (Workgroup-fähig)

Schreibzugriff
: Einzelner primärer Server (Single-Master)

Sichere Updates
: Nicht verfügbar

AD-integrierte Zone

Speicherort
: Active Directory-Datenbank (AD DS)

Voraussetzung
: Domänencontroller und AD DS erforderlich

Schreibzugriff
: Jeder beschreibbare DC (Multi-Master)

Sichere Updates
: Über AD-Zugriffslisten (ACLs) steuerbar

Gemeinsam: Verwaltung der Zonen und Resource Records über das Modul `DnsServer`

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Autoritative Antworten versus rekursive Abfragen

Ein DNS-Server erfüllt im Netzwerk zwei grundlegend unterschiedliche Aufgaben, die sich im Betrieb und bei der Fehleranalyse klar unterscheiden müssen:

- **Autoritative DNS-Antworten:** Verwaltet dein Server die Zone `example.test` als primäre Zone, ist er für alle Namen innerhalb dieser Zone die verbindliche Informationsquelle. Fragt ein Client nach `web01.example.test`, antwortet der Server direkt aus seinem eigenen Zonenbestand und setzt das `AA`-Flag (*Authoritative Answer*). Existiert der Name in der Zone nicht, liefert er unmittelbar `NXDOMAIN`.
- **Rekursive DNS-Abfragen:** Erhält der Server eine Anfrage nach einem externen Namen außerhalb seiner lokalen Zonen (etwa `learn.microsoft.com`), besitzt er selbst keine autoritativen Daten dafür. Ist Rekursion aktiviert, übernimmt der DNS-Server stellvertretend für den Client die vollständige Auflösung – entweder indem er die Anfrage an konfigurierte **Forwarder** weiterreicht oder direkt über die **Root Hints** im Internet auflöst.

Forward-Lookup und Reverse-Lookup

In der praktischen Administration begegnen dir zwei entgegengesetzte Abfragerichtungen:

- **Forward-Lookup:** Der Standardfall im Alltag. Ein Hostname (FQDN wie `app01.example.test`) wird über `A`-Records in eine IPv4-Adresse oder über `AAAA`-Records in eine IPv6-Adresse aufgelöst.
- **Reverse-Lookup:** Die umgekehrte Richtung. Zu einer bekannten IP-Adresse wird der zugehörige Hostname abgefragt. Dazu nutzt DNS spezielle Rückwärtszonen (für IPv4 unterhalb von `in-addr.arpa`), in denen `PTR`-Records gepflegt werden. Reverse-Lookups werden häufig für Protokollauswertungen, Diagnosewerkzeuge wie `nslookup` oder Dienstprüfungen benötigt.

Voraussetzungen und Netzwerkplanung

Mit diesem Architekturverständnis aus Rolle, Zonen und Abfragerichtung lassen sich die Netzwerkgrundlagen für `WIN-CORE01` festlegen, bevor das erste Paket den DNS-Dienst erreicht.

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Ein DNS-Server ist ein Infrastrukturdienst: Ändert sich seine eigene Adresse oder blockiert eine Firewall den Antwortpfad, schlägt die Namensauflösung im gesamten angeschlossenen Netzsegment fehl.

Systembasis und administrativer Zugang

Dieser Leitfaden setzt die lauffähige Basisinstallation aus dem vorherigen Server-Core-Artikel voraus:

- **Betriebssystem:** Windows Server 2025 Standard Evaluation (Server Core, x64), Hostname `WIN-CORE01`, Workgroup-Mitgliedschaft ohne Active Directory.
- **Verwaltungszugang:** Bestehende OpenSSH-Verbindung (Port 22/TCP) mit administrativen Rechten in Windows PowerShell 5.1.
- **Netzwerkanbindung:** Direkte Erreichbarkeit innerhalb des Labornetzes zwischen Administrations-Client, `WIN-CORE01` und dem vorgelagerten Router beziehungsweise Upstream-Resolver.

Statische IPv4-Adresse und DNS-Clientkonfiguration des Servers

Ein DNS-Server benötigt zwingend eine **statische IPv4-Adresse**. Bezieht der Host seine Adresse per DHCP ohne feste Reservierung und wechselt nach einem Neustart die IP, laufen alle im Netz verteilten DNS-Clientkonfigurationen ins Leere. Prüfe vor Beginn die aktuelle Schnittstellenkonfiguration des Servers:

```
# Aktuelle IPv4-Adressen und Schnittstellenindizes anzeigen
Get-NetIPAddress -AddressFamily IPv4 |
    Where-Object { $_.InterfaceAlias -notlike "*Loopback*" } |
    Format-Table InterfaceIndex, InterfaceAlias, IPAddress, PrefixLength, PrefixOrigin

# Aktuelle DNS-Clientkonfiguration des Servers prüfen
Get-DnsClientServerAddress -AddressFamily IPv4

# Erreichbarkeit des künftigen Upstream-Forwarders vorab auf Port 53 testen
```

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
Test-NetConnection -ComputerName 192.0.2.1 -Port 53
```

Schematische Beispielausgabe (auf dem Zielsystem nicht live ausgeführt):

```
InterfaceIndex InterfaceAlias IPAddress  PrefixLength PrefixOrigin
-------------- -------------- ---------  ------------ ------------
             6 Ethernet       192.0.2.10           24 Manual

InterfaceAlias               InterfaceIndex AddressFamily ServerAddresses
--------------               -------------- ------------- ---------------
Ethernet                                  6 IPv4          {192.0.2.1}
```

Achte bei `PrefixOrigin` auf `Manual` (statisch vergeben, kein `Dhcp`) und notiere dir den exakten `InterfaceAlias` (etwa `Ethernet` oder `vEthernet`), da du ihn für spätere Änderungen mit `Set-DnsClientServerAddress` benötigst. Besteht zusätzlich schon vor der Rolleninstallation eine saubere TCP-Verbindung zu `192.0.2.1:53` (`TcpTestSucceeded : True`), weißt du sicher, dass das Labornetz ausgehende DNS-Abfragen zulässt.

Prüfe in Workgroup-Umgebungen außerdem direkt das aktive Windows-Netzwerkprofil: Nach einer Neuinstallation steht die Schnittstelle oft auf `Public`, wodurch spätere Standard-Firewallregeln für `Private` oder `Domain` nicht greifen. Stelle es bei Bedarf auf `Private` um:

```
# Netzwerkprofil prüfen und im internen Labornetz auf Private setzen
Get-NetConnectionProfile | Select-Object InterfaceAlias, NetworkCategory
Set-NetConnectionProfile -InterfaceAlias "Ethernet" -NetworkCategory Private

# Nur falls die IP noch per DHCP bezogen wird: Statische IPv4-Adresse 192.0.2.10/24
vergeben
# New-NetIPAddress -InterfaceAlias "Ethernet" -IPAddress "192.0.2.10" -PrefixLength 24 -
```

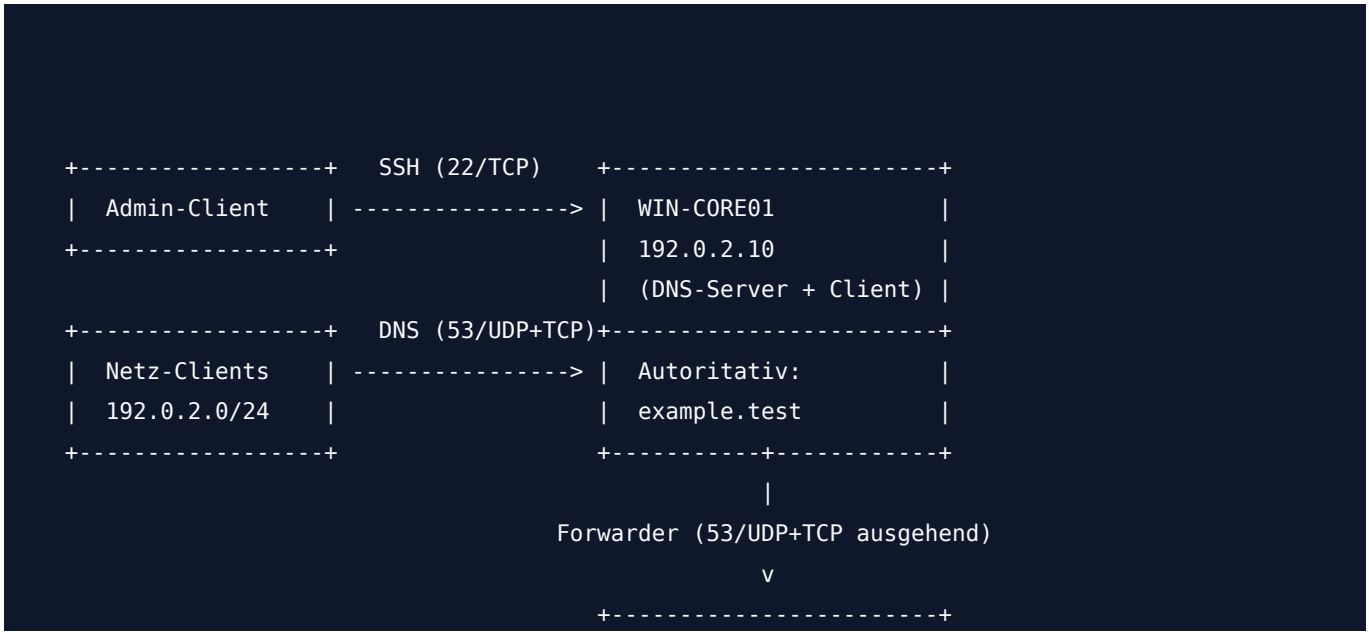
```
DefaultGateway "192.0.2.1"
```

Soll `WIN-CORE01` für lokale Abfragen auch selbst seinen eigenen DNS-Dienst nutzen, passt du die DNS-Clientadresse der Netzwerkschnittstelle später über Set-DnsClientServerAddress an.

> ⚠ **Rückkopplung auf die eigene Namensauflösung:** Ändere die DNS-Clientkonfiguration von `WIN-CORE01` nicht voreilig auf `127.0.0.1` oder die eigene Server-IP, solange der DNS-Dienst noch nicht läuft und keine Forwarder konfiguriert sind. Solange kein funktionsfähiger Resolver lokal antwortet, verliert der Server selbst sofort jede externe Namensauflösung – Windows Updates, PowerShell-Modulabfragen oder Zeitabgleiche schlagen dann fehl.

Netzwerktopologie und Port-Matrix für den DNS-Betrieb

Im Gegensatz zur allgemeinen Ersteinrichtung (mit Proxmox-GUI, SMB oder WinRM) reduziert sich die Angriffsfläche im reinen DNS-Betrieb auf wenige klar abgegrenzte Verbindungen. Clients senden ihre Abfragen an `WIN-CORE01`, während der Server selbst nicht auflösbare Namen ausgehend an einen definierten Upstream-Resolver weiterleitet:



Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
                                    |  Upstream-Resolver     |
                                    |  (Router / Labor-DNS)  |
                                    +------------------------+
```

Für den Datenverkehr müssen auf der Windows Defender Firewall sowie auf vorgeschalteten Netz-Firewalls folgende Ports berücksichtigt werden:

Port	Protokoll	Richtung	Zweck im DNS-Betrieb
22	TCP	Eingehend (Admin-Netz)	OpenSSH-Verwaltung in der PowerShell
53	UDP	Eingehend (Clients)	Standard-DNS-Abfragen und kurze Antworten
53	TCP	Eingehend (Clients)	Fallback bei großen Antworten (> 512 Byte ohne EDNS0) und Zonentransfers
53	UDP/TCP	Ausgehend (zum Forwarder)	Rekursive Weiterleitung externer DNS-Anfragen
49152–65535	UDP/TCP	Ausgehend / Rückkanal	Ephemeral Ports für Antworten auf ausgehende DNS-Abfragen

In der Praxis wird oft vergessen, neben **53/UDP** auch **53/TCP** freizugeben. Überschreitet eine Antwort die zulässige UDP-Paketgröße, setzt der Server das *Truncated*-Bit (`TC`), woraufhin der Client automatisch auf TCP-Port 53 wechselt. Bleibt TCP blockiert, scheitern genau solche Abfragen scheinbar zufällig mit Timeouts.

Testzone, Adressbereiche und Trennung der Auflösungswege

Damit die Beispiele dieses Guides reproduzierbar bleiben und mit keinen realen Domains oder Netzen kollidieren, nutzen wir durchgehend reservierte Dokumentationswerte:

- **Testzone (RFC 6761):** `example.test` (Endung `.test` ist für interne Tests reserviert und wird im öffentlichen DNS nicht aufgelöst).
- **IPv4-Dokumentationsnetz (RFC 5737):** `192.0.2.0/24` (mit `192.0.2.10` als Beispieladresse für `WIN-CORE01`).

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

- **IPv6-Dokumentationspräfix (RFC 3849):** `2001:db8::/32`.

> **Dokumentationsadressen im eigenen Labor anpassen:** Die Netze `192.0.2.0/24` und `2001:db8::/32` sind offizielle Dokumentationsbereiche und werden im Internet nicht geroutet. Ersetze diese Platzhalter bei der praktischen Umsetzung in deiner Umgebung durch die tatsächlichen privaten IP-Adressen deines lokalen Labor- oder Testsegments.

Plane bereits vor dem Anlegen der ersten Zone die klare Trennung beider Auflösungswege: Alle Hostnamen innerhalb von `example.test` sowie die zugehörige Reverse-Zone beantwortet `WIN-CORE01` ausschließlich **autoritativ** aus seinen lokalen Zonendateien. Alle übrigen Anfragen (wie öffentliche Domains) gehen über explizit definierte **Forwarder** an einen vorgelagerten Resolver im Netz. So vermeidest du, dass lokale Testnamen ins Internet geleakt werden oder der Server unkontrolliert direkte Root-Abfragen startet.

DNS-Serverrolle installieren

Stehen IP-Adresse, Zielnetz und Port-Anforderungen fest, folgt auf `WIN-CORE01` die Installation der DNS-Serverrolle per SSH in einer administrativen PowerShell-Sitzung. Wie Microsoft im Leitfaden Installieren und Konfigurieren von DNS-Server unter Windows Server beschreibt, trennt auch die Installation sauber zwischen den Server-Binärdateien und den Verwaltungswerkzeugen.

Ausgangszustand prüfen und Rolle installieren

Prüfe zuerst, ob die Rolle bereits vorhanden ist, damit keine unnötige Änderung am System vorgenommen wird:

```
# Ausgangszustand: Aktuellen Installationsstatus der DNS-Serverrolle und -Tools abfragen
Get-WindowsFeature -Name DNS, RSAT-DNS-Server
```

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Im Ausgangszustand zeigen beide Einträge den Status `Available`. Um die DNS-Serverrolle zusammen mit dem PowerShell-Modul `DnsServer` zu installieren, nutzt du Install-WindowsFeature mit dem Parameter `-IncludeManagementTools`:

```
# DNS-Serverrolle samt Verwaltungswerkzeugen und DnsServer-Modul installieren
Install-WindowsFeature -Name DNS -IncludeManagementTools
```

- **`-Name DNS`:** Installiert die eigentliche DNS-Serverrolle inklusive des Systemdienstes `DNS` (`dns.exe`).
- **`-IncludeManagementTools`:** Installiert zusätzlich die zugehörigen Verwaltungswerkzeuge (`RSAT-DNS-Server`), zu denen auch das PowerShell-Modul `DnsServer` gehört. Ohne diesen Schalter fehlt dir auf Server Core das PowerShell-Modul zur weiteren Konfiguration.

Die Installation erfordert unter Windows Server 2025 keinen Neustart (`Restart Needed: No`). Nach Abschluss der Rolleninstallation kontrollierst du zuerst den Feature-Status selbst:

```
# Kontrollieren, ob Rolle und Management-Tools auf "Installed" stehen
Get-WindowsFeature -Name DNS, RSAT-DNS-Server | Format-Table Name, InstallState
```

Schematische Beispielausgabe (auf dem Zielsystem nicht live ausgeführt):

```
Name            InstallState
----            ------------
DNS                Installed
```

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
RSAT-DNS-Server    Installed
```

Dienststatus, Autostart und PowerShell-Modul kontrollieren

Ein installiertes Windows-Feature bedeutet noch nicht automatisch, dass der Dienst dauerhaft sauber startet und die benötigten PowerShell-Cmdlets bereitstehen. Prüfe daher Dienstzustand, Starttyp und Modulverfügbarkeit in zwei gezielten Schritten:

```
# Dienststatus und Starttyp des DNS-Servers prüfen
Get-Service -Name DNS | Select-Object Name, Status, StartType
```

Schematische Beispielausgabe (auf dem Zielsystem nicht live ausgeführt):

```
Name  Status StartType
----  ------ ---------
DNS  Running Automatic
```

Der Dienst `DNS` muss hier `Running` und `Automatic` zeigen. Sollte er nach einer Sonderkonfiguration gestoppt sein oder auf `Manual` stehen, korrigierst du das explizit:

```
# Nur ausführen, falls Status nicht Running oder StartType nicht Automatic ist
Set-Service -Name DNS -StartupType Automatic
Start-Service -Name DNS
```

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Kontrolliere anschließend, ob das Modul `DnsServer` in der PowerShell-Sitzung geladen werden kann und die erwarteten Cmdlets bereitstellt:

```
# Prüfen, ob das Modul DnsServer verfügbar ist
Get-Module -ListAvailable -Name DnsServer

# Anzahl der verfügbaren Cmdlets im Modul prüfen (optional zur Kontrolle)
(Get-Command -Module DnsServer).Count
```

Listener auf Port 53 und Firewall-Regeln prüfen

Läuft der `DNS`-Dienst, bindet er sich an die lokalen IP-Schnittstellen auf Port 53. Prüfe direkt auf Socket-Ebene, ob sowohl UDP als auch TCP auf Port 53 lauschen:

```
# Offene UDP- und TCP-Listener auf Port 53 prüfen
Get-NetUDPEndpoint -LocalPort 53 | Format-Table LocalAddress, LocalPort, OwningProcess
Get-NetTCPConnection -LocalPort 53 -State Listen | Format-Table LocalAddress, LocalPort,
State, OwningProcess
```

Schematische Beispielausgabe (auf dem Zielsystem nicht live ausgeführt):

```
LocalAddress LocalPort OwningProcess
------------ --------- -------------
::                  53          2480
0.0.0.0             53          2480

LocalAddress LocalPort  State OwningProcess
```

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
------------ ---------  ----- -------------
::                  53 Listen          2480
0.0.0.0             53 Listen          2480
```

- **`Get-NetUDPEndpoint -LocalPort 53`:** Zeigt die gebundenen IP-Adressen (etwa `0.0.0.0`, `::` oder spezifische Schnittstellen-IPs) für Standard-DNS-Anfragen.
- **`Get-NetTCPConnection -LocalPort 53 -State Listen`:** Bestätigt, dass der Server auch für TCP-Fallbacks und Zonentransfers empfangsbereit ist. Die identische `OwningProcess`-ID verweist auf den Prozess `dns.exe`.

Prüfe direkt im Anschluss die aktiven Firewall-Regeln für Port 53. Über `Get-NetFirewallPortFilter` findest du die Regeln unabhängig davon, ob das Betriebssystem auf Deutsch oder Englisch installiert wurde:

```
# Firewall-Regeln für Port 53 (UDP/TCP) sprachunabhängig auslesen
Get-NetFirewallPortFilter |
    Where-Object { $_.LocalPort -eq 53 } |
    Get-NetFirewallRule |
    Select-Object DisplayName, Direction, Action, Enabled, Profile
```

Schematische Beispielausgabe (auf dem Zielsystem nicht live ausgeführt):

```
DisplayName              Direction Action Enabled Profile
-----------              --------- ------ ------- -------
DNS (UDP-In)               Inbound  Allow    True Domain, Private
DNS (TCP-In)               Inbound  Allow    True Domain, Private
```

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Achte bei der Ausgabe besonders auf `Enabled = True`, `Direction = Inbound` und `Action = Allow` sowie auf das aktive Netzwerkprofil (`Domain`, `Private` oder `Public`). Steht die Schnittstelle auf `WIN-CORE01` in einer Workgroup-Umgebung etwa auf `Public` und die vordefinierte Regel erlaubt nur `Domain` / `Private`, verwirft die Windows Defender Firewall eingehende Client-Anfragen trotz laufendem Dienst stillschweigend.

Fehlen in einem gehärteten Core-Image passende Port-53-Regeln oder möchtest du den Zugriff direkt auf das vorgesehene Client-Subnetz `192.0.2.0/24` einschränken, legst du für UDP und TCP explizite Regeln an:

```
# Nur falls noch keine passende Regel existiert: DNS auf Client-Subnetz 192.0.2.0/24
freigeben
New-NetFirewallRule -DisplayName "DNS-Server (UDP-In, Lab-Subnetz)" -Direction Inbound -
Protocol UDP -LocalPort 53 -RemoteAddress "192.0.2.0/24" -Profile Private -Action Allow
New-NetFirewallRule -DisplayName "DNS-Server (TCP-In, Lab-Subnetz)" -Direction Inbound -
Protocol TCP -LocalPort 53 -RemoteAddress "192.0.2.0/24" -Profile Private -Action Allow
```

> ⚠ **Installierte Rolle ist noch kein Funktionsnachweis:** Dass `Install-WindowsFeature`, `Get-Service` und die Port-Prüfung auf 53 ohne Fehler durchlaufen, bestätigt lediglich die Betriebsbereitschaft des Basisdienstes. Ob `WIN-CORE01` autoritative Antworten liefert oder externe Namen korrekt weiterleitet, ist damit noch nicht bewiesen – dafür fehlen bislang sowohl eigene DNS-Zonen als auch konfigurierte Forwarder.

Rolle sauber zurückbauen (optionales Rollback im Labor)

Falls du einen Laboraufbau auf `WIN-CORE01` vollständig zurücksetzen möchtest, entfernst du die DNS-Serverrolle gezielt über `Uninstall-WindowsFeature`. Beachte dabei, dass Windows aus Sicherheitsgründen die physischen Zonendateien unter `%SystemRoot%\System32\dns\` bei einer Deinstallation nicht automatisch löscht – sie bleiben erhalten und lassen sich bei einer späteren Neuinstallation entweder weiterverwenden oder vorab gezielt archivieren:

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
# DNS-Serverrolle (inkl. Verwaltungswerkzeuge) sauber deinstallieren
Uninstall-WindowsFeature -Name DNS -IncludeManagementTools -Restart
```

Forward-Lookupzone erstellen

Der DNS-Dienst läuft auf Port 53 und das Modul `DnsServer` steht bereit – bislang kennt der Server jedoch noch keinen eigenen Namensraum. Im nächsten Schritt richtest du die erste autoritative Forward-Lookupzone (`example.test`) ein, damit `WIN-CORE01` Anfragen für eigene Hostnamen lokal und verbindlich beantworten kann.

Auswahl des Zonennamens und Abgrenzung der Zonentypen

Für interne Test- und Laborzonen gilt eine klare Regel: Verwende niemals produktive öffentliche Domains, die dir nicht gehören, und vermeide veraltete Pseudo-Endungen wie `.local` (reserviert für Multicast DNS / mDNS nach RFC 6762). Die Top-Level-Domain `.test` ist nach RFC 6761 dauerhaft für Testzwecke reserviert und kollidiert weder mit dem öffentlichen DNS noch mit mDNS-Diensten im lokalen Netz.

Bei der Erstellung einer neuen Zone unterscheidet Windows Server DNS grundsätzlich zwischen drei Zonentypen:

- **Primäre Zone (`Primary`):** Hält die einzige beschreibbare Master-Kopie der Zonendaten. Nur in einer primären Zone legst du neue Resource Records an oder änderst bestehende Einträge.
- **Sekundäre Zone (`Secondary`):** Hält eine schreibgeschützte Kopie, die ihre Daten regelmäßig per Zonentransfer (AXFR/IXFR) von einem primären DNS-Server bezieht. Sie dient der Ausfallsicherheit oder Lastverteilung und erlaubt keine direkten Änderungen.
- **Stubzone (`Stub`):** Speichert ausschließlich die für die Namensserver-Ermittlung nötigen Einträge (`SOA`, `NS` und zugehörige Glue-Records), um Delegierungen zu finden.

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Merkmal	Primäre Zone	Sekundäre Zone	Stubzone
Schreibzugriff	Ja (Master)	Nein (nur lesend)	Nein (nur Verweise)
Gespeicherte Records	Alle Zonen-Records	Alle Zonen-Records	Nur `SOA`, `NS`, Glue-`A`/`AAAA`
Datenquelle	Lokale Datei (oder AD)	Zonentransfer (`AXFR`/`IXFR`)	Zonentransfer der Nameserver-Liste
Typischer Einsatz	Hauptserver wie `WIN-CORE01`	Zweiter DNS zur Ausfallsicherheit	Schnelles Finden delegierter Subzonen

Für unseren eigenständigen Workgroup-Server erstellen wir eine **dateibasierte primäre Forward-Lookupzone**.

Ausgangszustand der Zonen prüfen

Kontrolliere vor dem Anlegen der Zone mit Get-DnsServerZone, welche Zonen auf dem frisch installierten DNS-Server bereits existieren:

```
# Ausgangszustand: Vorhandene DNS-Zonen auf dem Server anzeigen
Get-DnsServerZone
```

Auf einem neu aufgesetzten Server ohne Active Directory erscheinen hier zunächst nur systeminterne Einträge (wie `TrustAnchors` für DNSSEC-Vertrauensanker), jedoch keine eigenen Forward-Lookupzonen.

Primäre dateibasierte Zone mit PowerShell anlegen

Die Einrichtung erfolgt über Add-DnsServerPrimaryZone. Da `WIN-CORE01` kein Domänencontroller ist, darfst du keine Active-Directory-Parameter wie `-ReplicationScope` verwenden – stattdessen ist der Parameter `-ZoneFile` zwingend erforderlich, um die Speicherung als lokale Zonendatei festzulegen:

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
# Primäre dateibasierte Forward-Lookupzone ohne dynamische Updates anlegen
Add-DnsServerPrimaryZone -Name "example.test" -ZoneFile "example.test.dns" -DynamicUpdate
"None" -PassThru
```

Die Parameter steuern das Verhalten eindeutig:

- **`-Name "example.test"`:** Definiert den DNS-Zonennamen. Alle darunter liegenden Hostnamen (etwa `app01.example.test`) werden Teil dieser Zone.
- **`-ZoneFile "example.test.dns"`:** Legt den Dateinamen der Zonendatenbank fest. Ohne vollständige Pfadangabe erstellt Windows Server die Datei automatisch im Standardverzeichnis `%SystemRoot%\System32\dns\example.test.dns`.
- **`-DynamicUpdate "None"`:** Deaktiviert dynamische Client-Registrierungen vollständig.
- **`-PassThru`:** Gibt das erstellte Zonenobjekt nach dem Anlegen direkt in der PowerShell-Sitzung zur ersten Kontrolle zurück.

Das Cmdlet erzeugt im Hintergrund sofort die Zonendatei auf der Festplatte, generiert automatisch die initialen `SOA`- und `NS`-Records für `WIN-CORE01` und lädt die Zone aktiv in den Arbeitsspeicher des `DNS`-Dienstes.

Dynamische Aktualisierungen richtig einordnen

Der Parameter `-DynamicUpdate` verdient bei eigenständigen Workgroup-Servern besondere Aufmerksamkeit. Microsoft kennt hier drei Modi, von denen in einer Workgroup-Umgebung nur zwei technisch unterstützt werden:

- **`None` (Empfohlen für statische Server-Zonen):** Kein Client darf selbstständig Einträge hinzufügen oder überschreiben. Alle Änderungen erfolgen explizit und nachvollziehbar durch dich als Administrator per PowerShell.
- **`NonsecureAndSecure`:** Erlaubt unsichere dynamische Updates ohne Authentifizierung. Jeder Host im Netz, der Port 53 erreicht, könnte theoretisch beliebige DNS-Namen in

`example.test` registrieren oder bestehende Einträge manipulieren.

- **`Secure`:** Erfordert Kerberos-Authentifizierung über Active Directory und steht daher **ausschließlich** bei AD-integrierten Zonen zur Verfügung. Ein Aufruf mit `-DynamicUpdate "Secure"` schlägt bei dateibasierten Zonen mit einer Fehlermeldung fehl.

Zoneneigenschaften und Zonendatei kontrollieren

Prüfe unmittelbar nach dem Anlegen sowohl die logische Zonenkonfiguration im Dienst als auch die physische Datei auf dem Datenträger:

```
# Detaillierte Eigenschaften der neuen Zone auslesen
Get-DnsServerZone -Name "example.test" | Format-List ZoneName, ZoneType, IsAutoCreated,
IsDsIntegrated, DynamicUpdate, ZoneFile
```

Die schematische Beispielausgabe zeigt, woran du eine sauber angelegte dateibasierte Zone erkennst:

```
ZoneName       : example.test
ZoneType       : Primary
IsAutoCreated  : False
IsDsIntegrated : False
DynamicUpdate  : None
ZoneFile       : example.test.dns
```

Entscheidend sind die Werte `ZoneType : Primary`, `IsDsIntegrated : False` und `DynamicUpdate : None`. Sie bestätigen, dass die Zone unabhängig von Active Directory als lokale Datei geführt wird.

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Betrachte nun die erzeugte Zonendatei direkt im Dateisystem. Da der `DNS`-Dienst Änderungen im Betrieb zunächst im Arbeitsspeicher hält und erst periodisch, beim sauberen Beenden oder nach einem expliziten `dnscmd /ZoneWriteBack example.test` auf die Platte schreibt, lohnt vor einem direkten Dateivergleich nach frischen Änderungen ein kurzer Writeback:

```
# Optional: Zonenpuffer sofort in die Datei example.test.dns schreiben und Inhalt prüfen
dnscmd.exe /ZoneWriteBack example.test
Get-Content -Path "$env:SystemRoot\System32\dns\example.test.dns"
```

Eine neu erstellte Zonendatei folgt dem standardisierten Textformat (schematisches Beispiel zur Orientierung):

```
;
;  Database file example.test.dns for example.test zone.
;      Zone version:  1
;

@                       IN  SOA win-core01.  hostmaster.  (
                                1            ; serial number
                                900          ; refresh
                                600          ; retry
                                86400        ; expire
                                3600       ) ; default TTL

;
;  Zone NS records
;

@                       NS□win-core01.
win-core01.             A□192.0.2.10
```

Beachte zwei Details in diesem Aufbau:

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

1. **Abschließender Punkt bei FQDNs:** Namen mit abschließendem Punkt (`win-core01.`) stehen absolut; Namen ohne Punkt werden vom Parser automatisch um den Zonennamen ergänzt.
2. **Glue-Eintrag für den Nameserver:** Windows trägt den Hostnamen des Servers (`win-core01`) beim Anlegen oft automatisch als `NS` ein. Sobald ein vollqualifizierter Zonenbezug (etwa `win-core01.example.test.`) gewünscht ist, solltest du den `NS`- und `SOA`-Record sauber darauf abstimmen, damit Clients den zuständigen Nameserver eindeutig innerhalb der Zone auflösen können.

SOA- und NS-Records verstehen und prüfen

Beim Erstellen jeder primären Zone legt Windows automatisch genau einen `SOA`-Record (*Start of Authority*) und mindestens einen `NS`-Record (*Name Server*) am Zonen-Apex (`@`) an. Du prüfst alle vorhandenen Einträge über `Get-DnsServerResourceRecord`:

```
# Alle initial angelegten Resource Records der Zone anzeigen
Get-DnsServerResourceRecord -ZoneName "example.test"
```

Schematische Beispielausgabe (auf dem Zielsystem nicht live ausgeführt):

```
HostName  RecordType Type       Timestamp           TimeToLive      RecordData
--------  ---------- ----       ---------           ----------      ----------
@         NS         2          0                   01:00:00        win-core01.
@         SOA        6          0                   01:00:00        [1][win-
core01.][hostmaster.]
```

Für die Praxis lohnt außerdem ein Blick auf die detaillierten `SOA`-Felder über `Format-List`, weil hier insbesondere die `MinimumTimeToLive` über das Negativ-Caching entscheidet:

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
# SOA-Eigenschaften im Detail ausgeben
(Get-DnsServerResourceRecord -ZoneName "example.test" -RRType SOA).RecordData | Format-
List
```

Schematische Beispielausgabe (auf dem Zielsystem nicht live ausgeführt):

```
SerialNumber      : 1
PrimaryServer     : win-core01.
ResponsiblePerson : hostmaster.
RefreshInterval   : 00:15:00
RetryDelay        : 00:10:00
ExpireLimit       : 1.00:00:00
MinimumTimeToLive : 01:00:00
```

Der **`SOA`-Record** enthält die zentralen Metadaten der Zone:

- **`PrimaryServer`:** Der Hostname des maßgeblichen primären DNS-Servers für diese Zone.
- **`ResponsiblePerson`:** Die Kontaktadresse des Zonenverantwortlichen in DNS-Notation (ohne `@`-Zeichen, das durch einen Punkt ersetzt wird, z. B. `hostmaster.example.test.`).
- **`SerialNumber`:** Eine fortlaufende Versionsnummer der Zone. Bei jeder Änderung an der Zone erhöht der Windows-DNS-Dienst diese Nummer automatisch, damit sekundäre Server Änderungen erkennen.
- **`RefreshInterval`, `RetryDelay`, `ExpireLimit`:** Zeitintervalle (standardmäßig 15 Minuten, 10 Minuten, 1 Tag), die steuern, wie oft sekundäre Server auf Änderungen prüfen und wann veraltete Kopien verworfen werden.
- **`MinimumTimeToLive`:** Standardwert (Negativ-Caching-TTL, häufig 1 Stunde / `01:00:00`), der unter anderem angibt, wie lange andere Resolver eine `NXDOMAIN`-Antwort („Name existiert nicht“) zwischenspeichern dürfen.

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Falls du `PrimaryServer` oder `ResponsiblePerson` auf einen sauberen FQDN innerhalb deiner Zone anpassen möchtest, liest du den bestehenden SOA-Record aus, änderst die Kopie im Speicher und schreibst ihn über `Set-DnsServerResourceRecord` zurück:

```
# Bestehenden SOA-Record auslesen und Kopie für die Bearbeitung erzeugen
$oldSoa = Get-DnsServerResourceRecord -ZoneName "example.test" -RRType SOA
$newSoa = $oldSoa.Clone()

# Maßgeblichen Server und Kontaktadresse auf FQDN innerhalb von example.test setzen
$newSoa.RecordData.PrimaryServer = "win-core01.example.test."
$newSoa.RecordData.ResponsiblePerson = "hostmaster.example.test."

# Geänderten SOA-Record in die Zone schreiben
Set-DnsServerResourceRecord -NewInputObject $newSoa -OldInputObject $oldSoa -ZoneName
"example.test"
```

Autoritative Antwort der Zone verifizieren

Dass `Get-DnsServerZone` die Zone auflistet, ist die interne Sicht des Servers. Prüfe abschließend über eine echte DNS-Abfrage gegen `127.0.0.1`, ob der Dienst für `example.test` tatsächlich autoritativ über Port 53 antwortet:

```
# SOA- und NS-Einträge der neuen Zone direkt über den lokalen DNS-Socket abfragen
Resolve-DnsName -Name "example.test" -Type SOA -Server 127.0.0.1 -DnsOnly -NoHostsFile
Resolve-DnsName -Name "example.test" -Type NS -Server 127.0.0.1 -DnsOnly -NoHostsFile
```

Schematische Beispielausgabe für `Type SOA` (auf dem Zielsystem nicht live ausgeführt):

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
Name         Type   TTL   Section    PrimaryServer             NameAdministrator
SerialNumber
----         ----   ---   -------    -------------             -----------------
------------
example.test SOA    3600  Answer     win-core01.example.test    hostmaster.example.test  2
```

In der Ausgabe von `Resolve-DnsName` erkennst du die erfolgreiche autoritative Antwort daran, dass der Eintrag direkt im Abschnitt `Answer` (`Section : Answer`) mit `Type : SOA` beziehungsweise `Type : NS` und den gesetzten Daten (`win-core01.example.test`) zurückkommt, ohne dass eine Weiterleitung an externe Resolver stattfindet. Beachte auch, dass `SerialNumber` durch die vorherige `SOA`-Anpassung sauber auf `2` hochgezählt wurde.

> ⚠ **Typische Fehlerquellen beim Anlegen der Zone:** Schlägt `Add-DnsServerPrimaryZone` fehl, liegt dies in Workgroup-Umgebungen meist an einem fehlenden `-ZoneFile`-Parameter oder an dem Versuch, AD-Parameter wie `-ReplicationScope` oder `-DynamicUpdate "Secure"` zu übergeben. Erscheint die Zone zwar in `Get-DnsServerZone`, liefert `Resolve-DnsName` aber dennoch einen Timeout, blockiert die Firewall Port 53 oder der `DNS`-Dienst lauscht nicht auf der abgefragten IP-Adresse.

Ausblick: Sekundäre DNS-Zone zur Ausfallsicherheit

In einer reinen Workgroup-Umgebung ohne Active-Directory-Replikation existiert zunächst genau ein beschreibbarer Master (`WIN-CORE01`). Fällt dieser Server während eines Wartungsfensters aus, steht für interne Namen kein Ersatz-Resolver bereit. Sobald du später einen zweiten Server-Core-Host (etwa `WIN-CORE02` unter `192.0.2.11`) ergänzt, richtest du dort statt einer weiteren primären Zone eine schreibgeschützte **sekundäre Zone** ein, die ihren Bestand automatisch per Zonentransfer (`AXFR`/`IXFR` über Port 53/TCP) von `WIN-CORE01` zieht:

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
# Auf dem zweiten Server WIN-CORE02 (192.0.2.11): Sekundäre Zone anlegen
Add-DnsServerSecondaryZone -Name "example.test" -ZoneFile "example.test.dns" -
MasterServers "192.0.2.10" -PassThru

# Auf dem Master WIN-CORE01 (192.0.2.10): Transfer gezielt nur für 192.0.2.11 freigeben
Set-DnsServerPrimaryZone -Name "example.test" -SecureSecondaries "TransferToSecureServers"
-SecondaryServers "192.0.2.11" -Notify "NotifyServers" -NotifyServers "192.0.2.11"
```

So bleibt `WIN-CORE01` die alleinige Quelle der Wahrheit, während `WIN-CORE02` bei Änderungen an der `SOA`-Seriennummer automatisch informiert wird und alle Clients weiterhin interne Namen auflösen können.

DNS Resource Records verwalten

Mit der aktiven Forward-Lookupzone `example.test` kennt der DNS-Server nun seinen Zuständigkeitsbereich – allerdings enthält die Zone bislang nur `SOA`- und `NS`-Verwaltungsdaten. Damit Clients echte Systeme erreichen können, legst du jetzt die eigentlichen Resource Records an.

Server-Änderung versus Abfrage sauber trennen

Im Alltag mit dem Modul `DnsServer` vermischen Administratoren oft zwei völlig unterschiedliche Ebenen:

- **Änderungen an der Zonendatenbank (`Add-`, `Set-`, `Remove-DnsServerResourceRecord*`):** Diese Cmdlets schreiben direkt in die Zone des DNS-Servers, erhöhen die `SOA`-Seriennummer und verändern das Antwortverhalten für alle Clients im Netz.
- **Reine Abfragen (`Get-DnsServerResourceRecord` und `Resolve-DnsName`):** `Get-DnsServerResourceRecord` liest lediglich den gespeicherten Zonenbestand auf dem Server aus, während `Resolve-DnsName` eine echte DNS-Abfrage wie ein Client über den Netzwerk-Socket abschickt. Beide Befehle verändern keine Zonendaten.

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

A-Records (IPv4-Hosteinträge)

Ein `A`**-Record** (*Address Record*) ordnet einem Hostnamen innerhalb der Zone genau eine IPv4-Adresse zu. Erforderlich sind der Hostname (`-Name`), die Zielzone (`-ZoneName`) und eine gültige IPv4-Adresse (`-IPv4Address`). Optional setzt `-TimeToLive` (`TTL` , im Format `hh:mm:ss`), wie lange Clients die Antwort im Cache behalten dürfen.

Legst du beispielsweise den Webserver `web01.example.test` unter `192.0.2.20` mit einer TTL von einer Stunde an, nutzt du Add-DnsServerResourceRecordA:

```
# A-Record für web01.example.test auf 192.0.2.20 anlegen
Add-DnsServerResourceRecordA -Name "web01" -ZoneName "example.test" -IPv4Address
"192.0.2.20" -TimeToLive 01:00:00
```

Der Parameter `-CreatePtr` legt bei Bedarf zeitgleich den passenden Reverse-Eintrag an – allerdings **nur dann**, wenn die passende Reverse-Lookupzone auf dem Server bereits existiert (dazu kommen wir direkt im nächsten Abschnitt).

Prüfe den neuen Eintrag anschließend sowohl im Zonenbestand als auch über eine echte DNS-Abfrage:

```
# Bestand in der Zone prüfen
Get-DnsServerResourceRecord -ZoneName "example.test" -Name "web01" -RRType "A"

# Echte DNS-Abfrage gegen den lokalen Dienst testen
Resolve-DnsName -Name "web01.example.test" -Type A -Server 127.0.0.1 -DnsOnly -NoHostsFile
```

Schematische Beispielausgabe von `Resolve-DnsName` (auf dem Zielsystem nicht live ausgeführt):

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
Name                      Type   TTL   Section    IPAddress
----                      ----   ---   -------    ---------
web01.example.test        A      3600  Answer     192.0.2.20
```

AAAA-Records (IPv6-Hosteinträge)

Ein **`AAAA`-Record** (*Quad-A Record*) erfüllt exakt dieselbe Aufgabe für IPv6-Adressen (128 Bit). Erforderlich sind Hostname, Zone und eine gültige IPv6-Adresse (`-IPv6Address`). Beachte, dass derselbe Hostname problemlos parallel einen `A` - und einen `AAAA` -Record besitzen darf (Dual-Stack-Betrieb).

Für den IPv6-Eintrag von `web01.example.test` nutzt du `Add-DnsServerResourceRecordAAAA` :

```
# AAAA-Record für web01.example.test mit Dokumentationspräfix 2001:db8::/32 anlegen
Add-DnsServerResourceRecordAAAA -Name "web01" -ZoneName "example.test" -IPv6Address
"2001:db8::20" -TimeToLive 01:00:00
```

Die Überprüfung erfolgt analog getrennt nach Zonenbestand und echter Abfrage:

```
# AAAA-Eintrag in der Zone und per DNS-Abfrage verifizieren
Get-DnsServerResourceRecord -ZoneName "example.test" -Name "web01" -RRType "AAAA"
Resolve-DnsName -Name "web01.example.test" -Type AAAA -Server 127.0.0.1 -DnsOnly -
NoHostsFile
```

Schematische Beispielausgabe von `Resolve-DnsName` (auf dem Zielsystem nicht live ausgeführt):

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
Name                      Type   TTL   Section    IPAddress
----                      ----   ---   -------    ---------
web01.example.test        AAAA   3600  Answer     2001:db8::20
```

CNAME-Records (Aliase)

Ein **`CNAME`-Record** (*Canonical Name*) verweist von einem Alias-Namen auf einen anderen vollqualifizierten Hostnamen (`-HostNameAlias`), den kanonischen Namen. Erforderlich sind der Alias-Name in der eigenen Zone, der Zonenname und das Ziel als FQDN.

`CNAME`-Einträge sind ideal, wenn ein Server mehrere Dienste bereitstellt (etwa `portal.example.test` verweist auf `web01.example.test`). Ändert sich später die IP-Adresse von `web01`, musst du nur dessen `A`-/`AAAA`-Record anpassen – alle Aliase folgen automatisch.

> ⚠ **CNAME-Regeln nach DNS-Standard:** Ein Hostname, für den ein `CNAME`-Record existiert, darf keine anderen Resource Records (`A`, `AAAA`, `MX`, `TXT` oder `NS`) tragen – und am Zonen-Apex (`example.test` selbst) ist ein `CNAME` grundsätzlich unzulässig, da dort bereits `SOA` und `NS` liegen.

Erstelle den Alias über Add-DnsServerResourceRecordCName und prüfe die Auflösungskette:

```
# CNAME-Alias portal.example.test auf web01.example.test einrichten
Add-DnsServerResourceRecordCName -Name "portal" -ZoneName "example.test" -HostNameAlias
"web01.example.test" -TimeToLive 01:00:00

# Abfrage liefert zuerst den CNAME und löst anschließend das Ziel auf
Resolve-DnsName -Name "portal.example.test" -Server 127.0.0.1 -DnsOnly -NoHostsFile
```

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Schematische Beispielausgabe (auf dem Zielsystem nicht live ausgeführt):

```
Name                      Type   TTL   Section    NameHost / IPAddress
----                      ----   ---   -------    --------------------
portal.example.test       CNAME  3600  Answer     web01.example.test
web01.example.test        A      3600  Answer     192.0.2.20
```

In der Antwortkette siehst du sofort beide Schritte: `portal.example.test` verweist über den `CNAME` auf `web01.example.test`, und der DNS-Server liefert im selben Antwortpaket direkt den zugehörigen `A`-Record mit.

PTR-Records (Reverse-Zuordnung)

Ein **`PTR`-Record** (*Pointer Record*) bildet die Gegenrichtung zum `A`- oder `AAAA`-Record: Er verknüpft eine IP-Adresse rückwärts mit einem Hostnamen (`-PtrDomainName`).

Wichtiger struktureller Unterschied: Während `A`, `AAAA` und `CNAME` in deiner Forward-Lookupzone `example.test` liegen, gehört ein `PTR`-Record **immer** in eine eigene Reverse-Lookupzone (für `192.0.2.20` beispielsweise unter `2.0.192.in-addr.arpa`). Als `-Name` gibst du dort nur den verbleibenden Hostanteil der IP-Adresse an (hier `"20"`), und als `-PtrDomainName` den vollständigen FQDN (`"web01.example.test."`).

```
# Beispielaufruf (setzt eine angelegte Reverse-Zone 2.0.192.in-addr.arpa voraus)
Add-DnsServerResourceRecordPtr -Name "20" -ZoneName "2.0.192.in-addr.arpa" -PtrDomainName
"web01.example.test." -TimeToLive 01:00:00
```

Wie du die passende Reverse-Lookupzone sauber aufbaust und `PTR`-Einträge anschließend per Reverse-Abfrage prüfst, behandeln wir direkt im nächsten Abschnitt.

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Bestehende Resource Records sicher ändern

Im Modul `DnsServer` überschreibt ein `Set-`-Befehl einen Eintrag nicht einfach anhand eines Namens – stattdessen übergibst du sowohl das alte unveränderte Objekt (`-OldInputObject`) als auch eine geänderte Kopie (`-NewInputObject`). Dieses Muster verhindert, dass beim Vorhandensein mehrerer Records (etwa bei Round-Robin-Einträgen) versehentlich der falsche Datensatz überschrieben wird.

□□ Praktisches Beispiel:

Die IPv4-Adresse von `web01.example.test` soll von `192.0.2.20` auf `192.0.2.25` umgestellt und die TTL für eine bevorstehende Migration auf 5 Minuten verkürzt werden:

```
# 1. Ausgangszustand: Bestehenden A-Record exakt auslesen
$oldRecord = Get-DnsServerResourceRecord -ZoneName "example.test" -Name "web01" -RRType
"A"

# 2. Arbeitskopie im Speicher erstellen und Werte anpassen
$newRecord = $oldRecord.Clone()
$newRecord.RecordData.IPv4Address = [System.Net.IPAddress]::Parse("192.0.2.25")
$newRecord.TimeToLive = [System.TimeSpan]::FromMinutes(5)

# 3. Änderung auf dem DNS-Server anwenden
Set-DnsServerResourceRecord -OldInputObject $oldRecord -NewInputObject $newRecord -
ZoneName "example.test"

# 4. Wirkung über eine echte Abfrage gegen den Server prüfen
Resolve-DnsName -Name "web01.example.test" -Type A -Server 127.0.0.1
```

Achte bei mehreren `A`-Records unter demselben Namen darauf, `$oldRecord` vor dem `.Clone()` über `Where-Object` exakt auf die gewünschte IP-Adresse einzugrenzen – andernfalls enthält `$oldRecord` ein Array und `.Clone()` schlägt fehl. Das folgende Muster zeigt, wie du bei zwei Round-Robin-Einträgen für `app01.example.test` (`192.0.2.30` und `192.0.2.31`) gezielt nur den zweiten Datensatz anpasst:

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
# Zwei A-Records unter demselben Hostnamen (DNS Round Robin)
Add-DnsServerResourceRecordA -Name "app01" -ZoneName "example.test" -IPv4Address
"192.0.2.30" -TimeToLive 00:05:00
Add-DnsServerResourceRecordA -Name "app01" -ZoneName "example.test" -IPv4Address
"192.0.2.31" -TimeToLive 00:05:00

# Nur den Record mit 192.0.2.31 eindeutig herausfiltern und ändern
$oldApp = Get-DnsServerResourceRecord -ZoneName "example.test" -Name "app01" -RRType "A" |
    Where-Object { $_.RecordData.IPv4Address.IPAddressToString -eq "192.0.2.31" }
$newApp = $oldApp.Clone()
$newApp.RecordData.IPv4Address = [System.Net.IPAddress]::Parse("192.0.2.32")

Set-DnsServerResourceRecord -OldInputObject $oldApp -NewInputObject $newApp -ZoneName
"example.test"
```

TTL-Planung vor geplanten Server-Umzügen

Bei einer TTL von `01:00:00` (3.600 Sekunden) halten Clients und zwischengeschaltete Resolver den alten `A`-Record im Extremfall bis zu einer vollen Stunde im Cache. Senke die TTL eines betroffenen Hosts daher **mindestens eine alte TTL-Periode vor der eigentlichen IP-Umstellung** auf `00:05:00` (300 Sekunden) ab, führe danach den IP-Wechsel durch und hebe die TTL erst nach erfolgreicher Verifikation wieder auf den Standardwert an.

Nicht mehr benötigte Einträge kontrolliert entfernen

Das Löschen eines Resource Records wirkt sich unmittelbar auf die Namensauflösung im Netz aus: Sobald der Eintrag fehlt, beantwortet `WIN-CORE01` Abfragen nach diesem Namen mit `NXDOMAIN` (oder Clients nutzen veraltete Cache-Einträge, bis deren TTL abläuft).

Verwende bei Remove-DnsServerResourceRecord vor dem eigentlichen Eingriff immer `-WhatIf`, um exakt zu prüfen, welches Objekt betroffen ist, und grenze den Datensatz über `-RecordData` eindeutig ein:

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
# Schritt 1: Löschvorgang vorab simulieren (ohne Änderung am Server)
Remove-DnsServerResourceRecord -ZoneName "example.test" -RRType "CNAME" -Name "portal" -
RecordData "web01.example.test." -WhatIf

# Schritt 2: Nach Prüfung ohne -WhatIf ausführen (PowerShell fragt zur Sicherheit nach)
Remove-DnsServerResourceRecord -ZoneName "example.test" -RRType "CNAME" -Name "portal" -
RecordData "web01.example.test."

# Schritt 3: Prüfen, ob der Eintrag aus der Zone entfernt wurde
Get-DnsServerResourceRecord -ZoneName "example.test" -Name "portal" -ErrorAction
SilentlyContinue
```

Vermeide pauschale Pipeline-Konstrukte mit `-Force` ohne vorangehenden Filter auf `-RRType` und `-RecordData`: Ein versehentlich gelöschter `NS`- oder Haupt-`A`-Record legt umgehend abhängige Dienste oder Aliase lahm.

Weitere praxisrelevante Records: SRV, TXT und MX

Neben `A`, `AAAA`, `CNAME` und `PTR` begegnen dir auch auf eigenständigen Workgroup-DNS-Servern regelmäßig drei weitere Record-Typen für Diensterkennung, Metadaten und Mail-Routing:

- **`TXT`-Records:** Speichern beliebige Textinformationen (etwa für interne Dienst-Verifizierungen oder Konfigurationshinweise).
- **`SRV`-Records:** Verknüpfen einen Dienstnamen samt Protokoll (z. B. `_http._tcp`) mit einem Zielhost, einem Port, einer Priorität und einer Gewichtung.
- **`MX`-Records:** Benennen den zuständigen Mail-Server (`MailExchange`) inklusive Priorität für eine Domain.

```
# TXT-Record zur internen Dienst-Dokumentation anlegen
Add-DnsServerResourceRecord -ZoneName "example.test" -Name "_meta" -Txt -DescriptiveText
```

```
"admindocs-lab-zone" -TimeToLive 01:00:00

# SRV-Record für einen internen HTTPS-Dienst auf web01.example.test (Port 443) anlegen
Add-DnsServerResourceRecord -ZoneName "example.test" -Name "_https._tcp" -Srv -DomainName
"web01.example.test." -Port 443 -Priority 10 -Weight 100 -TimeToLive 01:00:00

# MX-Record mit Priorität 10 anlegen (falls ein interner Mail-Relay existiert)
Add-DnsServerResourceRecordMX -ZoneName "example.test" -Name "." -MailExchange
"mail01.example.test." -Preference 10 -TimeToLive 01:00:00
```

Verschaffe dir nach größeren Anpassungen immer einen Gesamtüberblick über alle vorhandenen Einträge der Zone:

```
# Vollständigen Zonenbestand tabellarisch prüfen
Get-DnsServerResourceRecord -ZoneName "example.test" | Format-Table HostName, RecordType,
TimeToLive, RecordData
```

Schematische Beispielausgabe (auf dem Zielsystem nicht live ausgeführt):

```
HostName     RecordType TimeToLive RecordData
--------     ---------- ---------- ----------
@            NS         01:00:00   win-core01.example.test.
@            SOA        01:00:00   [5][win-core01.example.test.][hostmaster.example.test.]
_https._tcp  SRV        01:00:00   [10][100][443][web01.example.test.]
_meta        TXT        01:00:00   admindocs-lab-zone
app01        A          00:05:00   192.0.2.30
app01        A          00:05:00   192.0.2.32
web01        A          00:05:00   192.0.2.25
web01        AAAA       01:00:00   2001:db8::20
```

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Nach einem `dnscmd.exe /ZoneWriteBack example.test` spiegelt auch die physische Zonendatei `%SystemRoot%\System32\dns\example.test.dns` diesen Stand im Klartext wider (schematisches Beispiel):

```
;
;  Database file example.test.dns for example.test zone.
;      Zone version:  5
;

@                       IN  SOA win-core01.example.test.  hostmaster.example.test.  (
                                5            ; serial number
                                900          ; refresh
                                600          ; retry
                                86400        ; expire
                                3600       ) ; default TTL

;
;  Zone NS records
;

@                       NS      win-core01.example.test.

;
;  Zone records
;

_https._tcp             SRV     10 100 443 web01.example.test.
_meta                   TXT     ( "admindocs-lab-zone" )
app01                   300 A   192.0.2.30
app01                   300 A   192.0.2.32
web01                   300 A   192.0.2.25
web01                   AAAA    2001:db8::20
win-core01              A       192.0.2.10
```

Beachte zwei Details in der geschriebenen Zonendatei:

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

- **Abweichende TTLs:** Records mit veränderter TTL (`300` Sekunden für `app01` und `web01`) tragen ihren Sekundenwert explizit vor dem Typ (`300 A`), während Einträge mit der Zonen-Standard-TTL (`3600` Sekunden) ohne eigenen TTL-Wert geschrieben werden.
- **Automatische `SOA`-Seriennummer:** Jede Änderung per PowerShell erhöht `serial number` automatisch. Würdest du die Datei stattdessen im Notfall manuell im Editor anpassen, müsstest du diese Nummer zwingend manuell erhöhen (oft im Format `YYYYMMDDnn`, solange sie numerisch größer bleibt als der vorherige Wert), damit sekundäre Server und Caches die Änderung erkennen.

Reverse-Lookupzone konfigurieren

Am Ende des vorherigen Abschnitts stieß der `PTR`-Record bereits an eine natürliche Grenze: Solange `WIN-CORE01` keine passende Reverse-Lookupzone kennt, läuft sowohl `Add-DnsServerResourceRecordPtr` als auch der Parameter `-CreatePtr` beim Anlegen eines `A`-Records ins Leere. Genau diese Rückwärtsauflösung richten wir jetzt systematisch ein.

Zweck der Reverse-Namensauflösung im Betrieb

Auf den ersten Blick wirkt die Rückwärtsauflösung wie ein optionales Extra, da Clients ihre Ziele über Hostnamen (`A` / `AAAA`) finden. Im administrativen Alltag und in Protokollen begegnen dir jedoch ständig rohe IP-Adressen – und genau dort sorgt Reverse-DNS für Klarheit und Verlässlichkeit:

- **Diagnosewerkzeuge (`nslookup`):** Fragt ein Administrator eine DNS-Antwort per `nslookup` ab, versucht das Werkzeug im ersten Schritt immer, die IP-Adresse des antwortenden DNS-Servers selbst rückwärts aufzulösen. Fehlt ein `PTR`-Record für `192.0.2.10`, meldet `nslookup` direkt zu Beginn `Server: UnKnown`, was Einsteiger oft fälschlich als DNS-Ausfall interpretieren.
- **Protokolle und Monitoring:** Firewalls, SSH-Logs, Proxy-Server und Monitoring-Agenten zeichnen Verbindungen zunächst mit Quell-IPs auf. Ein konsistenter `PTR`-Bestand erlaubt eine eindeutige Zuordnung zu `web01.example.test` statt anonymer Adressen.
- **Forward-Confirmed Reverse DNS (FCrDNS):** Viele Dienste und Sicherheitsprüfungen (etwa Mail-Relays oder Zugriffsregeln) prüfen, ob ein `PTR`-Record auf einen Hostnamen

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

zeigt, dessen `A`-Record wiederum exakt auf dieselbe IP-Adresse zurückverweist.

Warum `in-addr.arpa` die Oktette umkehrt

DNS ist ein streng hierarchischer Baum, bei dem die allgemeinste Ebene ganz rechts steht (`.`, dann `.test`, dann `example.test`) und der spezifischste Teil ganz links (`web01`). Bei einer IPv4-Adresse ist die Logik genau umgekehrt: Der allgemeine Netzanteil steht links (`192.0.2`), der spezifische Hostanteil rechts (`.25`).

Damit DNS-Delegationen pro Netzsegment sauber funktionieren, dreht die Spezialdomain `in-addr.arpa` die Reihenfolge der Oktette vollständig um:

- Aus dem IPv4-Netz **`192.0.2.0/24`** wird der Zonenname **`2.0.192.in-addr.arpa`**.
- Die vollständige Reverse-Adresse für **`192.0.2.25`** lautet folglich **`25.2.0.192.in-addr.arpa`**.
- Innerhalb der Zone `2.0.192.in-addr.arpa` genügt für den `PTR`-Record deshalb als `-Name` das verbleibende Host-Oktett `"25"`.

Ausgangszustand prüfen und IPv4-Reverse-Lookupzone anlegen

Prüfe zunächst mit `Get-DnsServerZone`, ob auf `WIN-CORE01` bereits eine Reverse-Zone für `192.0.2.0/24` existiert:

```
# Ausgangszustand: Alle Zonen prüfen (Filter auf Reverse-Zonen)
Get-DnsServerZone | Where-Object { $_.IsReverseLookupZone -eq $true }
```

Auf einem neu eingerichteten Server liefert diese Abfrage keine Treffer (beziehungsweise nur automatisch erzeugte Standard-Reverse-Zonen für Loopback/Link-Local, sofern aktiviert).

Zum Anlegen der eigenständigen, dateibasierten IPv4-Reverse-Lookupzone nutzt du erneut Add-DnsServerPrimaryZone – diesmal mit dem Parameter `-NetworkID` statt `-Name`. Windows Server DNS berechnet daraus automatisch den korrekten `in-addr.arpa`-Zonennamen:

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
# Dateibasierte IPv4-Reverse-Lookupzone für 192.0.2.0/24 anlegen
Add-DnsServerPrimaryZone -NetworkID "192.0.2.0/24" -ZoneFile "2.0.192.in-addr.arpa.dns" -
DynamicUpdate "None" -PassThru
```

- **`-NetworkID "192.0.2.0/24"`:** Übergibt das IPv4-Subnetz in CIDR-Notation. Das Cmdlet leitet daraus automatisch den Zonennamen `2.0.192.in-addr.arpa` ab.
- **`-ZoneFile "2.0.192.in-addr.arpa.dns"`:** Speichert die Zonendaten unabhängig von Active Directory in `%SystemRoot%\System32\dns\2.0.192.in-addr.arpa.dns`.
- **`-DynamicUpdate "None"`:** Hält auch die Reverse-Zone unter manueller administrativer Kontrolle, sodass kein Client fremde `PTR`-Records injizieren kann.

Prüfe unmittelbar nach dem Erstellen die Zoneneigenschaften:

```
# Eigenschaften der neuen Reverse-Zone kontrollieren
Get-DnsServerZone -Name "2.0.192.in-addr.arpa" | Format-List ZoneName, ZoneType,
IsReverseLookupZone, IsDsIntegrated, DynamicUpdate, ZoneFile
```

Die schematische Beispielausgabe bestätigt die korrekte Einrichtung:

```
ZoneName            : 2.0.192.in-addr.arpa
ZoneType            : Primary
IsReverseLookupZone : True
IsDsIntegrated      : False
DynamicUpdate       : None
ZoneFile            : 2.0.192.in-addr.arpa.dns
```

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Auch in der Reverse-Zone hat Windows automatisch initiale `SOA`- und `NS`-Records angelegt. Passe bei Bedarf den `SOA`-`PrimaryServer` und die `ResponsiblePerson` analog zur Forward-Zone auf `win-core01.example.test.` an, damit die Metadaten beider Zonen konsistent sind.

Subnetzgröße und Eindeutigkeit von PTR-Records

Achte beim Anlegen von Reverse-Zonen darauf, dass das `-NetworkID`-Präfix exakt zur Netzstruktur passt: Ein `/16`-Netz wie `192.0.0.0/16` erzeugt die Zone `0.192.in-addr.arpa`, in der `PTR`-Namen zweistellig (`"25.2"` für `192.0.2.25`) gepflegt werden müssen. Ein `/24`-Netz erzeugt dagegen `2.0.192.in-addr.arpa` mit einfachem Host-Oktett (`"25"`). Vermeide außerdem mehrere `PTR`-Records für dieselbe IP-Adresse, sofern kein zwingender Sonderfall vorliegt – Diagnosewerkzeuge und Log-Auswertungen liefern sonst bei jeder Abfrage wechselnde oder mehrdeutige Hostnamen zurück.

PTR-Records für Server und bestehende Hosts anlegen

Nachdem `2.0.192.in-addr.arpa` aktiv ist, pflegst du für die bereits vergebenen IPv4-Adressen die passenden `PTR`-Records ein. Wichtig ist vor allem ein `PTR`-Record für den DNS-Server `WIN-CORE01` (`192.0.2.10`) selbst sowie für den in Abschnitt 6 auf `192.0.2.25` aktualisierten Host `web01.example.test`:

```
# PTR-Record für den DNS-Server WIN-CORE01 (192.0.2.10) anlegen
Add-DnsServerResourceRecordPtr -Name "10" -ZoneName "2.0.192.in-addr.arpa" -PtrDomainName
"win-core01.example.test." -TimeToLive 01:00:00

# PTR-Record für web01.example.test (192.0.2.25) anlegen
Add-DnsServerResourceRecordPtr -Name "25" -ZoneName "2.0.192.in-addr.arpa" -PtrDomainName
"web01.example.test." -TimeToLive 01:00:00
```

Achte beim Parameter `-PtrDomainName` auf den **vollständigen FQDN inklusive abschließendem Punkt** (`"web01.example.test."`). Ein vergessener Domainteil führt sonst zu unvollständigen Hostnamen in Reverse-Antworten.

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Zusammenspiel mit `-CreatePtr` bei neuen Hosts

Sobald eine passende Reverse-Lookupzone existiert, kannst du bei künftigen IPv4-Hosts beide Einträge in einem einzigen Schritt anlegen. Übergibst du bei `Add-DnsServerResourceRecordA` zusätzlich `-CreatePtr`, sucht der DNS-Server automatisch nach der passenden `in-addr.arpa`-Zone und trägt dort den `PTR`-Record ein:

```
# Forward-A-Record für app01.example.test inklusive automatischem PTR-Record anlegen
Add-DnsServerResourceRecordA -Name "app01" -ZoneName "example.test" -IPv4Address
"192.0.2.30" -CreatePtr -TimeToLive 01:00:00
```

Beachte dabei eine wichtige Grenze in der Praxis: Wenn du einen bestehenden `A`-Record nachträglich per `Set-DnsServerResourceRecord` auf eine andere IP-Adresse änderst (wie bei `web01` im vorigen Abschnitt) oder mit `Remove-DnsServerResourceRecord` löschst, wird ein zuvor manuell oder per `-CreatePtr` angelegter `PTR`-Record **nicht automatisch mitgeändert oder gelöscht**. Verwaiste `PTR`-Records auf alte Hostnamen musst du in der Reverse-Zone explizit bereinigen, damit Reverse- und Forward-Lookup konsistent bleiben.

Mit einem kurzen PowerShell-Abgleich zwischen der Forward-Zone `example.test` und der Reverse-Zone `2.0.192.in-addr.arpa` findest du solche verwaisten `PTR`-Einträge jederzeit ohne manuelles Suchen:

```
# Alle A-Records der Forward-Zone als Nachschlage-Tabelle erfassen
$forwardIps = Get-DnsServerResourceRecord -ZoneName "example.test" -RRType "A" |
    ForEach-Object { $_.RecordData.IPv4Address.IPAddressToString }

# Jeden PTR-Record prüfen, ob die zugehörige IP noch als A-Record existiert
Get-DnsServerResourceRecord -ZoneName "2.0.192.in-addr.arpa" -RRType "PTR" |
    ForEach-Object {
        $ip = "192.0.2.$($_.HostName)"
```

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
            [PSCustomObject]@{
                IP          = $ip
                PtrTarget   = $_.RecordData.PtrDomainName
                HatARecord  = ($forwardIps -contains $ip)
            }
        } | Format-Table -AutoSize
```

Einträge mit `HatARecord : False` zeigen direkt auf verwaiste `PTR`-Records (oder auf Hosts, deren `A`-Record bewusst in einer anderen Zone liegt) und lassen sich anschließend gezielt mit `Remove-DnsServerResourceRecord -WhatIf` prüfen und bereinigen.

Reverse-DNS-Abfragen gezielt verifizieren

Prüfe zunächst den Bestand in der Reverse-Zone und teste anschließend echte Reverse-Abfragen gegen `127.0.0.1`:

```
# 1. Bestand in der Reverse-Zone prüfen
Get-DnsServerResourceRecord -ZoneName "2.0.192.in-addr.arpa" -RRType "PTR"

# 2. Reverse-Abfrage direkt über die IPv4-Adresse mit Resolve-DnsName
Resolve-DnsName -Name "192.0.2.25" -Server 127.0.0.1

# 3. Explizite PTR-Abfrage über die in-addr.arpa-Notation
Resolve-DnsName -Name "25.2.0.192.in-addr.arpa" -Type PTR -Server 127.0.0.1
```

In der PowerShell erkennt `Resolve-DnsName` bei Angabe einer IPv4-Adresse automatisch, dass eine `PTR`-Abfrage gemeint ist, und übersetzt `192.0.2.25` intern in `25.2.0.192.in-addr.arpa`. Eine schematische Beispielausgabe zeigt das erwartete Ergebnis:

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
Name                           Type   TTL   Section    NameHost
----                           ----   ---   -------    --------
25.2.0.192.in-addr.arpa        PTR    3600  Answer     web01.example.test
```

Gegenprobe mit `nslookup` (sofern auch `192.0.2.10` seinen `PTR`-Record erhalten hat):

```
nslookup 192.0.2.25 127.0.0.1
```

Schematische Beispielausgabe bei vollständig konsistentem Forward- und Reverse-DNS:

```
Server:  win-core01.example.test
Address:  127.0.0.1

Name:    web01.example.test
Address:  192.0.2.25
```

Erscheint oben statt `win-core01.example.test` weiterhin `UnKnown`, fehlt der `PTR`-Record für die abgefragte Server-IP (oder bei Abfragen über `127.0.0.1` für das Loopback-Interface, während `nslookup 192.0.2.25 192.0.2.10` sauber `win-core01.example.test` auflöst).

Einordnung: Wie IPv6-Reverse-DNS (`ip6.arpa`) funktioniert

Für den Dual-Stack-Betrieb in Abschnitt 6 haben wir bereits einen `AAAA`-Record (`2001:db8::20`) angelegt. Während wir die praktische Einrichtung hier bewusst auf IPv4 beschränken, folgt IPv6-Reverse-DNS exakt demselben Prinzip – allerdings nicht oktettweise, sondern **Nibble-weise**

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

(jeweils 4 Bit pro Hex-Ziffer) unterhalb der Spezialdomain `ip6.arpa`.

Aus dem 32-Bit-Präfix `2001:0db8::/32` werden die acht Hex-Zeichen einzeln umgekehrt und durch Punkte getrennt:

- Präfix in voller Schreibweise: `2001:0db8`
- Umgekehrte Nibble-Zone: **`8.b.d.0.1.0.0.2.ip6.arpa`**

Auch hier nimmt dir `Add-DnsServerPrimaryZone -NetworkID "2001:db8::/32" -ZoneFile "..."` bei Bedarf das manuelle Umdrehen der Nibbles vollständig ab. Für `web01` (`2001:db8::20`) sähe eine solche Einrichtung in der Praxis so aus:

```
# Optional für Dual-Stack-Netze: IPv6-Reverse-Zone für 2001:db8::/32 anlegen
Add-DnsServerPrimaryZone -NetworkID "2001:db8::/32" -ZoneFile
"8.b.d.0.1.0.0.2.ip6.arpa.dns" -DynamicUpdate "None" -PassThru

# PTR-Record für 2001:db8::20 (24 umgekehrte Nibbles für den Hostanteil)
Add-DnsServerResourceRecordPtr -Name "0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0" -
ZoneName "8.b.d.0.1.0.0.2.ip6.arpa" -PtrDomainName "web01.example.test." -TimeToLive
01:00:00
```

Für einen sauberen **Forward-Confirmed Reverse DNS (FCrDNS)**-Abgleich gilt anschließend bei IPv4 wie IPv6 dieselbe Regel: Die Reverse-Abfrage auf die IP liefert einen `PTR` auf `web01.example.test`, und die anschließende Forward-Abfrage auf `web01.example.test` liefert exakt dieselbe IP-Adresse zurück.

> ⚠ **Typische Fehlerquellen bei Reverse-Zonen:** Schlägt `-CreatePtr` bei `Add-DnsServerResourceRecordA` stillschweigend fehl oder liefert `Resolve-DnsName` für eine lokale IP `NXDOMAIN`, stimmt fast immer die Netzmaske der Reverse-Zone nicht mit dem Subnetz überein (etwa eine `/16`-Zone `0.192.in-addr.arpa` statt `/24` `2.0.192.in-addr.arpa` oder umgekehrt falscher Host-Name im `PTR`-Record). Prüfe zudem bei abweichenden

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

> Subnetzgrößen (kleiner als `/24`), dass Windows Server DNS klassenlose Reverse-Delegationen nach RFC 2317 nicht automatisch über `-NetworkID` abbildet.

Forwarder und Rekursion konfigurieren

Mit `example.test` und `2.0.192.in-addr.arpa` beantwortet `WIN-CORE01` nun sämtliche internen Forward- und Reverse-Abfragen **autoritativ**. Fragt ein Client (oder der Server selbst) jedoch nach einer externen Domain wie `learn.microsoft.com`, kennt keine lokale Zone diesen Namen. Damit der DNS-Server solche Anfragen sauber auflöst – und nicht unkontrolliert ins Leere läuft –, konfigurierst du jetzt **Forwarder** und **Rekursion** gezielt über PowerShell.

Wie der DNS-Server zwischen Zone, Forwarder und Root Hints entscheidet

Bevor du Parameter änderst, solltest du die genaue Reihenfolge kennen, in der der Windows-DNS-Dienst jede eingehende Abfrage abarbeitet:

1. **Client-Anfrage an Port 53**Abfrage trifft lokal oder über das Netz auf WIN-CORE01 ein.
2. **Lokaler DNS-Cache**Liegt bereits eine gültige, nicht abgelaufene Antwort vor, liefert der Server sofort aus dem Cache.
3. **Autoritative Zonenprüfung**Gehört der Name zu example.test oder 2.0.192.in-addr.arpa, antwortet der Server verbindlich selbst.
4. **Rekursionsprüfung**Ist Rekursion deaktiviert, lehnt der Server fremde Namen ab und leitet nichts weiter.
5. **DNS-Forwarder**Bei aktiver Rekursion fragt der Server der Reihe nach die konfigurierten Upstream-Resolver.

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

6. **Root-Hints-Fallback**Antwortet kein Forwarder, löst der Server nur bei UseRootHint = True iterativ ab Root auf.

Unterschied zwischen Weiterleitung und direkter Rekursion

In der Praxis werden „Rekursion“ und „Forwarder“ häufig verwechselt. Technisch steuern beide Optionen unterschiedliche Mechanismen:

- **Rekursion (`Set-DnsServerRecursion`):** Der Grundschalter, der festlegt, ob `WIN-CORE01` überhaupt stellvertretend für Clients fremde Domains auflösen darf. Ist Rekursion vollständig deaktiviert (`-Enable $false`), arbeitet der Server rein autoritativ für seine eigenen Zonen – auch konfigurierte Forwarder werden dann nicht genutzt.
- **DNS-Forwarder (`Set-DnsServerForwarder`):** Delegiert die rekursive Arbeit für unbekannte Domains an einen oder mehrere vorgeschaltete DNS-Resolver (etwa den internen Labor-Router, eine zentrale Firewall oder einen ISP-Resolver). Der Forwarder übernimmt das Abklappern der DNS-Hierarchie und schickt `WIN-CORE01` das fertige Ergebnis zurück.
- **Root Hints (Stammhinweise):** Eine fest hinterlegte Liste der weltweiten Root-Nameserver (`a.root-servers.net` bis `m.root-servers.net`). Sind keine Forwarder eingetragen – oder schlagen alle Forwarder fehl und `-UseRootHint` steht auf `$true` –, beginnt Windows Server DNS selbst von der Wurzel (`.`) abwärts iterativ nach den zuständigen TLD- und Domain-Nameservern zu fragen.

Forwarder (`UseRootHint = False`)

Ausgehender Port 53
: Nur zum definierten Upstream-Resolver

Antwortzeit
: Nutzt meist warmen Cache des Upstream-Resolvers

Firewall-Kontrolle
: Streng auf `192.0.2.1` begrenzbar

Ausfallverhalten
: Klare `SERVFAIL`-Meldung bei Forwarder-Ausfall

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Direkte Root Hints (`UseRootHint = True`)

Ausgehender Port 53
: Zu beliebigen autoritativen Servern weltweit

Antwortzeit
: Mehrere iterative Rückfragen nötig

Firewall-Kontrolle
: Breite Port-53-Freigabe nach außen nötig

Ausfallverhalten
: Löst selbst weiter, scheitert in abgeschotteten Netzen mit Timeout

Gemeinsam: Setzen aktivierte Rekursion (`Enable : True`) auf `WIN-CORE01` voraus

> **Auswahl externer Forwarder ist umgebungsabhängig:** Es gibt keinen universell „besten“ öffentlichen DNS-Resolver. In Labor- und Unternehmensnetzen leiten Forwarder meist zuerst auf den vorgelagerten Router, die Standort-Firewall oder interne DNS-Filter weiter – dort gelten die bestehenden Sicherheits-, Logging- und Split-DNS-Regeln des Netzes. Externe Resolver kommen nur dann infrage, wenn ausgehender Port-53-Verkehr direkt ins Internet erlaubt und gewünscht ist.

Ausgangszustand von Rekursion und Forwardern prüfen

Lies zuerst die aktuelle Rekursions- und Forwarder-Konfiguration von `WIN-CORE01` aus:

```
# Ausgangszustand: Aktive Rekursionseinstellungen abfragen
Get-DnsServerRecursion

# Ausgangszustand: Konfigurierte Forwarder und Timeout prüfen
Get-DnsServerForwarder

# Vorhandene Root Hints kontrollieren (Ausgabe gekürzt)
```

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
Get-DnsServerRootHint | Select-Object -First 5
```

Im Standardzustand einer frischen Windows-Server-DNS-Installation ist die Rekursion aktiviert (`Enable : True`), während die Forwarder-Liste (`IPAddress`) zunächst leer ist und `UseRootHint` auf `True` steht. Ohne Anpassung würde der Server ab sofort versuchen, externe Namen direkt über die Root Hints im Internet aufzulösen – was in abgeschotteten Labornetzen ohne direkten ausgehenden Port-53-Zugriff auf fremde IPs zu langen Timeouts führt.

Forwarder und Fallback-Verhalten sauber konfigurieren

Im Beispielnetz `192.0.2.0/24` übernimmt der vorgelagerte Router/Resolver unter `192.0.2.1` die externe Auflösung (ersetze diese Adresse in deiner Umgebung durch die IP deines lokalen Gateways oder Upstream-Resolvers). Mit Set-DnsServerForwarder hinterlegst du den Forwarder, definierst das Timeout und legst verbindlich fest, ob bei dessen Ausfall auf Root Hints zurückgefallen werden darf:

```
# Upstream-Forwarder setzen, Timeout auf 3 Sekunden begrenzen und Root-Hints-Fallback
deaktivieren
Set-DnsServerForwarder -IPAddress "192.0.2.1" -Timeout 3 -UseRootHint $false -PassThru
```

Schematische Beispielausgabe (auf dem Zielsystem nicht live ausgeführt):

```
UseRootHint        : False
Timeout(s)         : 3
EnableReordering   : True
IPAddress          : {192.0.2.1}
```

- **`-IPAddress "192.0.2.1"`:** Überschreibt die Forwarder-Liste mit dem gewünschten Upstream-Resolver. Möchtest du einen bestehenden Forwarder lediglich ergänzen statt die Liste zu ersetzen, verwende `Add-DnsServerForwarder -IPAddress "..."`; zum gezielten Entfernen dient `Remove-DnsServerForwarder`.
- **`-Timeout 3`:** Wartet maximal 3 Sekunden auf eine Antwort dieses Forwarders, bevor der nächste Eintrag in der Liste angesprochen wird. Ein kurzes Timeout verhindert, dass Clients bei einem ausgefallenen ersten Forwarder unnötig lange hängen bleiben.
- **`-UseRootHint $false`:** Unterbindet den automatischen Rückfall auf Root Hints. Sind alle Forwarder nicht erreichbar, antwortet `WIN-CORE01` kontrolliert mit einem Fehler (`ServerFailure`), statt unkontrolliert direkte Root-Abfragen ins Internet zu versuchen.

Stelle zusätzlich sicher, dass die Rekursion auf dem Server aktiviert ist (`Enable : True`), damit der Forwarder-Pfad genutzt wird:

```
# Rekursion explizit prüfen und bei Bedarf aktivieren
Get-DnsServerRecursion
# Falls Enable auf False steht:
# Set-DnsServerRecursion -Enable $true -PassThru
```

Schematische Beispielausgabe (auf dem Zielsystem nicht live ausgeführt):

```
Enable                : True
AdditionalTimeout(s)  : 4
RetryInterval(s)      : 1
Timeout(s)            : 8
SecureResponse        : True
```

Bedingte Weiterleitung (Conditional Forwarder) für isolierte Teilnetze

---

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Manchmal soll nicht der gesamte externe DNS-Verkehr über denselben Upstream-Resolver laufen, sondern nur eine bestimmte Nachbar- oder Labor-Domain (etwa `lab.example.test`) an einen anderen internen DNS-Server (`192.0.2.50`). Statt den globalen Forwarder zu verändern, richtest du dafür mit `Add-DnsServerConditionalForwarderZone` eine gezielte bedingte Weiterleitung ein:

```
# Abfragen für lab.example.test gezielt an 192.0.2.50 weiterleiten
Add-DnsServerConditionalForwarderZone -Name "lab.example.test" -MasterServers 192.0.2.50 -
ForwarderTimeout 3 -PassThru
```

Alle übrigen unbekannten Domains gehen weiterhin über `192.0.2.1`, während `lab.example.test` direkt an den zuständigen Server `192.0.2.50` delegiert wird.

Mehrere Forwarder und direkte Server-Selbstprüfung (`Test-DnsServer`)

Stehen im Labor- oder Firmennetz zwei vorgelagerte Resolver bereit (etwa `192.0.2.1` und `192.0.2.2`), hinterlegst du beide Adressen als geordnete Liste. Mit `Test-DnsServer` prüfst du anschließend auf Knopfdruck, ob `WIN-CORE01` seine Forwarder tatsächlich erreichen und auflösen kann:

```
# Zwei redundante Forwarder mit 3 Sekunden Timeout konfigurieren
Set-DnsServerForwarder -IPAddress "192.0.2.1", "192.0.2.2" -Timeout 3 -UseRootHint $false
-PassThru

# Direkte Funktionsprüfung des DNS-Servers inklusive Forwarder-Kontext
Test-DnsServer -IPAddress "192.0.2.10"
Test-DnsServer -IPAddress "192.0.2.10" -Context Forwarder
```

Schematische Beispielausgabe von `Test-DnsServer -Context Forwarder` (auf dem Zielsystem nicht live ausgeführt):

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
IPAddress   Result     Message
---------   ------     -------
192.0.2.10  Success    DNS server 192.0.2.10 responded cleanly using configured
forwarders.
```

Erscheint statt `Success` ein Fehler, weist das Feld `Message` direkt auf eine fehlende Forwarder-Erreichbarkeit oder deaktivierte Rekursion hin. Soll einer der beiden Forwarder später gezielt aus der Liste entfernt werden, nutzt du `Remove-DnsServerForwarder -IPAddress "192.0.2.2" -Force`.

Beachte bei mehreren Forwardern das Feld `EnableReordering` (`True` im Standard): Antwortet der erste Forwarder deutlich langsamer oder fällt aus, sortiert Windows Server DNS die Liste im laufenden Betrieb dynamisch nach Antwortzeit um, bis ein Server wieder zuverlässig reagiert.

Für den produktiven Einzel-Lab-Betrieb ohne zweiten Upstream-Resolver belässt du es anschließend einfach bei dem einen erreichbaren Forwarder `192.0.2.1`.

Verhalten bei nicht erreichbaren Forwardern

Tritt beim Upstream-Resolver `192.0.2.1` eine Störung auf, arbeitet der Windows-DNS-Dienst die konfigurierte Liste streng sequenziell ab:

1. Er sendet die Anfrage an den ersten Forwarder und wartet bis zum Ablauf von `-Timeout` (hier 3 Sekunden).
2. Bleibt die Antwort aus, geht dieselbe Anfrage an den nächsten eingetragenen Forwarder.
3. Antwortet keiner der konfigurierten Forwarder, entscheidet `-UseRootHint`: Steht der Parameter auf `$true`, startet der Server eine eigene iterative Auflösung über die Root Hints. Steht er auf `$false`, bricht der Server die Anfrage sofort ab und meldet dem Client `SERVFAIL` (`ServerFailure`).

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Sicherheitsfolgen offener Rekursion

Sobald Rekursion aktiviert ist (`Enable : True`), beantwortet `WIN-CORE01` Anfragen nach beliebigen externen Domains – vorausgesetzt, die Firewall lässt den Client durch. Genau darin liegt ein klassisches Sicherheitsrisiko:

- **DNS Amplification / Reflection DDoS:** Ist ein rekursiver DNS-Server versehentlich aus dem öffentlichen Internet über UDP 53 erreichbar, wird er als **Open Resolver** missbraucht. Angreifer fälschen Quell-IP-Adressen und lösen mit kleinen Anfragen um ein Vielfaches größere Antworten an das Opfernetz aus.
- **Ressourcenerschöpfung & Cache Poisoning:** Unkontrollierte Anfragen von beliebigen Netzen füllen den DNS-Cache, lasten CPU sowie Bandbreite aus und vergrößern die Angriffsfläche für Cache-Manipulationen.

Beschränke eingehende Port-53-Regeln in der Windows Defender Firewall und auf der Netz-Firewall daher immer strikt auf die tatsächlich berechtigten internen Client-Subnetze – niemals auf `Any` aus dem Internet. Soll ein DNS-Server ausschließlich eigene Zonen im Internet veröffentlichen (rein autoritativer Server), bleibt Rekursion dauerhaft deaktiviert (`Set-DnsServerRecursion -Enable $false`).

DNS-Client des Servers umstellen und Weiterleitung testen

Erst jetzt, da sowohl die lokalen Zonen als auch der Upstream-Forwarder konfiguriert sind, greifen wir die Warnung aus Abschnitt 3 wieder auf: Du kannst nun die DNS-Clientkonfiguration von `WIN-CORE01` selbst sicher auf die eigene IP-Adresse (oder `127.0.0.1`) umstellen, ohne dass der Server seine externe Namensauflösung verliert:

```
# Vorhandene Schnittstelle ermitteln und eigene DNS-Adresse setzen (InterfaceAlias ggf. anpassen)
Set-DnsClientServerAddress -InterfaceAlias "Ethernet" -ServerAddresses ("192.0.2.10")
```

Prüfe abschließend beide Wege getrennt über `Resolve-DnsName`, damit autoritative Antworten und Forwarder-Auflösung sauber nachgewiesen sind:

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
# 1. Forwarder-Konfiguration am Server auslesen
Get-DnsServerForwarder

# 2. Autoritative lokale Zone testen (keine Weiterleitung)
Resolve-DnsName -Name "web01.example.test" -Server 192.0.2.10 -DnsOnly -NoHostsFile

# 3. Externe Namensauflösung über Forwarder testen (sofern Upstream im Labor erreichbar)
Resolve-DnsName -Name "learn.microsoft.com" -Server 192.0.2.10 -DnsOnly -NoHostsFile
```

Schematische Beispielausgabe für den externen Forwarder-Test (auf dem Zielsystem nicht live ausgeführt):

```
Name                                 Type   TTL   Section    IPAddress
----                                 ----   ---   -------    ---------
learn.microsoft.com                  CNAME  300   Answer
learn.microsoft.com.edgekey.net
learn.microsoft.com.edgekey.net      CNAME  60    Answer     e1234.dscx.akamaiedge.net
e1234.dscx.akamaiedge.net            A      20    Answer     203.0.113.50
```

In einer funktionierenden Umgebung liefert die lokale Zone sofort die in Abschnitt 6 gesetzte Adresse `192.0.2.25`, während die externe Abfrage über den Forwarder `192.0.2.1` aufgelöst wird und anschließend lokal im Cache von `WIN-CORE01` landet.

> ⚠ **Typische Fehlerquellen bei Forwardern:** Liefert die externe Abfrage trotz korrektem Forwarder einen Timeout, blockiert meist die Firewall ausgehenden Port 53 (`UDP/TCP`) von `WIN-CORE01` zum Upstream-Resolver, der Upstream-Resolver selbst lehnt rekursive Anfragen von der IP `192.0.2.10` ab (`REFUSED`) oder Rekursion wurde auf

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

> Serverebene deaktiviert (`Set-DnsServerRecursion -Enable $false`).

DNS-Abfragen und Troubleshooting

Sobald lokale Zonen, `PTR`-Einträge und Forwarder eingerichtet sind, beginnt im Alltag die eigentliche Bewährungsprobe: Ein Client meldet, ein Hostname sei nicht erreichbar, oder liefert nach einer IP-Änderung noch den alten Wert. Anstatt jetzt reflexartig die Windows-Firewall abzuschalten oder den `DNS`-Dienst pauschal neu zu starten, gehst du vom beobachtbaren Symptom aus und grenzt die Ursache Schritt für Schritt ein.

Saubere Testabfragen mit `Resolve-DnsName` und `nslookup`

Das größte Hindernis bei der DNS-Fehlersuche auf Windows-Clients sind unsichtbare Fallbacks: Schlägt eine reguläre Abfrage fehl, schaut Windows zusätzlich in die lokale `hosts`-Datei oder weicht auf LLMNR und NetBIOS aus. Damit du wirklich ausschließlich die Antwort deines DNS-Servers `192.0.2.10` siehst, erzwingst du bei Resolve-DnsName zwei entscheidende Schalter:

```
# Reine DNS-Abfrage ohne hosts-Datei und ohne LLMNR/NetBIOS direkt gegen WIN-CORE01
Resolve-DnsName -Name "web01.example.test" -Server 192.0.2.10 -DnsOnly -NoHostsFile

# Gezielte Prüfung einzelner Record-Typen (Forward, Reverse, SOA und extern via Forwarder)
Resolve-DnsName -Name "web01.example.test" -Type AAAA -Server 192.0.2.10 -DnsOnly -
NoHostsFile
Resolve-DnsName -Name "192.0.2.25" -Type PTR -Server 192.0.2.10 -DnsOnly -NoHostsFile
Resolve-DnsName -Name "example.test" -Type SOA -Server 192.0.2.10 -DnsOnly -NoHostsFile
Resolve-DnsName -Name "learn.microsoft.com" -Type A -Server 192.0.2.10 -DnsOnly -
NoHostsFile
```

- **`-Server 192.0.2.10`:** Ignoriert die auf der Netzwerkkarte hinterlegten DNS-Einstellungen und fragt gezielt deinen DNS-Server.
- **`-DnsOnly`:** Unterbindet Fallbacks auf LLMNR und NetBIOS.

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

- `-NoHostsFile`**:** Blendet Einträge aus `%SystemRoot%\System32\drivers\etc\hosts` aus.
- `-TcpOnly` **(optional bei Größenproblemen):** Zwingt die Abfrage auf TCP Port 53, um gezielt zu testen, ob Firewalls zwar UDP 53 durchlassen, aber TCP 53 verwerfen.

Zum schnellen Gegen-Check – auch von einem Linux-Administrationsclient aus – dient weiterhin `nslookup` (auf Linux alternativ `dig @192.0.2.10 web01.example.test`):

```
# Direkte Abfrage gegen 192.0.2.10 mit nslookup (Forward und Reverse)
nslookup web01.example.test 192.0.2.10
nslookup 192.0.2.25 192.0.2.10

# Nicht existierenden Namen gezielt prüfen (erwartet NXDOMAIN)
Resolve-DnsName -Name "gibt-es-nicht.example.test" -Server 192.0.2.10 -DnsOnly -
NoHostsFile
```

Schematische Beispielausgabe für `NXDOMAIN` (auf dem Zielsystem nicht live ausgeführt):

```
Resolve-DnsName : gibt-es-nicht.example.test : DNS name does not exist
```

Bei einem tatsächlich nicht vorhandenen Hostnamen innerhalb deiner autoritativen Zone muss der Server unmittelbar mit `DNS name does not exist` (`NXDOMAIN`) antworten – **nicht** mit einem Timeout. Ein sofortiges `NXDOMAIN` ist ein Beleg dafür, dass Netzwerk, Port 53 und der DNS-Dienst selbst einwandfrei arbeiten und lediglich der gesuchte Record fehlt.

Häufige DNS-Antwortcodes (RCODE) richtig deuten

Bevor du in die sechs konkreten Fehlerbilder einsteigst, hilft ein schneller Blick auf die vier wichtigsten DNS-Statuscodes in Diagnoseausgaben (`nslookup`, `dig` oder PowerShell-Fehlermeldungen):

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Antwortcode	Bedeutung	Typische Ursache in der Praxis
`NOERROR`	Anfrage erfolgreich	Entweder liegt ein Record vor – oder der Name existiert zwar (z. B. `web01` hat einen `A`-Record), aber der konkret abgefragte Typ (`AAAA`) fehlt (`NODATA`).
`NXDOMAIN`	Name existiert nicht	Der abgefragte Hostname existiert in der autoritativen Zone nicht (oder es fehlt die erwartete Subdomain).
`SERVFAIL`	Serverfehler	Der DNS-Server konnte die Anfrage nicht abschließen – meist ist der Forwarder nicht erreichbar, eine Zone pausiert (`IsPaused : True`) oder eine Zonendatei defekt.
`REFUSED`	Anfrage abgelehnt	Der Ziel-Resolver verweigert die Antwort (z. B. Rekursion für fremde Client-IPs nicht erlaubt oder Zugriffsrichtlinie am Upstream-Resolver).

Sechs Fehlerbilder anhand beobachtbarer Symptome eingrenzen

Je nach Fehlermeldung von `Resolve-DnsName` oder `nslookup` lässt sich die Ursache präzise auf eine von sechs Ebenen eingrenzen:

Symptom 1: Netzwerk- oder Routingproblem

- **Beobachtung:** `Resolve-DnsName` bricht mit `This operation returned because the timeout period expired` ab; auch Abfragen nach `example.test` oder `192.0.2.10` erreichen den Server nicht.
- **Prüfung:** Kontrolliere vom Client aus zunächst die grundlegende IP-Erreichbarkeit und speziell den TCP-Handshake auf Port 53:

```
# Grundlegende Erreichbarkeit und TCP-Port-53-Check vom Client aus
Test-NetConnection -ComputerName 192.0.2.10 -Port 53
```

Schematische Beispielausgabe bei erreichbarem DNS-Port (auf dem Zielsystem nicht live ausgeführt):

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
ComputerName     : 192.0.2.10
RemoteAddress    : 192.0.2.10
RemotePort       : 53
InterfaceAlias   : Ethernet
SourceAddress    : 192.0.2.25
TcpTestSucceeded : True
```

Schlägt schon `PingSucceeded` fehl (sofern ICMP erlaubt ist) und liefert `TcpTestSucceeded : False`, liegt ein Routing-, VLAN- oder IP-Adressproblem vor – oder die Netzwerkkarte auf `WIN-CORE01` besitzt nicht die erwartete statische IP.

Symptom 2: Nicht erreichbarer DNS-Dienst oder blockierende Firewall

- **Beobachtung:** Der Host `192.0.2.10` antwortet im Netz, aber sowohl `Resolve-DnsName -Server 192.0.2.10` auf dem Client läuft in ein Timeout als auch `TcpTestSucceeded` ist `False`. Lokal auf `WIN-CORE01` gegen `127.0.0.1` antwortet DNS unter Umständen ebenfalls nicht – oder nur lokal, aber nicht von außen.
- **Prüfung auf `WIN-CORE01`:**

```
# 1. Läuft der Dienst?
Get-Service -Name DNS | Select-Object Name, Status, StartType

# 2. Lauscht der Dienst auf 192.0.2.10 (nicht nur auf 127.0.0.1)?
Get-NetUDPEndpoint -LocalPort 53 | Format-Table LocalAddress, LocalPort, OwningProcess
Get-NetTCPConnection -LocalPort 53 -State Listen | Format-Table LocalAddress, LocalPort,
State

# 3. Welches Netzwerkprofil ist aktiv und erlauben die Firewall-Regeln genau dieses
Profil?
Get-NetConnectionProfile | Select-Object InterfaceAlias, NetworkCategory
Get-NetFirewallPortFilter | Where-Object { $_.LocalPort -eq 53 } | Get-NetFirewallRule |
```

```
    Select-Object DisplayName, Enabled, Profile, Action
```

Ist der Dienst gestoppt, startest du ihn gezielt und prüfst zuvor das Ereignisprotokoll auf die Absturz- oder Startursache. Läuft der Dienst und lauscht auf `0.0.0.0` beziehungsweise `192.0.2.10`, liegt das Problem bei lokal funktionierendem `127.0.0.1` an der Windows Defender Firewall – meist weil die Schnittstelle im Profil `Public` steht, während die DNS-Regel nur für `Private` / `Domain` gilt. Passe in diesem Fall gezielt das Schnittstellenprofil (`Set-NetConnectionProfile`) oder den Geltungsbereich der konkreten Port-53-Regel an, statt die Firewall global zu deaktivieren.

Symptom 3: Fehlende oder falsch benannte Zone

- **Beobachtung:** Du fragst `web01.example.test` ab, erwartest eine lokale Antwort, erhältst aber entweder `NXDOMAIN` von einem externen Resolver, `SERVFAIL` oder eine völlig unerwartete fremde IP.
- **Ursache:** Entweder existiert die Zone auf `WIN-CORE01` nicht unter dem exakten Namen (etwa Tippfehler `exampel.test`), oder die Zone ist angehalten/pausiert. Da der Server sich selbst nicht als autoritativ für den abgefragten Namen erkennt, leitet er die Anfrage wie eine fremde Domain über die Forwarder weiter.
- **Prüfung auf `WIN-CORE01`:**

```
# Existenz, Typ und Pausenzustand aller Zonen prüfen
Get-DnsServerZone | Format-Table ZoneName, ZoneType, IsPaused, IsShutdown, ZoneFile
```

Schematische Beispielausgabe bei einer versehentlich pausierten Zone (auf dem Zielsystem nicht live ausgeführt):

```
ZoneName              ZoneType IsPaused IsShutdown ZoneFile
--------              -------- -------- ---------- --------
example.test          Primary      True      False example.test.dns
2.0.192.in-addr.arpa  Primary     False      False 2.0.192.in-addr.arpa.dns
```

Achte auf `IsPaused : False` und `IsShutdown : False` sowie auf die exakte Schreibweise von `ZoneName`. Eine pausierte Zone aktivierst du direkt wieder mit `Resume-DnsServerZone -Name "example.test"`.

Symptom 4: Fehlerhafter oder fehlender Resource Record

- **Beobachtung:** `Resolve-DnsName -Name "example.test" -Type SOA -Server 192.0.2.10` antwortet sauber und autoritativ, aber die konkrete Abfrage nach `web01.example.test` liefert `NXDOMAIN` (oder `NoError` mit leerer `Answer`-Sektion bei `-Type AAAA`), oder ein `CNAME` verweist auf ein nicht auflösbares Ziel.
- **Prüfung auf `WIN-CORE01`:**

```
# Gesamten Zonenbestand kontrollieren und nach konkretem Namen filtern
Get-DnsServerResourceRecord -ZoneName "example.test" | Format-Table HostName, RecordType,
TimeToLive, RecordData
```

Schematische Beispielausgabe bei einem versehentlich doppelt angehängten Zonennamen oder falschem Alias-Ziel (auf dem Zielsystem nicht live ausgeführt):

```
HostName          RecordType TimeToLive RecordData
```

```
--------            ---------- ---------- ----------
portal              CNAME      01:00:00   web02.example.test.
web01.example.test A           00:05:00   192.0.2.25
```

Häufige Ursachen sind ein fehlender `AAAA`-Record bei IPv6-präferierenden Clients, ein Tippfehler im `-HostNameAlias` eines `CNAME` (`web02` statt `web01`), ein vergessener `PTR`-Eintrag nach einer IP-Änderung oder ein versehentlich angelegter Unterordner (`web01.example.test` statt `web01` als `-Name`, was effektiv zu `web01.example.test.example.test` führt).

Auch in der Gegenrichtung lässt sich ein fehlender Eintrag schnell durch eine gezielte Abfrage auf exakt den verdächtigen Namen eingrenzen:

```
# Prüfen, ob versehentlich ein doppelt qualifizierter Name in der Zone gelandet ist
Get-DnsServerResourceRecord -ZoneName "example.test" |
    Where-Object { $_.HostName -like "*example.test*" } |
    Format-Table HostName, RecordType, RecordData
```

Symptom 5: Rekursions- oder Forwarder-Problem

- **Beobachtung:** Interne Abfragen für `web01.example.test` und `192.0.2.25` funktionieren sofort, aber jede externe Abfrage (etwa `learn.microsoft.com`) endet mit einem Timeout oder `ServerFailure` (`SERVFAIL`).
- **Prüfung auf `WIN-CORE01`:**

```
# 1. Rekursion und Forwarder-Einstellungen am DNS-Server prüfen
Get-DnsServerRecursion
Get-DnsServerForwarder
```

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
# 2. Direkt vom Server WIN-CORE01 aus testen, ob der konfigurierte Forwarder 192.0.2.1
überhaupt antwortet
Resolve-DnsName -Name "learn.microsoft.com" -Server 192.0.2.1 -DnsOnly -NoHostsFile
```

Antwortet `192.0.2.1` direkt von `WIN-CORE01` aus nicht, liegt das Problem auf dem Upstream-Resolver oder auf dem ausgehenden Port-53-Pfad dorthin. Antwortet `192.0.2.1` im Direkt-Test hingegen sofort, `192.0.2.10` jedoch nicht, ist auf `WIN-CORE01` meist `Enable : False` bei `Get-DnsServerRecursion` gesetzt oder das Forwarder-Timeout zu knapp gewählt.

Symptom 6: DNS-Cache-Effekte

- **Beobachtung:** Du hast einen `A`-Record auf `WIN-CORE01` von `192.0.2.20` auf `192.0.2.25` geändert (`Get-DnsServerResourceRecord` zeigt bereits den neuen Wert), doch der Client – oder `WIN-CORE01` selbst bei externen Forwarder-Daten – liefert weiterhin die alte IP-Adresse oder eine alte `NXDOMAIN`-Negativantwort.
- **Ursache:** Sowohl der lokale DNS-Clientdienst (`Dnscache`) auf Windows-Clients als auch der `DNS`-Serverdienst selbst puffern Antworten für die Dauer der jeweiligen TTL (beziehungsweise bei `NXDOMAIN` bis zum Ablauf der `MinimumTimeToLive` aus dem `SOA`-Record).
- **Prüfung und gezielte Bereinigung:**

```
# Cache auf dem lokalen Client prüfen und bei Bedarf leeren
Get-DnsClientCache | Where-Object { $_.Entry -like "*example.test*" }
Clear-DnsClientCache

# Cache auf dem DNS-Server WIN-CORE01 prüfen und bei Bedarf leeren
Show-DnsServerCache | Where-Object { $_.HostName -like "*example.test*" }
Clear-DnsServerCache -Force
```

Schematische Beispielausgabe von `Get-DnsClientCache` vor der Bereinigung (auf dem Zielsystem nicht live ausgeführt):

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
Entry               RecordName          RecordType Status    TimeToLive Data
-----               ----------          ---------- ------    ---------- ----
web01.example.test  web01.example.test  A          Success          248 192.0.2.20
```

Solange `TimeToLive` noch herunterzählt, liefert der lokale Windows-Client die alte Adresse `192.0.2.20` direkt aus dem Speicher, ohne `WIN-CORE01` überhaupt erneut zu fragen. Nach dem Leeren beider Caches muss die nächste Abfrage gegen `192.0.2.10` unmittelbar den aktuellen Zonenwert liefern.

Windows-Ereignisprotokolle gezielt auswerten

Bleibt das Verhalten unklar – etwa wenn der Dienst nicht startet, eine Zonendatei Syntaxfehler enthält oder Konfigurationsänderungen nachvollzogen werden müssen –, liefern zwei Ereigniskanäle unter Windows Server 2025 belastbare Hinweise via `Get-WinEvent`:

- **Klassisches Dienstprotokoll (`DNS Server`):** Zeichnet Dienststarts, Ladefehler von Zonendateien (z. B. nach manueller Bearbeitung), fehlende Schreibrechte und Netzwerkbinding-Fehler auf.
- **Audit-Kanal (`Microsoft-Windows-DNSServer/Audit`):** Protokolliert standardmäßig aktiviert alle Änderungen an Zonen, Forwardern und Resource Records (wie Event-ID `515` beim Anlegen eines Resource Records oder Event-ID `541` beim Ändern einer Server- oder Zoneneinstellung).

```
# Letzte 20 Ereignisse (Fehler und Warnungen zuerst prüfen) aus dem DNS-Server-Dienstlog
Get-WinEvent -LogName "DNS Server" -MaxEvents 20 |
    Select-Object TimeCreated, Id, LevelDisplayName, Message |
    Format-List

# Letzte Konfigurations- und Record-Änderungen im Audit-Log prüfen
```

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
Get-WinEvent -FilterHashtable @{
    LogName   = 'Microsoft-Windows-DNSServer/Audit'
    StartTime = (Get-Date).AddHours(-24)
} -MaxEvents 20 | Select-Object TimeCreated, Id, Message
```

Schematische Beispielausgabe aus `Microsoft-Windows-DNSServer/Audit` (auf dem Zielsystem nicht live ausgeführt):

```
TimeCreated          Id Message
-----------          -- -------
12.11.2026 08:15:10 515 A resource record of type 1, name web01, TTL 3600 and RDATA
192.0.2.25 was created in scope Default of zone example.test.
12.11.2026 08:17:42 541 The setting SecureSecondaries on scope example.test has been set
to NoTransfer.
```

Solltest du vorübergehend einzelne Abfragen und Antworten auf Paketebene mitschneiden müssen, prüfe zuerst über `Get-DnsServerDiagnostics`, welche Protokollierung bereits aktiv ist. Aktiviere das analytische Log (`Microsoft-Windows-DNSServer/Analytical`) oder das temporäre Debug-Log per `Set-DnsServerDiagnostics` nur kurzzeitig für die gezielte Fehlersuche und schalte es danach wieder ab, um unnötige I/O-Last zu vermeiden:

```
# Aktuelle Diagnose-Protokollierung prüfen
Get-DnsServerDiagnostics

# Nur für akute Fehlersuche: Abfragen und Antworten vorübergehend in eine Datei schreiben
New-Item -Path "C:\Logs" -ItemType Directory -Force | Out-Null
Set-DnsServerDiagnostics -Queries $true -Answers $true -EnableLoggingToFile $true -
LogFilePath "C:\Logs\dns-debug.log"
```

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
# Nach Abschluss der Analyse sofort wieder deaktivieren
Set-DnsServerDiagnostics -Queries $false -Answers $false -EnableLoggingToFile $false
```

> **Keine blinden Neustarts bei der Diagnose:** Ein Neustart des `DNS`-Dienstes leert zwar zwangsläufig den Server-Cache und lädt Zonendateien neu, beseitigt aber weder Firewall-Blockaden noch fehlerhafte Zonen-Records oder defekte Forwarder – und verschleiert unter Umständen die eigentliche Ursache. Führe zuerst `Resolve-DnsName`, die Socket-/Firewall-Prüfung und `Get-WinEvent` durch, bevor du in den Dienst eingreifst.

Komplette 6-Punkte-Verifikation nach Änderungen

Nach jeder größeren Anpassung – und insbesondere bevor du Clients auf `192.0.2.10` umstellst – empfiehlt sich ein standardisierter Durchlauf über alle sechs Funktionsbereiche. Das folgende PowerShell-Prüfskript bündelt Dienst, Sockets, Zonen, Forward-/Reverse-Auflösung, `NXDOMAIN` und den Forwarder-Test in einem einzigen Ablauf:

```
# 1. Dienstzustand & Autostart prüfen
Get-Service -Name DNS | Select-Object Name, Status, StartType

# 2. Port-53-Listener (UDP & TCP) verifizieren
Get-NetUDPEndpoint -LocalPort 53 | Select-Object LocalAddress, LocalPort
Get-NetTCPConnection -LocalPort 53 -State Listen | Select-Object LocalAddress, LocalPort,
State

# 3. Geladene Zonen ohne Pause/Fehler prüfen
Get-DnsServerZone | Where-Object { -not $_.IsAutoCreated } |
    Format-Table ZoneName, ZoneType, IsPaused, IsShutdown, DynamicUpdate,
SecureSecondaries

# 4. Forward- und Reverse-Auflösung gegen 192.0.2.10 testen
Resolve-DnsName -Name "web01.example.test" -Type A -Server 192.0.2.10 -DnsOnly -
```

```
NoHostsFile
Resolve-DnsName -Name "192.0.2.25" -Type PTR -Server 192.0.2.10 -DnsOnly -NoHostsFile

# 5. Erwartetes NXDOMAIN für unbekannten internen Namen prüfen
try {
    Resolve-DnsName -Name "nxdomain-check.example.test" -Server 192.0.2.10 -DnsOnly -
NoHostsFile -ErrorAction Stop
} catch {
    "Erwartetes NXDOMAIN erhalten: $($_.Exception.Message)"
}

# 6. Externe Auflösung über Forwarder testen
Resolve-DnsName -Name "learn.microsoft.com" -Type A -Server 192.0.2.10 -DnsOnly -
NoHostsFile
```

EDNS0, UDP-Fragmentierung und Split-DNS-Stolperfallen

Zwei typische Fehlerbilder sorgen im Alltag selbst dann für Verwirrung, wenn Port 53 grundsätzlich erreichbar ist:

- **Große Antworten und TCP-Fallback:** Klassische DNS-Pakete über UDP waren ursprünglich auf 512 Byte begrenzt; mit EDNS0 sind größere UDP-Nutzlasten (typischerweise bis 1232 oder 4096 Byte) möglich. Überschreitet eine Antwort das UDP-Limit oder werden fragmentierte UDP-Pakete auf Zwischen-Firewalls verworfen, weicht der Client auf `53/TCP` aus. Funktioniert `Resolve-DnsName ... -DnsOnly` ohne Schalter, scheitert aber `Resolve-DnsName ... -TcpOnly`, blockiert eine Firewall gezielt `53/TCP`.
- **Split-DNS-Kollisionen:** Verwendest du intern exakt denselben Domainnamen wie im öffentlichen Internet (etwa eine reale Firmendomain), kennen interne Clients nur die auf `WIN-CORE01` gepflegten Records – jeder nicht lokal nachgepflegte öffentliche Subdomain-Eintrag (wie eine externe Web-App) liefert dann intern `NXDOMAIN`. Im Workgroup-Lab vermeidest du genau diese Pflegefalle durch die klare Trennung über `example.test`.

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Betrieb, Sicherheit und Wartung

Ein DNS-Server gilt nicht schon deshalb als produktionsreif, weil eine erste `Resolve-DnsName`-Abfrage erfolgreich war. Zwischen einem funktionierenden Lab-Aufbau und einem belastbaren Infrastrukturdienst liegen klare Zugriffsgrenzen, abgesicherte Zonentransfers, ein getestetes Backup- und Restore-Verfahren sowie eine feste Prüfroutine nach Updates und Neustarts.

Berechtigungen und administrativer Zugriff

Auf dem eigenständigen Workgroup-Server `WIN-CORE01` existiert kein Active Directory für zentrale Gruppenrichtlinien oder delegierte AD-Berechtigungen. Die Verwaltung des DNS-Dienstes liegt deshalb bei den lokalen Administratoren (`Administrators`) sowie der bei der Rolleninstallation angelegten lokalen Sicherheitsgruppe `DnsAdmins`:

- **Minimale Rechte im Alltag:** Nutze für die tägliche PowerShell-Administration über SSH nur dedizierte Administrationskonten statt geteilter Standardzugänge. Wer lediglich Zoneninhalte pflegen soll, benötigt nicht zwingend volle Systemrechte, solange entsprechende Delegationsgrenzen über `DnsAdmins` eingehalten werden – im reinen Workgroup-Betrieb bleiben Änderungen an Dienst, Registry und Firewall jedoch Aufgabe lokaler Administratoren.
- **Lausch-Adressen gezielt festlegen:** Besitzt ein Server mehrere Netzwerkschnittstellen (etwa getrennt für Client-Netz und Management-VLAN), sollte der DNS-Dienst nicht unkontrolliert auf allen Adressen lauschen. Über `Get-DnsServerSetting` und `Set-DnsServerSetting` lässt sich die Bindung auf die definierte Server-IP `192.0.2.10` beschränken:

```
# Aktuelle Lausch-Adressen prüfen
(Get-DnsServerSetting -All).ListeningIpAddress

# DNS-Dienst explizit auf 192.0.2.10 und Loopback 127.0.0.1 binden
$dnsSettings = Get-DnsServerSetting -All
$dnsSettings.ListeningIpAddress = @("192.0.2.10", "127.0.0.1")
```

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
Set-DnsServerSetting -InputObject $dnsSettings -PassThru |
    Select-Object -ExpandProperty ListeningIpAddress
```

Nimm `127.0.0.1` bewusst mit in die Liste auf, wenn lokale Skripte auf `WIN-CORE01` weiterhin gegen Loopback testen sollen – andernfalls nimmt der Dienst auf `127.0.0.1` keine Abfragen mehr an.

- **Firewall-Regeln auf das Client-Subnetz eingrenzen:** In Abschnitt 4 haben wir geprüft, ob Port 53 grundsätzlich geöffnet ist. Für den produktiven Betrieb solltest du sicherstellen, dass die aktiven Port-53-Regeln den Zugriff über `-RemoteAddress` auf die tatsächlichen Client- und Management-Netze (hier exemplarisch `192.0.2.0/24`) eingrenzen und nicht für beliebige Quelladressen aus fremden Netzen offenstehen:

```
# Bestehende Port-53-Firewall-Regeln samt Quelladress-Filter prüfen
Get-NetFirewallPortFilter |
    Where-Object { $_.LocalPort -eq 53 } |
    Get-NetFirewallRule |
    ForEach-Object {
        $addr = $_ | Get-NetFirewallAddressFilter
        [PSCustomObject]@{
            DisplayName   = $_.DisplayName
            Enabled       = $_.Enabled
            Profile       = $_.Profile
            RemoteAddress = $addr.RemoteAddress
        }
    } | Format-Table -AutoSize
```

> ⚠ **SSH- und DNS-Schutz greifen ineinander:** Wie im Grundlagenartikel beschrieben, ist Port `22/TCP` der administrative Zugang zu `WIN-CORE01`. Beschränke den SSH-Zugriff auf dein Administrationsnetz und Port `53/UDP` sowie `53/TCP` auf die Netze, die diesen Resolver tatsächlich nutzen dürfen.

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Zonentransfers absichern und dynamische Updates kontrollieren

Standardmäßig solltest du niemals zulassen, dass beliebige Hosts per Zonentransfer (`AXFR` oder `IXFR`) den gesamten Inhalt deiner Forward- und Reverse-Zonen abziehen können. Eine offene Zone legt sonst sämtliche internen Hostnamen, IP-Strukturen und Dienste offen.

Mit <code class="badge-code">Set-DnsServerPrimaryZone legst du für jede primäre Zone fest, welche Gegenstellen überhaupt einen Transfer anfordern dürfen:

- `NoTransfer`: Verbietet Zonentransfers vollständig – die richtige Standardwahl, solange `WIN-CORE01` allein ohne Secondary-DNS betrieben wird.
- `TransferToSecureServers`: Erlaubt Transfers ausschließlich an explizit über `-SecondaryServers` benannte IP-Adressen (sinnvoll, sobald du später einen zweiten DNS-Server zur Ausfallsicherheit ergänzt).
- `TransferToZoneNameServer`: Beschränkt Transfers auf Server, die als `NS`-Record in der Zone eingetragen sind.
- `TransferAnyServer`: Erlaubt Transfers an jeden beliebigen Anfrager und ist im Produktivbetrieb zu vermeiden.

```
# Zonentransfers für beide Zonen auf WIN-CORE01 vollständig unterbinden
Set-DnsServerPrimaryZone -Name "example.test" -SecureSecondaries "NoTransfer" -PassThru |
    Select-Object ZoneName, SecureSecondaries, DynamicUpdate

Set-DnsServerPrimaryZone -Name "2.0.192.in-addr.arpa" -SecureSecondaries "NoTransfer" -
PassThru |
    Select-Object ZoneName, SecureSecondaries, DynamicUpdate
```

Schematische Beispielausgabe (auf dem Zielsystem nicht live ausgeführt):

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
ZoneName              SecureSecondaries DynamicUpdate
--------              ----------------- -------------
example.test          NoTransfer        None
2.0.192.in-addr.arpa  NoTransfer        None
```

Kontrolliere dabei gleichzeitig `DynamicUpdate`: Im Workgroup-Betrieb ohne Active Directory steht `Secure` nicht zur Verfügung, und `NonsecureAndSecure` würde unauthentifizierte Überschreibungen bestehender Records durch beliebige Clients im Netz ermöglichen. Belasse `DynamicUpdate` deshalb auf `None` und pflege Records kontrolliert per PowerShell.

Trennung von autoritativer und rekursiver Funktion

In Abschnitt 8 wurde bereits erläutert, warum ein Server mit aktivierter Rekursion niemals aus dem Internet oder nicht vertrauenswürdigen Netzen auf Port 53 erreichbar sein darf: Ein offener Resolver lässt sich für DNS-Amplification-Angriffe missbrauchen und vergrößert die Angriffsfläche für Cache Poisoning.

Für den Betrieb ergeben sich daraus zwei klare Betriebsmodelle:

- **Interner Kombinations-Resolver (`WIN-CORE01`):** Beantwortet interne Zonen autoritativ und löst externe Namen über `192.0.2.1` auf. Voraussetzung ist zwingend die oben gezeigte Netzwerk- und Firewall-Begrenzung auf vertrauenswürdige interne Subnetze.
- **Rein autoritativer DNS-Server (ohne Client-Forwarding):** Soll ein Server ausschließlich eigene Zonen für andere Resolver ausliefern und keinerlei fremde Namen auflösen, schaltest du die Rekursion explizit ab (`Set-DnsServerRecursion -Enable $false`).

Backup und Restore dateibasierter DNS-Zonen

Für die Datensicherung ist die technische Architektur aus Abschnitt 2 entscheidend: Auf einem Active-Directory-integrierten DNS-Server liegen Zonen in der AD-Datenbank und werden über System-State-Backups oder `Export-DnsServerZone` gesichert. Auf unserem eigenständigen Workgroup-Server (`IsDsIntegrated : False`) liegen die eigentlichen Resource Records jedoch **bereits als Textdateien** im Dateisystem unter `%SystemRoot%\System32\dns\`, während die

zugehörigen Zonen-Metadaten (Zonentyp, `DynamicUpdate`, `SecureSecondaries`) und Server-Parameter (Forwarder, Rekursion, Lausch-Adressen) in der Windows-Registry gespeichert sind.

Für ein vollständiges, konsistentes Backup eines dateibasierten DNS-Servers musst du daher **beide** Ebenen sichern:

1. Alle `.dns`-Zonendateien aus `$env:SystemRoot\System32\dns\` (`example.test.dns`, `2.0.192.in-addr.arpa.dns` sowie `cache.dns`).
2. Die beiden DNS-Registry-Pfade `HKLM\SYSTEM\CurrentControlSet\Services\DNS\Parameters` und `HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\DNS Server\Zones`.

```
# Reproduzierbares DNS-Backup (Zonendateien + Registry + lesbare Konfiguration)
$backupRoot = "C:\Backup\DNS\$(Get-Date -Format 'yyyy-MM-dd_HHmm')"
New-Item -Path $backupRoot -ItemType Directory -Force | Out-Null

# 1. Zonendateien sichern
Copy-Item -Path "$env:SystemRoot\System32\dns\*.dns" -Destination $backupRoot -Force

# 2. Registry-Konfiguration exportieren
reg.exe export "HKLM\SYSTEM\CurrentControlSet\Services\DNS\Parameters" "$backupRoot\dns-parameters.reg" /y
reg.exe export "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\DNS Server\Zones" "$backupRoot\dns-zones.reg" /y

# 3. Lesbare Gesamtkonfiguration als zusätzliche Referenz sichern
Get-DnsServer | Export-Clixml -Path "$backupRoot\dns-server-config.clixml"

Get-ChildItem -Path $backupRoot | Select-Object Name, Length, LastWriteTime
```

Schematische Beispielausgabe des vollständigen Backup-Satzes (auf dem Zielsystem nicht live ausgeführt):

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
Name                      Length LastWriteTime
----                      ------ -------------
2.0.192.in-addr.arpa.dns     612 12.11.2026 08:20:01
cache.dns                   3328 12.11.2026 08:20:01
dns-parameters.reg          4180 12.11.2026 08:20:01
dns-server-config.clixml   18420 12.11.2026 08:20:01
dns-zones.reg               2940 12.11.2026 08:20:01
example.test.dns             845 12.11.2026 08:20:01
```

> ⚠ **Keine Zonendateien im laufenden Betrieb überschreiben:** Der `DNS`-Dienst hält Zonenänderungen zunächst im Arbeitsspeicher und schreibt sie verzögert oder beim Beenden auf die Platte. Für einen Restore musst du den Dienst deshalb zuerst stoppen, Dateien und Registry zurückspielen und den Dienst danach wieder starten.

Im Wiederherstellungsfall gehst du in exakt dieser Reihenfolge vor:

```
# Beispielhafter Restore-Ablauf (Pfad an den tatsächlichen Backup-Ordner anpassen)
$restoreFrom = "C:\Backup\DNS\2026-11-12_0800"

# 1. DNS-Dienst sauber anhalten
Stop-Service -Name DNS

# 2. Zonendateien und Registry-Einträge wiederherstellen
Copy-Item -Path "$restoreFrom\*.dns" -Destination "$env:SystemRoot\System32\dns\" -Force
reg.exe import "$restoreFrom\dns-parameters.reg"
reg.exe import "$restoreFrom\dns-zones.reg"

# 3. Dienst starten und Zustand sofort verifizieren
Start-Service -Name DNS
Get-Service -Name DNS | Select-Object Name, Status, StartType
```

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
Get-DnsServerZone | Format-Table ZoneName, ZoneType, IsPaused, IsShutdown
Resolve-DnsName -Name "web01.example.test" -Server 192.0.2.10 -DnsOnly -NoHostsFile
```

Ein Backup gilt erst dann als belastbar, wenn du diesen Restore-Weg einmal kontrolliert auf einem Testsystem oder in einem Wartungsfenster durchgespielt hast.

Typische Fehlerbilder beim Backup und Restore

- **Nur `.dns`-Dateien zurückgespielt, Registry vergessen:** Der `DNS`-Dienst liest beim Start nicht einfach blind alle `.dns`-Dateien im Ordner `System32\dns`, sondern lädt ausschließlich Zonen, die unter `HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\DNS Server\Zones` registriert sind. Ohne den Registry-Export tauchen die Zonen nach einem Server-Neuaufbau in `Get-DnsServerZone` nicht auf.
- **Registry importiert, aber `.dns`-Datei fehlt oder hat falsche Rechte:** Die Zone erscheint zwar in `Get-DnsServerZone`, schlägt beim Laden aber fehl (`IsShutdown : True` oder Fehler im Ereignisprotokoll `DNS Server`), weil die zugehörige Datei unter `%SystemRoot%\System32\dns\` fehlt oder der Dienst nicht darauf zugreifen kann.
- **Dateien bei laufendem Dienst ersetzt:** Schreibt `dns.exe` beim späteren Herunterfahren seinen Speicherstand zurück, überschreibt er deine manuell kopierten Restore-Dateien wieder. Halte den Dienst vor jedem Restore daher immer konsequent mit `Stop-Service -Name DNS` an.

Automatisierte tägliche Sicherung per Aufgabenplanung

Damit Backups im Alltag nicht vergessen werden, legst du auf Windows Server 2025 Core ein kleines Sicherungsskript unter `C:\Scripts\Backup-DnsServer.ps1` ab und planst es über die PowerShell-Task-Scheduler-Cmdlets einmal täglich um 02:00 Uhr als `SYSTEM` ein:

```
# Backup-Skript anlegen (inkl. Bereinigung von Sicherungen älter als 30 Tage)
New-Item -Path "C:\Scripts" -ItemType Directory -Force | Out-Null
```

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
@'
$backupRoot = "C:\Backup\DNS\$(Get-Date -Format 'yyyy-MM-dd_HHmm')"
New-Item -Path $backupRoot -ItemType Directory -Force | Out-Null
Copy-Item -Path "$env:SystemRoot\System32\dns\*.dns" -Destination $backupRoot -Force
reg.exe export "HKLM\SYSTEM\CurrentControlSet\Services\DNS\Parameters" "$backupRoot\dns-
parameters.reg" /y | Out-Null
reg.exe export "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\DNS Server\Zones"
"$backupRoot\dns-zones.reg" /y | Out-Null
Get-DnsServer | Export-Clixml -Path "$backupRoot\dns-server-config.clixml"
Get-ChildItem -Path "C:\Backup\DNS" -Directory |
    Where-Object { $_.CreationTime -lt (Get-Date).AddDays(-30) } |
    Remove-Item -Recurse -Force
'@ | Set-Content -Path "C:\Scripts\Backup-DnsServer.ps1" -Encoding UTF8

# Geplante Aufgabe täglich um 02:00 Uhr als SYSTEM registrieren
$action    = New-ScheduledTaskAction -Execute "powershell.exe" -Argument "-NoProfile -
ExecutionPolicy Bypass -File C:\Scripts\Backup-DnsServer.ps1"
$trigger   = New-ScheduledTaskTrigger -Daily -At "02:00"
$principal = New-ScheduledTaskPrincipal -UserId "SYSTEM" -LogonType ServiceAccount -
RunLevel Highest
Register-ScheduledTask -TaskName "Backup-WinCore01-DNS" -Action $action -Trigger $trigger
-Principal $principal -Force
```

Cache-Wartung, Event-Logs und Überwachung im Alltag

Im laufenden Betrieb benötigt der Windows-DNS-Server nur wenig manuelle Eingriffe, profitiert aber von regelmäßigen Kontrollen:

- **Cache-Einstellungen prüfen:** Mit `Get-DnsServerCache` kontrollierst du unter anderem `MaxTTL`, `MaxNegativeTtl` und `MaxKBSize` sowie `EnablePollutionProtection` (Schutz gegen unaufgeforderte Zusatzdaten in Antworten). Den Cache leerst du mit `Clear-DnsServerCache -Force` nur dann, wenn veraltete Einträge nach einer Änderung gezielt verworfen werden müssen – nicht pauschal per Skript im Stundentakt.

```
# Aktuelle Cache-Parameter prüfen
```

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
Get-DnsServerCache | Format-List MaxTTL, MaxNegativeTtl, MaxKBSize,
EnablePollutionProtection
```

Schematische Beispielausgabe (auf dem Zielsystem nicht live ausgeführt):

```
MaxTTL                   : 1.00:00:00
MaxNegativeTtl           : 00:15:00
MaxKBSize                : 0
EnablePollutionProtection : True
```

Möchtest du im Labor- oder Produktivbetrieb verhindern, dass fehlerhafte `NXDOMAIN`-Antworten von externen Forwardern lange zwischengespeichert werden, begrenzt du `MaxNegativeTtl` gezielt per `Set-DnsServerCache`:

```
# Negatives Caching auf 5 Minuten begrenzen und Cache-Größe optional deckeln (z. B. 51200
KB = 50 MB)
Set-DnsServerCache -MaxNegativeTtl 00:05:00 -MaxKBSize 51200 -PassThru
```

- **Statische Records und Alterung (Scavenging):** Alle über PowerShell manuell angelegten Records tragen standardmäßig `Timestamp : 0` (statisch) und werden vom DNS-Aging niemals automatisch gelöscht. In einer Workgroup-Umgebung ohne dynamische Client-Registrierung (`DynamicUpdate : None`) bleibt das automatische Scavenging (`Get-DnsServerScavenging`) deaktiviert, sodass auch bei künftigen Anpassungen keine manuellen Einträge verloren gehen.
- **Ereignisprotokolle auswerten:** Prüfe regelmäßig `Get-WinEvent -LogName "DNS Server"` auf Fehler (`LevelDisplayName -eq 'Error'`) sowie `Microsoft-Windows-`

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

`DNSServer/Audit` auf unerwartete Konfigurationsänderungen, wie in Abschnitt 9 gezeigt.

- **Dienst- und Zonenstatus automatisieren:** Eine kurze Prüfroutine per PowerShell (Dienststatus `Running`, Ports 53 offen, `IsPaused : False` für alle Zonen, interne und externe Testauflösung) lässt sich direkt nach Änderungen oder aus einem Monitoring-Check heraus ausführen.

Verhalten bei Windows Updates und Neustarts

Sicherheitsupdates für Windows Server 2025 Core erfordern regelmäßig einen Server-Neustart. Da während eines Neustarts von `WIN-CORE01` keine DNS-Auflösung über `192.0.2.10` stattfindet, sollten Updates ausschließlich in angekündigten Wartungsfenstern erfolgen – idealerweise gibt es im Produktivnetz mindestens einen zweiten DNS-Server, damit Clients während eines Reboots nahtlos ausweichen können.

Prüfe nach jedem Patch-Zyklus und Neustart über deine SSH-Verbindung direkt drei Punkte:

1. Steht der Dienst `DNS` auf `StartType : Automatic` und `Status : Running`?
2. Sind alle Zonen ohne Pause geladen (`Get-DnsServerZone` mit `IsPaused : False` und `IsShutdown : False`)?
3. Antworten sowohl interne Records (`web01.example.test`, `192.0.2.25`) als auch externe Namen über den Forwarder (`learn.microsoft.com`)?

```
# Vor dem Patch-Neustart: Geplantes DNS-Backup einmal sofort anstoßen
Start-ScheduledTask -TaskName "Backup-WinCore01-DNS"

# Kompakte Nachkontrolle direkt nach einem Windows-Update-Neustart
(Get-CimInstance -ClassName Win32_OperatingSystem).LastBootUpTime
Get-Service -Name DNS | Select-Object Name, Status, StartType
Get-DnsServerZone | Where-Object { -not $_.IsAutoCreated } | Format-Table ZoneName,
IsPaused, IsShutdown
Get-WinEvent -LogName "DNS Server" -MaxEvents 5 | Select-Object TimeCreated, Id,
LevelDisplayName, Message
Resolve-DnsName -Name "web01.example.test" -Server 192.0.2.10 -DnsOnly -NoHostsFile
```

```
Resolve-DnsName -Name "192.0.2.25" -Type PTR -Server 192.0.2.10 -DnsOnly -NoHostsFile
```

Kriterien für die Produktionsreife

Bevor `WIN-CORE01` endgültig als produktiver DNS-Server für Clients im Netz eingetragen wird, sollten sämtliche folgenden Bedingungen nachweislich erfüllt sein:

- **Netzwerk und Adresse:** Statische IPv4-Konfiguration (`192.0.2.10/24`) und getesteter DNS-Client-Eintrag am Server selbst (ohne Ausfall bei leerem Cache).
- **Dienst und Bindung:** `DNS`-Dienst auf `Automatic`, Port `53/UDP` und `53/TCP` lauschen auf `192.0.2.10`.
- **Zugriffsschutz:** Firewall-Regeln für Port 53 (und SSH Port 22) auf bekannte interne Client- und Management-Subnetze begrenzt; kein offener Resolver nach außen.
- **Zonensicherheit:** `DynamicUpdate` steht auf `None`, `SecureSecondaries` steht auf `NoTransfer` (oder explizit auf benannte Secondary-Server).
- **Forwarder und Namensauflösung:** Forwarder getestet, Forward- (`A`, `AAAA`, `CNAME`) und Reverse-Zone (`PTR`) einschließlich `NXDOMAIN`-Verhalten per `Resolve-DnsName` verifiziert.
- **Sicherung und Redundanz:** Aktuelles Backup der `.dns`-Dateien und Registry-Schlüssel liegt vor und wurde im Restore getestet; für produktive Umgebungen ohne akzeptable Ausfallzeit ist zusätzlich ein zweiter DNS-Server eingeplant.

Kontrollierte Umstellung produktiver Clients

Sobald sämtliche Kriterien für die Produktionsreife erfüllt sind, stellst du Clients im Netz nicht schlagartig auf einmal um, sondern in drei kontrollierten Schritten:

1. **Einzelnen Testclient umstellen:** Trage `192.0.2.10` zunächst nur auf einem administrativen Testclient ein (`Set-DnsClientServerAddress`) und prüfe dort interne (`web01.example.test`, `192.0.2.25`) sowie externe Namen (`learn.microsoft.com`) inklusive Browser- und SSH-Zugriffen.
2. **DHCP-Option 006 schrittweise anpassen:** Erst wenn der Testclient stabil arbeitet, hinterlegst du `192.0.2.10` (beziehungsweise bei Redundanz `192.0.2.10` und `192.0.2.11`) in DHCP-Option `006` (*DNS Servers*) für das gewünschte Subnetz.

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

3. **Statisch konfigurierte Server nachziehen:** Hosts mit fester IP-Konfiguration erhalten den neuen DNS-Server gezielt nacheinander. Prüfe auf jedem umgestellten System per `ipconfig /all` oder `Get-DnsClientServerAddress`, dass keine alten, nicht mehr erreichbaren DNS-Adressen als primärer Eintrag stehen bleiben.

```
# Auf einem Windows-Testclient: DNS-Server umstellen, Cache leeren und reale Client-Auflösung prüfen
Set-DnsClientServerAddress -InterfaceAlias "Ethernet" -ServerAddresses ("192.0.2.10")
Clear-DnsClientCache

# Kontrolle aller Schnittstellen (auch IPv6!), welcher DNS-Server tatsächlich genutzt wird
Get-DnsClientServerAddress | Where-Object { $_.ServerAddresses.Count -gt 0 } |
    Format-Table InterfaceAlias, AddressFamily, ServerAddresses

# Abfrage ohne -Server-Parameter testen (nutzt die OS-Konfiguration des Clients)
Resolve-DnsName -Name "web01.example.test" -DnsOnly -NoHostsFile
Resolve-DnsName -Name "learn.microsoft.com" -DnsOnly -NoHostsFile
```

> **IPv6-DNS-Server auf Clients nicht übersehen:** Erhält ein Windows-Client im Dual-Stack-Netz per Router Advertisement (RDNSS) oder DHCPv6 weiterhin eine fremde IPv6-DNS-Adresse (etwa die des Routers), fragt Windows unter Umständen zuerst diesen IPv6-Resolver ab – und erhält für `web01.example.test` ein externes `NXDOMAIN`, obwohl `192.0.2.10` unter IPv4 korrekt eingetragen ist. Prüfe bei Dual-Stack-Clients daher immer `Get-DnsClientServerAddress` für **IPv4 und IPv6**.

DNS-Konfiguration gegen eine Baseline vergleichen

Gerade bei mehreren Administratoren hilft ein schneller Soll-Ist-Abgleich gegen die beim Abschluss erstellte Konfigurationsreferenz. So erkennst du sofort, ob seit der Inbetriebnahme Zonen hinzugekommen sind, `SecureSecondaries` aufgeweicht wurde oder Forwarder verändert wurden:

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
# Aktuelle Kernkonfiguration kompakt als Prüfbericht ausgeben
[PSCustomObject]@{
    ServerName    = $env:COMPUTERNAME
    ServiceStatus = (Get-Service -Name DNS).Status
    Forwarders    = ((Get-DnsServerForwarder).IPAddress -join ', ')
    UseRootHint   = (Get-DnsServerForwarder).UseRootHint
    Recursion     = (Get-DnsServerRecursion).Enable
    Zones         = ((Get-DnsServerZone | Where-Object { -not $_.IsAutoCreated } |
        ForEach-Object { "$($_.ZoneName) ($($_.ZoneType)/$($_.SecureSecondaries))" }) -
join '; ')
} | Format-List
```

Speichere diesen Zustand nach jeder gewollten Änderung zusammen mit deinem Backup-Ordner ab. Bei einer Störung siehst du innerhalb weniger Sekunden, ob die Laufzeitkonfiguration von deinem dokumentierten Soll-Zustand abweicht.

Schematische Beispielausgabe des Prüfberichts (auf dem Zielsystem nicht live ausgeführt):

```
ServerName    : WIN-CORE01
ServiceStatus : Running
Forwarders    : 192.0.2.1, 192.0.2.2
UseRootHint   : False
Recursion     : True
Zones         : 2.0.192.in-addr.arpa (Primary/NoTransfer); example.test
(Primary/NoTransfer)
```

Für ein automatisiertes Monitoring (etwa über eine geplante Aufgabe oder einen Monitoring-Agenten) lässt sich diese Prüfung auch als kompakter Health-Check mit sauberem Exit-Code umsetzen:

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

```
# Automatisierter DNS-Health-Check mit Exit-Code 0 (OK) oder 1 (Fehler)
try {
    $svc = Get-Service -Name DNS -ErrorAction Stop
    if ($svc.Status -ne 'Running') { throw "DNS-Dienst läuft nicht." }
    $null = Resolve-DnsName -Name "web01.example.test" -Type A -Server 192.0.2.10 -DnsOnly
-NoHostsFile -ErrorAction Stop
    $null = Resolve-DnsName -Name "192.0.2.25" -Type PTR -Server 192.0.2.10 -DnsOnly -
NoHostsFile -ErrorAction Stop
    Write-Output "OK: DNS-Dienst und Zonenauflösung auf WIN-CORE01 funktionsfähig."
    exit 0
} catch {
    Write-Error "FEHLER: $($_.Exception.Message)"
    exit 1
}
```

Befehlsreferenz (Cheatsheet)

Die folgende Übersicht bündelt alle im Artikel verwendeten PowerShell-Befehle nach Aufgabenbereich. Jedes Kommando bezieht sich auf die Workgroup-Referenzumgebung `WIN-CORE01` (`192.0.2.10`, Zone `example.test`) und lässt sich direkt per SSH in einer administrativen PowerShell-Sitzung ausführen.

DNS-Serverrolle

Rollenstatus prüfen Zeigt an, ob die DNS-Serverrolle und Verwaltungswerkzeuge auf Windows Server 2025 Core installiert sind

$ `Get-WindowsFeature -Name DNS, RSAT-DNS-Server`

DNS-Rolle inklusive Tools installieren Installiert die DNS-Serverrolle samt PowerShell-Modul DnsServer ohne AD-Abhängigkeit

$ `Install-WindowsFeature -Name DNS -IncludeManagementTools`

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

DnsServer-Modul und Cmdlets prüfen Prüft die Verfügbarkeit des PowerShell-Moduls für die DNS-Verwaltung

$ `Get-Module -ListAvailable -Name DnsServer`

Dienstverwaltung

Dienststatus und Starttyp prüfen Kontrolliert, ob der Dienst DNS läuft und automatisch mit Windows startet

$ `Get-Service -Name DNS | Select-Object Name, Status, StartType`

DNS-Dienst sauber neu starten Startet den DNS-Dienst kontrolliert neu (leert dabei auch den Server-Cache)

$ `Restart-Service -Name DNS`

Offene Port-53-Sockets prüfen Listet alle aktiven UDP- und TCP-Listener auf Port 53 auf

$ `Get-NetUDPEndpoint -LocalPort 53; Get-NetTCPConnection -LocalPort 53 -State Listen`

Sprachunabhängige Firewall-Prüfung für Port 53 Zeigt alle Firewall-Regeln an, die auf Port 53 wirken, unabhängig von der OS-Sprache

$ `Get-NetFirewallPortFilter | Where-Object { $_.LocalPort -eq 53 } | Get-NetFirewallRule | Select-Object DisplayName, Enabled, Profile, Action`

Firewall-Regel auf Client-Subnetz eingrenzen Erlaubt Port 53/UDP eingehend ausschließlich aus 192.0.2.0/24 im Profil Private

$ `New-NetFirewallRule -DisplayName "DNS-Server (UDP-In, Lab-Subnetz)" -Direction Inbound -Protocol UDP -LocalPort 53 -RemoteAddress "192.0.2.0/24" -Profile Private -Action Allow`

DNS-Zonen

Alle DNS-Zonen auflisten Zeigt Zonennamen, Zonentyp, Pausenzustand und Zonendatei auf dem Server

```
$ Get-DnsServerZone | Format-Table ZoneName, ZoneType, IsPaused, IsShutdown, ZoneFile
```

Dateibasierte Forward-Zone anlegen Erstellt die primäre Forward-Lookupzone example.test ohne dynamische Updates

```
$ Add-DnsServerPrimaryZone -Name "example.test" -ZoneFile "example.test.dns" -DynamicUpdate "None" -PassThru
```

Dateibasierte Reverse-Zone anlegen Erstellt die primäre IPv4-Reverse-Lookupzone für 192.0.2.0/24

```
$ Add-DnsServerPrimaryZone -NetworkID "192.0.2.0/24" -ZoneFile "2.0.192.in-addr.arpa.dns" -DynamicUpdate "None" -PassThru
```

Sekundäre Zone auf zweitem Server anlegen Legt auf einem zweiten Host eine schreibgeschützte Kopie von example.test an

```
$ Add-DnsServerSecondaryZone -Name "example.test" -ZoneFile "example.test.dns" -MasterServers "192.0.2.10" -PassThru
```

Zonendatei sofort auf Platte schreiben Erzwingt das sofortige Zurückschreiben des RAM-Zonenpuffers in die .dns-Datei

```
$ dnscmd.exe /ZoneWriteBack example.test
```

Zonentransfers vollständig sperren Unterbindet AXFR/IXFR-Transfers für eine primäre Zone im Einzelserver-Betrieb

```
$ Set-DnsServerPrimaryZone -Name "example.test" -SecureSecondaries "NoTransfer" -PassThru
```

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Resource Records

Alle Records einer Zone anzeigen Listet alle Resource Records inklusive Typ, TTL und Zieldaten auf

```
$ Get-DnsServerResourceRecord -ZoneName "example.test" | Format-Table HostName, RecordType, TimeToLive, RecordData
```

A-Record anlegen Erstellt einen IPv4-Host-Eintrag für web01.example.test

```
$ Add-DnsServerResourceRecordA -ZoneName "example.test" -Name "web01" -IPv4Address "192.0.2.25" -TimeToLive 00:05:00 -PassThru
```

AAAA-Record anlegen Erstellt einen IPv6-Host-Eintrag für web01.example.test

```
$ Add-DnsServerResourceRecordAAAA -ZoneName "example.test" -Name "web01" -IPv6Address "2001:db8::20" -TimeToLive 00:05:00 -PassThru
```

CNAME-Alias anlegen Verweist portal.example.test auf den kanonischen Hostnamen web01.example.test

```
$ Add-DnsServerResourceRecordCName -ZoneName "example.test" -Name "portal" -HostNameAlias "web01.example.test" -TimeToLive 00:05:00 -PassThru
```

PTR-Record in Reverse-Zone anlegen Ordnet der IP 192.0.2.25 den kanonischen Hostnamen web01.example.test. zu

```
$ Add-DnsServerResourceRecordPtr -ZoneName "2.0.192.in-addr.arpa" -Name "25" -PtrDomainName "web01.example.test." -TimeToLive 00:05:00 -PassThru
```

TXT-Record anlegen Speichert einen Text-Eintrag zur internen Dokumentation oder Verifikation

```
$ Add-DnsServerResourceRecord -ZoneName "example.test" -Name "_meta" -Txt -DescriptiveText "admindocs-lab-zone" -TimeToLive 01:00:00
```

SRV-Record anlegen Verknüpft einen Dienstnamen inklusive Port mit einem Zielhost

```
$ Add-DnsServerResourceRecord -ZoneName "example.test" -Name "_https._tcp" -Srv -DomainName "web01.example.test." -Port 443 -Priority 10 -Weight 100 -TimeToLive 01:00:00
```

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Resource Record sicher löschen (Dry-Run) Simuliert das Entfernen eines A-Records vor der eigentlichen Ausführung

$ `Remove-DnsServerResourceRecord -ZoneName "example.test" -RRType "A" -Name "web01" -RecordData "192.0.2.25" -WhatIf`

Forwarder und Rekursion

Forwarder und Rekursion prüfen Zeigt konfigurierte Upstream-Resolver sowie den Status der Rekursion an

$ `Get-DnsServerForwarder; Get-DnsServerRecursion`

Forwarder ohne Root-Hints setzen Leitet externe Anfragen gezielt an 192.0.2.1 weiter und deaktiviert Root-Hints als Fallback

$ `Set-DnsServerForwarder -IPAddress "192.0.2.1" -Timeout 3 -UseRootHint $false -PassThru`

Bedingte Weiterleitung (Conditional Forwarder) anlegen Delegiert Abfragen für lab.example.test gezielt an einen anderen internen DNS-Server

$ `Add-DnsServerConditionalForwarderZone -Name "lab.example.test" -MasterServers 192.0.2.50 -ForwarderTimeout 3 -PassThru`

Rekursion explizit deaktivieren Schaltet rekursive Weiterleitung auf rein autoritativen DNS-Servern ab

$ `Set-DnsServerRecursion -Enable $false -PassThru`

DNS-Abfragen

Autoritative Forward-Abfrage ohne Cache/Hosts Fragt web01.example.test direkt per Netzwerk gegen 192.0.2.10 ab

$ `Resolve-DnsName -Name "web01.example.test" -Type A -Server 192.0.2.10 -DnsOnly -NoHostsFile`

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

Reverse-Abfrage für IPv4 testen Prüft den PTR-Eintrag für 192.0.2.25 direkt gegen WIN-CORE01

```
$ Resolve-DnsName -Name "192.0.2.25" -Type PTR -Server 192.0.2.10 -DnsOnly -NoHostsFile
```

TCP-Fallback gezielt erzwingen Verifiziert, ob Port 53/TCP für große DNS-Antworten sauber erreichbar ist

```
$ Resolve-DnsName -Name "web01.example.test" -Server 192.0.2.10 -DnsOnly -TcpOnly
```

Klassische Diagnose mit nslookupFührt eine gezielte Abfrage gegen 192.0.2.10 auf Systemen ohne DnsClient-Modul aus

```
$ nslookup web01.example.test 192.0.2.10
```

Troubleshooting, Logs und Wartung

Netzwerk- und Port-53-Check vom Client Prüft IP-Erreichbarkeit und TCP-Verbindungsaufbau zu Port 53 auf WIN-CORE01

```
$ Test-NetConnection -ComputerName 192.0.2.10 -Port 53
```

DNS-Server-Ereignislog auslesen Zeigt die letzten 20 Einträge aus dem operativen DNS-Server-Protokoll

```
$ Get-WinEvent -LogName "DNS Server" -MaxEvents 20 | Select-Object TimeCreated, Id, LevelDisplayName, Message | Format-List
```

DNS-Audit-Log der letzten 24 Stunden Listet Konfigurations- und Record-Änderungen aus dem Audit-Kanal auf

```
$ Get-WinEvent -FilterHashtable @{ LogName = 'Microsoft-Windows-DNSServer/Audit'; StartTime = (Get-Date).AddHours(-24) } -MaxEvents 20 | Select-Object TimeCreated, Id, Message
```

DNS-Server-Cache prüfen und leeren Kontrolliert aktuelle Cache-Parameter und leert bei Bedarf den Server-Cache

```
$ Get-DnsServerCache; Clear-DnsServerCache -Force
```

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten

DNS-Server inklusive Forwarder selbst testen Führt eine integrierte Funktionsprüfung des DNS-Servers durch

```
$ Test-DnsServer -IPAddress "192.0.2.10" -Context Forwarder
```

Temporäres Paket-Debug-Logging aktivieren Schreibt DNS-Anfragen und Antworten kurzzeitig zur Fehlersuche in eine Logdatei

```
$ Set-DnsServerDiagnostics -Queries $true -Answers $true -EnableLoggingToFile $true -LogFilePath "C:\Logs\dns-debug.log"
```

DNS-Registry-Zonenkonfiguration exportieren Sichert die Zonen-Metadaten aus der Windows-Registry als .reg-Datei

```
$ reg.exe export "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\DNS Server\Zones" "C:\Backup\dns-zones.reg" /y
```

Weiterführende Ressourcen

Die folgenden Primärquellen vertiefen die eingesetzten DNS-Konzepte, PowerShell-Module und den zugrunde liegenden Server-Core-Aufbau:

Windows Server 2025 Core: Installation, SSH und Verwaltung Grundlagenartikel zu SConfig, statischem Netzwerk, OpenSSH und der Basiskonfiguration von WIN-CORE01

DNS-Übersicht in Windows Server (Microsoft Learn) Offizielle Architektur- und Funktionsübersicht zur DNS-Serverrolle unter Windows Server

Verwalten von DNS-Zonen (Microsoft Learn) Dokumentation zu primären, sekundären und Stub-Zonen sowie Zonentypen

Installieren und Konfigurieren eines DNS-Servers (Microsoft Learn) Microsoft-Leitfaden für die Installation der DNS-Rolle und Basiseinstellungen

PowerShell-Modul DnsServer (Microsoft Learn) Vollständige Cmdlet-Referenz für Zonen, Resource Records, Forwarder, Cache und Diagnose

Set-DnsServerPrimaryZone (Microsoft Learn) Parameterreferenz zur Absicherung von Zonentransfers (-SecureSecondaries) und dynamischen Updates

Resolve-DnsName (Microsoft Learn) Referenz für gezielte DNS-Abfragen per PowerShell inklusive -DnsOnly, -NoHostsFile und -TcpOnly

Quelle: https://admindocs.de/de/windows-server/windows-server-2025-core-dns-server-mit-powershell-einrichten-und-verwalten