Active.Directory.Utility: Installation Guide

Production installation and setup for the deployment engineer. November 2026.

Example names (corp.example.com, api.corp.example.com, devdc01, iis01, svc-adutility) must be replaced with your own. Validate every step in a test environment first.

1. How the pieces fit

```
Clients --HTTPS 443--> API load balancer --HTTPS 443--> IIS nodes (this app)
IIS nodes --WinRM over HTTPS 5986--> AD target server (or DC load-balancer VIP)
AD target server --AD Web Services 9389--> Domain controllers
```

The app is ASP.NET Core on .NET 11, hosted in IIS. It runs the PowerShell ActiveDirectory cmdlets on the AD target server over WinRM/SSL. Each API request opens its own PowerShell session and closes it, so the IIS tier is stateless.

From	To	Port	Purpose
Clients	API load balancer	443	API calls
API load balancer	IIS nodes	443	Re-encrypted HTTPS (do not offload to plain HTTP)
IIS nodes	AD target server / DC VIP	5986	WinRM over SSL
IIS nodes	Domain controllers	636	LDAPS check used by /api/HealthCheck only
AD target server	Domain controllers	9389, 389/636, 88, 135, RPC	Normal AD traffic for the cmdlets

If you run with UseRemotePowerShell=false (local mode), the IIS nodes themselves need RSAT AD PowerShell and direct access to AD Web Services (TCP 9389).

2. Information to collect before you start

- [ ] DNS name clients will use for the API (api.corp.example.com) and the IIS node names.
- [ ] The AD target server name for WinRM (a DC, a management server, or the DC load-balancer VIP name).
- [ ] Service account (domain account or gMSA) and its password if it is not a gMSA.
- [ ] Internal CA name, certificate templates, and the CA root/intermediate certificate files (.cer).
- [ ] Load balancer vendor and who administers it.
- [ ] Approved firewall changes for the ports in section 1.
- [ ] At least one strong API key and a CryptKey (section 8).

3. Accounts and permissions

IIS application pool identity. Use a dedicated domain account or a gMSA (for example CORP\svc-adutility). It needs full rights on the Active Directory domain controller(s) it manages and on the IIS web site folder. Where your policy allows, delegate only the specific AD permissions needed on the OUs involved instead of Domain Admin.

Remote PowerShell account (RemotePowerShellUserName). It must be allowed to connect to the AD target server over WinRM and run the AD cmdlets. It can be the same account.

1. Create the account(s) in AD.
2. On the AD target server, add the remote account to the local group Remote Management Users (or Administrators if your policy requires it).
3. On each IIS node, add the app pool account to the local IIS_IUSRS group.
4. Record the password in your secrets manager. It is stored in appsettings.json as an encrypted value (section 8.1).

4. Certificates

There are three certificates, plus trust on the IIS nodes.

#	Certificate	Installed on	Must contain	Who must trust it
A	IIS web server certificate	Each IIS node (or one shared PFX)	Server Authentication EKU; SAN = API DNS name and node FQDN	Clients and the API load balancer
B	WinRM HTTPS listener certificate	AD target server (each server behind the VIP)	Server Authentication EKU; SAN = exact name in RemoteADServer	The IIS nodes
C	LDAPS certificate	Each domain controller	Server Authentication EKU; DC FQDN (and VIP name if used)	The IIS nodes, for the health check

Use RSA 2048-bit or larger with SHA-256. Do not use self-signed certificates in production.

4.1 Certificate A: IIS web server certificate

On each IIS node create web.inf:

```
[Version]
Signature=$Windows NT$

[NewRequest]
Subject = "CN=api.corp.example.com"
KeySpec = 1
KeyLength = 2048
Exportable = FALSE
MachineKeySet = TRUE
ProviderName = "Microsoft RSA SChannel Cryptographic Provider"
ProviderType = 12
RequestType = PKCS10
KeyUsage = 0xa0
HashAlgorithm = SHA256

[EnhancedKeyUsageExtension]
OID=1.3.6.1.5.5.7.3.1

[Extensions]
2.5.29.17 = "{text}"
_continue_ = "dns=api.corp.example.com&"
_continue_ = "dns=iis01.corp.example.com&"

certreq -new web.inf web.req
certreq -submit -attrib 'CertificateTemplate:WebServer' web.req web.cer
certreq -accept web.cer
Get-ChildItem Cert:\LocalMachine\My | Format-List Subject, Thumbprint, NotAfter, DnsNameList
```

If your CA does not allow the SAN in the request, ask the PKI team for a template that does (for example a copy of Web Server with Supply in the request enabled). Note the thumbprint for section 7. The API load balancer must trust the CA that issued this certificate if it re-encrypts to the IIS nodes.

4.2 Certificate B: WinRM HTTPS listener (AD target server)

The name clients use (RemoteADServer, or the load-balancer VIP name) must appear in the certificate subject or SAN, otherwise you get certificate name errors.

1. Request a certificate on the AD target server as in 4.1, with Server Authentication EKU and SAN dns=devdc01.corp.example.com (add the VIP name if you connect through a load balancer). On a domain controller the built-in Domain Controller Authentication or Kerberos Authentication templates include Server Authentication but only carry the DC's own name, so use a custom template for a VIP name.
2. Find the thumbprint (no spaces, no angle brackets):

```
Get-ChildItem Cert:\LocalMachine\My |
    Where-Object { $_.EnhancedKeyUsageList.FriendlyName -contains 'Server Authentication' } |
    Format-List Subject, Thumbprint, NotAfter, DnsNameList
```

3. Create the HTTPS listener (use the real thumbprint):

```
$thumb = '0123456789ABCDEF0123456789ABCDEF01234567'
New-WSManInstance -ResourceURI winrm/config/Listener `
    -SelectorSet @{Address='*';Transport='HTTPS'} `
    -ValueSet @{Hostname='devdc01.corp.example.com';CertificateThumbprint=$thumb}
New-NetFirewallRule -DisplayName 'WinRM HTTPS 5986' -Direction Inbound `
    -LocalPort 5986 -Protocol TCP -Action Allow
winrm enumerate winrm/config/listener
```

The output must show Transport = HTTPS, Port = 5986 and the correct CertificateThumbprint. The Hostname must match the certificate subject or SAN.

4. Every API request opens a shell. Check and, if many requests can overlap, raise the quota:

```
Get-Item WSMan:\localhost\Shell\MaxShellsPerUser
Set-Item WSMan:\localhost\Shell\MaxShellsPerUser -Value 100
```

4.3 Certificate C: LDAPS on the domain controllers

Domain controllers normally enroll automatically for a Domain Controller Authentication or Kerberos Authentication certificate and use it for LDAPS on port 636. Verify rather than create:

```
Test-NetConnection devdc01.corp.example.com -Port 636
```

If the port is closed or the certificate is missing, ask the AD/PKI team to enable autoenrollment for domain controllers. The app uses this connection only for the /api/HealthCheck TLS test.

4.4 Trust on every IIS node

The IIS nodes must trust the CA that issued certificates B and C. Domain-joined servers usually already trust the enterprise root through Group Policy; confirm with:

```
Import-Certificate -FilePath .\RootCA.cer -CertStoreLocation Cert:\LocalMachine\Root
Import-Certificate -FilePath .\IssuingCA.cer -CertStoreLocation Cert:\LocalMachine\CA
certutil -verify -urlfetch .\devdc01.cer
```

Import into the Local Machine store, not Current User. The certutil check also proves the CRL/OCSP locations are reachable from the IIS node. In production leave SkipCACheck, SkipCNCheck and SkipRevocationCheck false; set SkipRevocationCheck to true only in a lab.

5. Prepare each IIS node

```
Install-WindowsFeature Web-Server, Web-Mgmt-Tools -IncludeManagementTools
```

1. Install the .NET 11 Hosting Bundle (dotnet.microsoft.com) as administrator.
2. Restart IIS: `net stop was /y` then `net start w3svc`.
3. Confirm the module: `Get-WebGlobalModule | Where-Object Name -like 'AspNetCoreModuleV2'`.

If you target .NET 10 instead, install the .NET 10 Hosting Bundle.

6. Build and copy the app

```
dotnet publish -c Release -r win-x64 --self-contained false -o .\publish
```

Copy the publish folder contents to C:\inetpub\Active.Directory.Utility on every IIS node, then grant the app pool account rights (Modify is enough; use Full control if your standard requires it):

```
icacls 'C:\inetpub\Active.Directory.Utility' /grant 'CORP\svc-adutility:(OI)(CI)M'
New-Item -ItemType Directory -Force 'C:\inetpub\Active.Directory.Utility\Logs'
```

7. IIS application pool and site

The application pool must target .NET CLR version: No Managed Code.

```
Import-Module WebAdministration
New-WebAppPool -Name 'ADUtility'
Set-ItemProperty 'IIS:\AppPools\ADUtility' -Name managedRuntimeVersion -Value ''
Set-ItemProperty 'IIS:\AppPools\ADUtility' -Name startMode -Value 'AlwaysRunning'
```

```
Set-ItemProperty 'IIS:\AppPools\ADUtility' -Name processModel.idleTimeout -Value ([TimeSpan]::Zero)
Set-ItemProperty 'IIS:\AppPools\ADUtility' -Name processModel.loadUserProfile -Value $true
Set-ItemProperty 'IIS:\AppPools\ADUtility' -Name processModel.identityType -Value 'SpecificUser'
Set-ItemProperty 'IIS:\AppPools\ADUtility' -Name processModel.userName -Value 'CORP\svc-adutility'
Set-ItemProperty 'IIS:\AppPools\ADUtility' -Name processModel.password -Value 'THE-PASSWORD'
```

For a gMSA, set the user name to CORP\gmsa-adutility$ and the password to an empty string. Load User Profile must be True because the PowerShell SDK needs a user profile.

Create the HTTPS-only site and attach certificate A:

```
New-Website -Name 'ADUtility' -PhysicalPath 'C:\inetpub\Active.Directory.Utility' `
    -ApplicationPool 'ADUtility' -Port 443 -Ssl
$thumb = 'THUMBPRINT-OF-CERTIFICATE-A'
(Get-WebBinding -Name 'ADUtility' -Protocol https).AddSslCertificate($thumb, 'My')
```

In IIS Manager open the site, choose SSL Settings and tick Require SSL. Do not add an HTTP binding. Allow only TLS 1.2 and 1.3 on the server.

8. Application settings

All settings are in appsettings.json in the site folder. Changes take effect after the app pool is recycled.

Setting	Production value
ShowSwaggerAPI_Documentation	false (or true only on an internal network)
ForceHttpsRedirection / HttpsPort	true / 443
FORCE_REQUIRED_HEADER_SECURITY_KEY_XApiKey	true
ApiKeys	one or more strong keys, each stored as an ENC2: value (section 8.1)
LogToFile / LogFileNamePath	true / absolute path, e.g. D:\Logs\ADUtility\ADUtility-.log (app pool account needs Modify)
UseRemotePowerShell	true
RemoteADServer	FQDN of the AD target server or VIP, matching certificate B
ADDomainController	optional: one specific DC (ideally the PDC emulator) so reads follow writes
RemotePowerShellUserName	the remote account, e.g. CORP\svc-adutility
RemotePowerShellUserPassword	the ENC2: value from the Encrypt method (section 8.1)
UseSSL_4WinRM / PortNr_4WinRM	true / 5986
SkipCACheck, SkipCNCheck, SkipRevocationCheck	all false
TrustForwardedHeaders / KnownProxies	true, and the load balancer IPs, when behind a load balancer
LoadBalancer.Active	the vendor key in use (F5_BIG_IP, Foundry_Brocade_ADX, NGINX_Plus, AWS_ELB, Cloudflare, Citrix_NetScaler, HAProxy_Enterprise) or None
CryptKey	the key used to encrypt the values above, at least 16 characters
EnableCryptoEndpoints / CryptoEndpointsLocalOnly	false after the values are encrypted; keep CryptoEndpointsLocalOnly true while enabled

8.1 Encrypt the secrets

RemotePowerShellUserPassword and every entry in ApiKeys are stored as encrypted text starting with ENC2:. The app decrypts them once at startup using CryptKey. A plain value still works, so you can migrate one setting at a time.

Encryption is AES-256-GCM with a key derived from CryptKey (PBKDF2-SHA256, 600,000 iterations, random salt and nonce per value). A wrong key or an altered value fails to decrypt. The result is not tied to one machine, so one encrypted value works on every IIS node.

Protect CryptKey. It is saved in appsettings.json. Anyone who can read it with the encrypted values can decrypt them, so limit that file to administrators and the app pool account. For better separation set CryptKey as an app pool

environment variable named ApplicationSettings__CryptKey.

1. Generate a CryptKey of at least 16 characters and store it in your secrets manager:

```
$b = New-Object byte[] 32
[Security.Cryptography.RandomNumberGenerator]::Create().GetBytes($b)
[Convert]::ToBase64String($b)
```

2. Deploy with the plain password and API keys first, CryptKey empty, EnableCryptoEndpoints true and CryptoEndpointsLocalOnly true.
3. Sign in to the IIS node and call the Encrypt method once for the password and once for each API key. Use the node's own name, not the load balancer: by default the call is refused (HTTP 403) unless it comes from the server itself.

```
$h = @{ 'X-Api-Key' = 'CURRENT-PLAIN-API-KEY' }
$body = @{ Text2Encrypt = 'THE-PASSWORD'; CryptKey = 'YOUR-CRYPT-KEY' } | ConvertTo-Json
(Invoke-RestMethod -Method Post -Uri 'https://iis01.corp.example.com/api/Encrypt' `
    -Headers $h -ContentType 'application/json' -Body $body).Data.Result
```

4. Paste each result into appsettings.json and set CryptKey:

```
"CryptKey": "YOUR-CRYPT-KEY",
"ApiKeys": [ "ENC2:...", "ENC2:..." ],
"RemotePowerShellUserPassword": "ENC2:..."
```

5. Recycle the app pool (Restart-WebAppPool ADUtility), then call /api/HealthCheck. Clients keep sending their normal plain API key.
6. Set EnableCryptoEndpoints to false and recycle again, so the Encrypt and Decrypt methods no longer exist on the server.

To check a stored value, call Decrypt the same way:

```
$body = @{ Text2Decrypt = 'ENC2:...'; CryptKey = 'YOUR-CRYPT-KEY' } | ConvertTo-Json
(Invoke-RestMethod -Method Post -Uri 'https://iis01.corp.example.com/api/Decrypt' `
    -Headers $h -ContentType 'application/json' -Body $body).Data.Result
```

Both methods take a JSON body and answer with JSON (Success, then Data.Result, or Error with HTTP 400 for a missing field, a CryptKey shorter than 16 characters, or a wrong key). They are hidden from Swagger and need the X-Api-Key header. The log records Text2Encrypt, Text2Decrypt and CryptKey masked as ****. If the app cannot decrypt a value at startup it does not start, and the Windows Event Log shows the reason. To change CryptKey, set EnableCryptoEndpoints back to true, decrypt each value with the old key, encrypt it with the new key, replace the values, set the new CryptKey and recycle the pool.

9. Load balancers

Pool	Port	Persistence	Health monitor
API to IIS nodes	443	None needed (stateless)	HTTPS GET /api/HealthCheck, expect 200
WinRM to AD target / DCs	5986	Source-IP persistence, or L4 TCP with connection reuse; timeout of at least 5 minutes	TCP 5986
AD Web Services (if balanced)	9389	Same persistence as WinRM is safest	TCP 9389

A WinRM session is created, used and deleted on one specific server, so those steps must reach the same backend. HealthCheck is exempt from the API key by default so monitors can call it. Supported load balancers: F5 BIG-IP, Foundry/Brocade ADX, NGINX Plus, AWS Elastic Load Balancing, Cloudflare, Citrix NetScaler (ADC) and HAProxy Enterprise. Terminate TLS at the load balancer only if it re-encrypts to IIS.

10. Smoke tests

From an IIS node, test each layer:

```
Test-NetConnection devdc01.corp.example.com -Port 5986
Test-WSMan devdc01.corp.example.com -UseSSL -Authentication Negotiate
$c = Get-Credential
Invoke-Command -ComputerName devdc01.corp.example.com -UseSSL -Credential $c `
    -ScriptBlock { Import-Module ActiveDirectory; (Get-ADDomain).DNSRoot }
```

Then from a client machine:

```
curl.exe https://api.corp.example.com/api/HealthCheck
curl.exe -H 'X-Api-Key: THE-API-KEY' `
    'https://api.corp.example.com/api/GetOUsInOU?LDAP_Path=LDAP://DC=corp,DC=example,DC=com'
curl.exe 'https://api.corp.example.com/api/GetOUsInOU?LDAP_Path=LDAP://DC=corp,DC=example,DC=com'
```

Expected: HealthCheck returns 200 with every check Ok; the second call returns JSON with Success true; the third (no key) returns 401. Then open today's log file and confirm each call was logged with time, IP, user, method and parameters, and that passwords show as ****.

11. Security checklist

- [ ] Sample API key replaced; keys stored only in the secrets manager.
- [ ] ApiKeys and RemotePowerShellUserPassword are ENC2: values; appsettings.json readable only by administrators and the app pool account; EnableCryptoEndpoints is false.
- [ ] Swagger off, or reachable only from an internal network.
- [ ] No HTTP binding; Require SSL ticked; TLS 1.0 and 1.1 disabled.
- [ ] Skip-certificate-check settings are all false.
- [ ] App pool account has only the AD rights it needs.
- [ ] NTFS permissions: site folder limited to administrators and the app pool account; log folder writable by the app pool account only.
- [ ] Firewall rules limited to the ports in section 1.
- [ ] Load balancer and IIS logs contain no passwords (password endpoints use JSON bodies, not URLs).
- [ ] Certificate expiry dates for A, B and C are in your monitoring calendar.

12. Operations

- **Logs:** the folder set in LogFileNamePath, one file per day, 10 MB limit, 31 files kept by default. Each node writes its own logs.
- **Deploying an update:** place app_offline.htm in the site folder, copy the new files, delete app_offline.htm, then check HealthCheck. Do nodes one at a time behind the load balancer.
- **Renewing certificate B:** import the new certificate, delete the HTTPS listener (Remove-WSManInstance), recreate it with the new thumbprint (section 4.2), and test with Test-WSMan -UseSSL.
- **Renewing certificate A:** import it and run AddSslCertificate again with the new thumbprint.
- **Rotating an API key:** encrypt the new key with the Encrypt method (section 8.1), add it as a second entry in ApiKeys, recycle, switch clients over, then remove the old entry.

13. Troubleshooting

Symptom	Likely cause and fix
Certificate signed by an unknown certificate authority	CA not trusted on the IIS node. Import the root into Cert:\LocalMachine\Root (4.4).
Certificate could not be checked for revocation	The CRL/OCSP URL is unreachable from the IIS node. Open access, or lab only: SkipRevocationCheck = true.
Certificate name mismatch	RemoteADServer does not match the certificate subject or SAN. Use the exact FQDN on the certificate.
401 on every call	Missing or wrong X-Api-Key header, or the ApiKeys setting is empty.
503 Unable to reach the PowerShell remoting endpoint	Firewall, listener not created, wrong port, or WinRM load balancer not routing 5986. Run the section 10 tests.
500: ActiveDirectory module could not be loaded	The AD target server lacks RSAT AD PowerShell or the remote account cannot load it.
504	The request exceeded ServerTimeLimit. Narrow the search or raise the limit.
500.30 or 502.5 at startup	Hosting Bundle missing, wrong .NET version, or a bad appsettings.json. Check the Windows Event Log.
App does not start after encrypting (500.30)	CryptKey is empty or is not the key used to encrypt. The Event Log (Application) names the value that could not be decrypted.
403 from /api/Encrypt or /api/Decrypt	The call did not come from the server itself. Run it on the IIS node using the node's own name.
Access denied from AD	The remote account lacks AD rights on that OU, or is not in Remote Management Users on the target.
Works on one node, fails on another	Certificate trust, environment variables, or appsettings.json differ between nodes.

14. Sign-off

- [ ] Certificates A, B and C installed and trusted
- [ ] WinRM HTTPS listener verified from every IIS node
- [ ] App pool No Managed Code, correct identity, Load User Profile true
- [ ] Production settings applied and secrets encrypted
- [ ] Load balancer pools, health monitors and persistence configured
- [ ] Smoke tests passed on every node
- [ ] Security checklist completed