Timeline of Active Directory. Still working though all other ActiveX tools now shut down

Active Directory (AD) is still actively maintained and widely used, despite the unrelated deprecation and shutdown of ActiveX tools*. ** *Although they share the word "Active" and were both introduced by Microsoft around the late 1990s, Active Directory is an enterprise identity and network management service, whereas ActiveX was a legacy browser and document plugin framework.*

Timeline of Active Directory

Microsoft has continuously updated Active Directory since its debut, evolving it from a strictly on-premises network directory into a hybrid identity backbone.

- **1999 – 2000 (Launch).** Microsoft previewed Active Directory in 1999 and officially released it with Windows 2000 Server. It replaced the older Windows NT domain model by introducing hierarchical organizational units, Group Policy, and native LDAP and Kerberos support.
- **2003 – 2008 (Expansion and Federation).** Windows Server 2003 added domain renaming and improved administration. Windows Server 2008 renamed the core service to Active Directory Domain Services (AD DS) and introduced Active Directory Federation Services (AD FS) for web single sign-on.
- **2012 – 2016 (Virtualization and Cloud Sync).** Windows Server 2012 introduced safer domain controller cloning and simplified deployment. During this era, Microsoft launched Azure Active Directory (now Microsoft Entra ID) alongside sync tools to bridge on-premises AD with cloud services.
- **2019 – 2022 (Stability Focus).** Windows Server 2019 and 2022 kept the forest and domain functional levels at the 2016 baseline. Microsoft focused on security hardening, hybrid integration, and encouraging cloud adoption rather than changing the core AD database engine.
- **2024 – 2026 (Modernization in Windows Server 2025).** With the release of Windows Server 2025, Microsoft delivered the largest architectural update to AD in nearly a decade. It introduced new 2025 domain and forest functional levels, expanded the database page size from 8 KB to 32 KB for larger objects, and added NUMA-aware scaling for massive enterprise environments.

Why Active Directory Survives While ActiveX Shut Down

The two technologies served entirely different purposes and faced very different security realities.

- **Different Architectures.** ActiveX (introduced in 1996) allowed websites and Office documents to execute native code directly on a user's machine. Active Directory is a backend directory database that manages users, computers, and security policies across corporate networks.
- **ActiveX Retirement.** Because running external code inside browsers and documents created severe malware and phishing risks, Microsoft retired Internet Explorer (the primary host for

web ActiveX) and disabled ActiveX controls by default across Office 2024 and Microsoft 365 apps in 2025.

- **Ongoing Enterprise Reliance on AD.** While many organizations adopt Microsoft Entra ID for cloud apps, Entra does not natively replace legacy on-premises protocols like Kerberos, NTLM, or Group Policy. As a result, most enterprises operate in a hybrid model where on-premises Active Directory remains essential for internal servers and infrastructure.

If you'd like to explore a specific area further, let me know:

- **Upgrading or migrating** an existing Active Directory environment (such as moving to Windows Server 2025 or hybrid Entra ID)
- **Replacing legacy ActiveX workflows** in older web apps or Office documents

I can walk you through the steps and modern alternatives.

.asp (Classic ASP) and .aspx ASP.Net web pages continue to be widely used and supported. Millions of web pages out there.

Yes, both Classic ASP (`.asp`) and ASP.NET Web Forms (`.aspx`) remain actively supported by Microsoft on modern Windows Server operating systems, with millions of legacy business applications still running on them. Unlike client-side ActiveX controls, which posed severe security risks on end-user devices, `.asp` and `.aspx` execute strictly on the web server and send standard HTML to the browser.

Current Support Lifecycle and Status

Microsoft ties the support of both server-side frameworks directly to the underlying Windows Server and .NET Framework lifecycles rather than shutting them down abruptly.

- **Classic ASP (`.asp`).** Introduced in 1996, Classic ASP is still included as an optional feature in Internet Information Services (IIS) on Windows Server 2022 and Windows Server 2025. Microsoft has committed to supporting Classic ASP on Windows Server 2025 through its extended support end date in October 2034.
- **ASP.NET Web Forms (`.aspx`).** Introduced in 2002 as part of the .NET Framework, `.aspx` applications run on .NET Framework 4.8 and 4.8.1. Because these versions are treated as core components of the Windows operating system, they remain supported for as long as the host Windows Server version is supported.
- **No New Features.** While both technologies receive ongoing security patches and bug fixes, Microsoft stopped adding new features to Classic ASP decades ago and placed ASP.NET Web Forms in maintenance mode in favor of modern, cross-platform ASP.NET Core.

Why They Survive While Client-Side ActiveX Died

The contrast between server-side scripts and client-side plugins comes down to where the code runs and how browsers interact with it.

- **Server-Side Execution.** When a user visits an `.asp` or `.aspx` page, the web server processes the VBScript or C# code behind the scenes and outputs standard HTML, CSS, and

JavaScript. Modern browsers like Chrome, Edge, and Safari render these pages effortlessly because they never see the underlying server code.

- **Massive Enterprise Footprint.** Many government agencies, banks, manufacturers, and internal corporate intranets rely on massive `.asp` and `.aspx` applications built over the last twenty years. Rewriting millions of lines of working business logic is expensive and risky, so organizations keep them running on modern Windows servers.
- **COM Objects on the Server.** Interestingly, Classic ASP still uses `Server.CreateObject` to call server-side COM components (which share DNA with ActiveX). Because those components run inside a locked-down data center rather than an end-user's browser, they remain viable for internal business tasks.

If you're maintaining or planning around legacy web apps, let me know if you'd like to explore:

- **Securing and optimizing** `.asp` or `.aspx` applications on modern Windows Server versions
- **Modernizing or migrating** `.aspx` Web Forms to modern .NET (such as Blazor or Razor Pages)

I can share specific best practices or migration strategies.

** End of Page