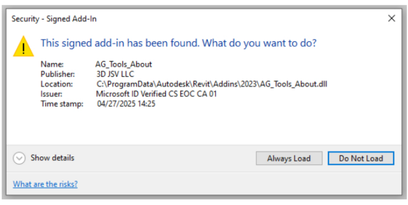
Azure Trusted Signing: a Much Cheaper Alternative to OV Code Signing Certificates

Let's BIM Together
728 followers

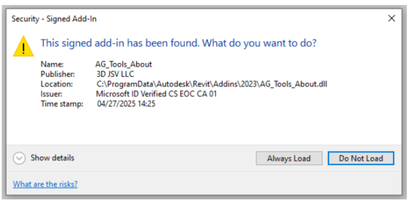
May 30, 2025

Three years ago, I bought my first *OV* code signing certificate for $70. It seemed reasonable at the time. Next year it jumped to $85, which was annoying but manageable. This year? They hit me with $400+ because of some new regulation requiring a physical *USB* (or a similar device)...

For those who haven't dealt with this, *OV* certificates remove the "Unknown Publisher" warning that *Windows* & *Revit* throw at unsigned applications. If you have multiple unsigned Revit plugins, it gets annoying quickly, plus it looks unprofessional for mass internal deployments or anything you want to release to the public. There are also corporate IT environments that outright block unsigned executables, and honestly, for good reasons.



Signed Revit plugin message. Only appears if this is this publisher's first plugin being used.



The unsigned Revit plugin message appears for every plugin that is loaded for the first time.

The price explosion reminded me of a tutorial I used years ago by *Konrad* from **archi-lab.net** about **code signing Revit plugins**. What's funny is that five years after writing that tutorial, in 2022, *Konrad* published another post about **creating self-signed certificates** because the costs were getting expensive (curious how he feels about them now LOL).

Self-signed certificates work great internally and are economically perfect (free!), but they don't help with external distribution. Your certificate needs to be manually added to each user's machine to be recognized, which is a big ask for typical users downloading your plugin. If you're considering this route, check out *Jeremy Tammik's* article on **The Building Coder blog** that covers these technical details.

Since I need a certificate for external public use, just like *Konrad* had his moment with certificate vendors, I found myself doing the same thing. But to my surprise, I think I actually found a good solution: **Azure Trusted Signing**.

Here's what makes it interesting:

- Only $10 per month (compared to $400+ /year for traditional *OV* certificates)
- No long-term commitment (cancel anytime)
- Cloud-based signing (no physical *USB* devices to manage)
- *Microsoft* identity validation means *Microsoft Defender SmartScreen* builds trust instantly

With traditional *OV* certificates, you're paying $400+ annually, deal with a physical *USB* delivery, and you still have to separately submit your plugin to *Microsoft SmartScreen* and ask them to flag your app as safe, which takes several business days.

Azure Trusted Signing costs $10/month with no commitment. *Microsoft* handles the identity validation, it's cloud-based so you can access it from anywhere, and it's integrated with *Microsoft's* trust infrastructure. For my use case, *Azure Trusted Signing* does everything I need. *Microsoft Defender SmartScreen* trusts the certificate instantly, and most importantly for me, *Revit* accepts it without throwing security warnings at users. I wasn't entirely sure it would work for Revit since it's technically not a traditional *OV* or *EV* certificate, but this does work for *Revit*.

The only purpose of having an *OV* certificate is if you need to sign more than 5,000 executables/*DLLs* a month or if you don't have an internet connection during signing. Also, Azure technically issues a new certificate every few days (this does not affect the signing process). Other than that, it seems like just a waste of money. Feel free to drop a comment and educate me on another reason for having an *OV* certificate 😊 I can't think of any more.

Here's how I batch sign all my executables and *DLLs* using *PowerShell*:

- First, create an **Azure** account.
- Download and install **Windows 10 SDK** (I installed everything, but you probably could get away with just installing the **Windows SDK Signing Tools** component). Make sure to add it to your *Environment Variables*.
- Download and install **Trusted Signing Client Tools**. Once installed, note the *Azure.CodeSigning.Dlib.Core.dll* location (you'll need to reference it in your *PowerShell* batch signing script). For me it's at

```
C:\Users\Arthur\AppData\Local\Microsoft\MicrosoftTrustedSigningClientTools
\Azure.CodeSigning.Dlib.Core.dll
```

- Download and install **Azure CLI**.

Once you do all this, fire up *PowerShell* terminal and sign in to *Azure* with this command (replace *TENNANTID* with your *Tenant ID*):

```
az login --tenant TENNANTID
```

Once logged in, you can run the following *PowerShell* script. This example signs everything inside *C:\Users\Arthur\Desktop\Signed_Build*:

```
$files = Get-ChildItem -Path "C:\Users\Arthur\Desktop\Signed_Build" -
Include *.exe, *.dll -Recurse

foreach ($file in $files)
{
    signtool sign /v /fd SHA256 /tr http://timestamp.digicert.com /td
SHA256 `
    /dlib
"C:\Users\Arthur\AppData\Local\Microsoft\MicrosoftTrustedSigningClientTool
s\Azure.CodeSigning.Dlib.dll" `
    /dmdf "C:\Users\Arthur\Desktop\test-sign\metadata.json" `
    $file.FullName
}
```

The ***metadata.json*** file contains your Azure configuration:

```
{
  "Endpoint": "https://eus.codesigning.azure.net/",
  "CodeSigningAccountName": "YourAccountName",
  "CertificateProfileName": "YourCertificateName"
}
```