Code signing on Windows with Azure Trusted Signing

Trash those overpriced third party certs! Set that clumsy dongle on fire! Sign on the line for $9.99. Code signing apps and plugins on Windows in 2024 is finally (more) sane and (same as Apple) cheap.

📝 Before April 2024, this service from Microsoft was in Private Preview and named Azure Code Signing (ACS). It's now named Azure Trusted Signing and is currently available to anyone (with 3 years of business history).

💡 This article walks you through how I set things up. You should visit Microsoft's official docs where they do something similar and also check out Koala DSP's guide.

⚠️ If you are a business, you'll need 3 years of history to use Trusted Signing! Microsoft says "*Trusted Signing at this time can only onboard Legal Business Entities that have verifiable tax history of*

three or more years." I have also heard reports of businesses younger than 3 years passing identity validation, so you could always give it a try.

💡 As of November 2024, trusted signing is now open for individual developers! See this GitHub issue in case you have problems.

Why is code signing needed on Windows at all?

Installers throw up an evil blue SmartScreen warnings on Windows by default. This frightens users and makes them think there's a virus.

Installation is the person's first experience with your product. Adding friction at the start of that experience sucks. Especially for less technical users.

That's a good enough reason for any paid product, in my opinion!

💡 Love pain? Check out my other detailed posts on Windows code signing with third party certs, or code singing and notarization on macOS here.

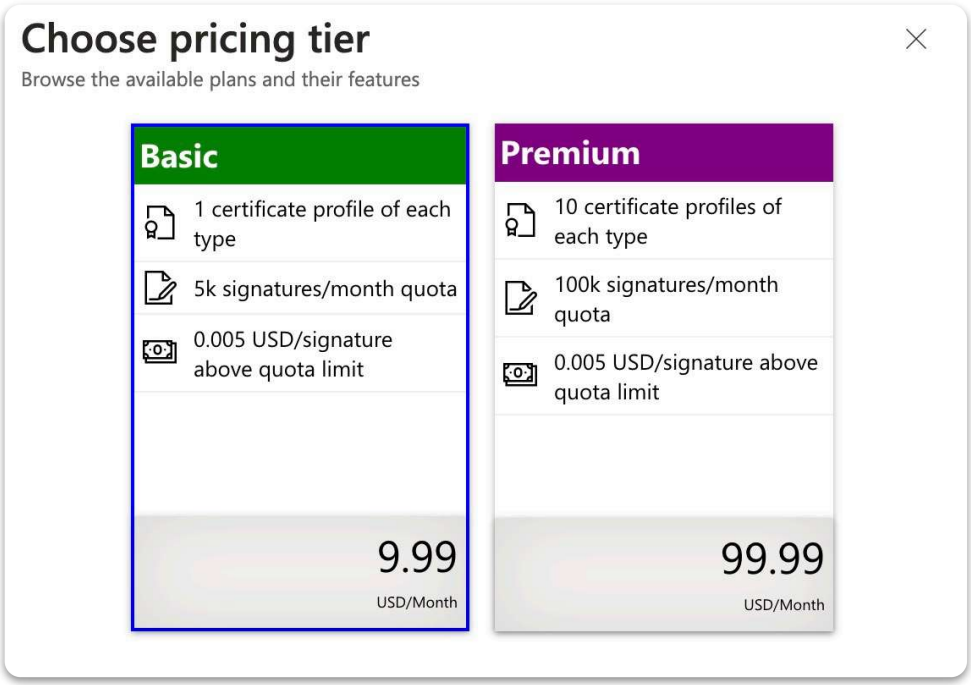
How Azure Trusted Signing works

I've been in the Trusted Signing private preview since late 2023. I've spent an hour chatting to the (very nice!) team one-on-one and have participated in a couple meetings. Here's the scoop:

Instead of buying an overpriced signing certificate from a third party, you'll pay $9.99 a month for a signing account. When you make a new installer, you'll use tools such `signtool` or the official GitHub Action to sign the installer.

Instead of lasting years, certs are now an implementation detail and (generated daily with a lifespan of 3 days). That allows for time-precise revocation if there's any need.

Trusted Signing has been used internally for all of Microsoft's products and close partners for years now. This isn't a "new" service. See this link for detailed compatibility info.



Most of you want the 9.99 option!

Getting started: Create an Azure account

Do it here.

Step 2: Create a Subscription

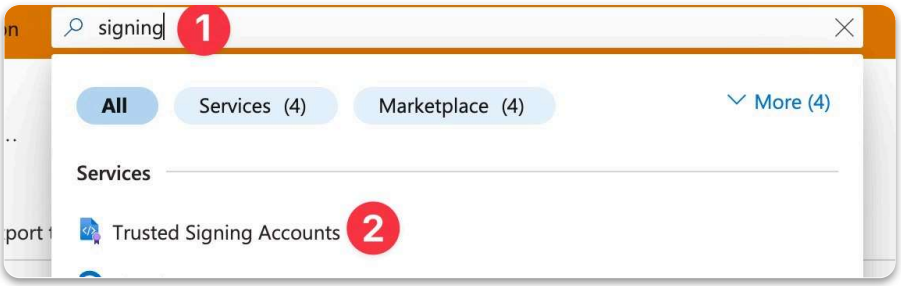
> 💡 According to a commenter, new Azure accounts now come with a Subscription record setup, but you'll have to update it to "pay-as-you-go."

In Azure, you add paid services through creating a "Subscription" record.

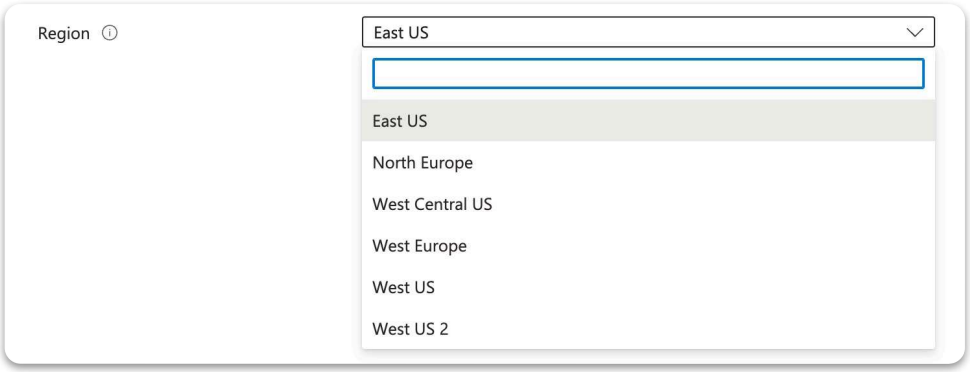
This is sort of a clunky and pointless bureaucratic thing, but hey, it's a pre-req to setting up a code signing account. There's no extra charge for setting up a "subscription."

Step 3: Create a "Trusted Signing Account"

Easiest just to stick `signing` in the search bar than to wade through hundreds of crazy service names.



Select the subscription you just created, pick an arbitrary name and select a region:



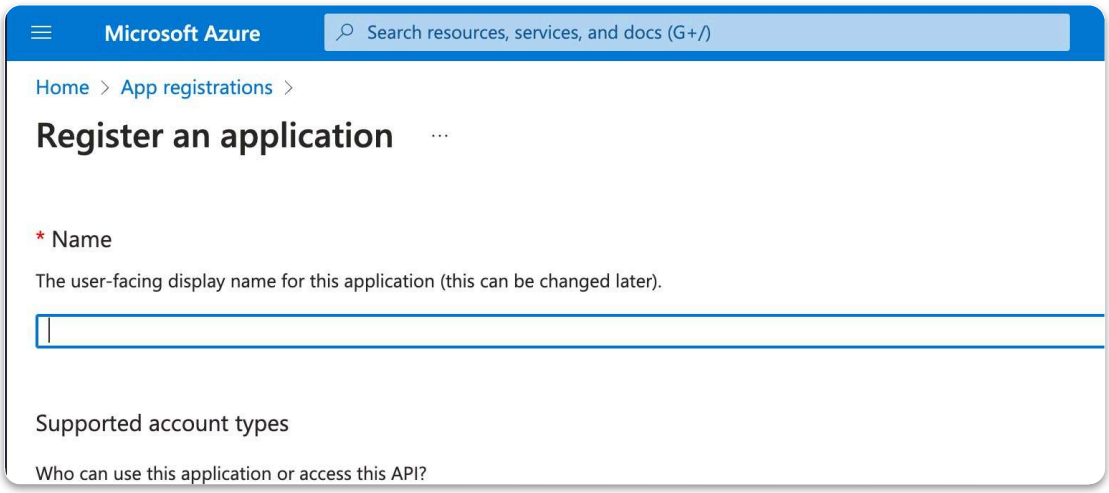
You'll need to specify the region's endpoint when signing. You'll see the url on the main trusted signing account page after creation. You'll need this URL later.

Step 4: Create "App Registration" user credentials

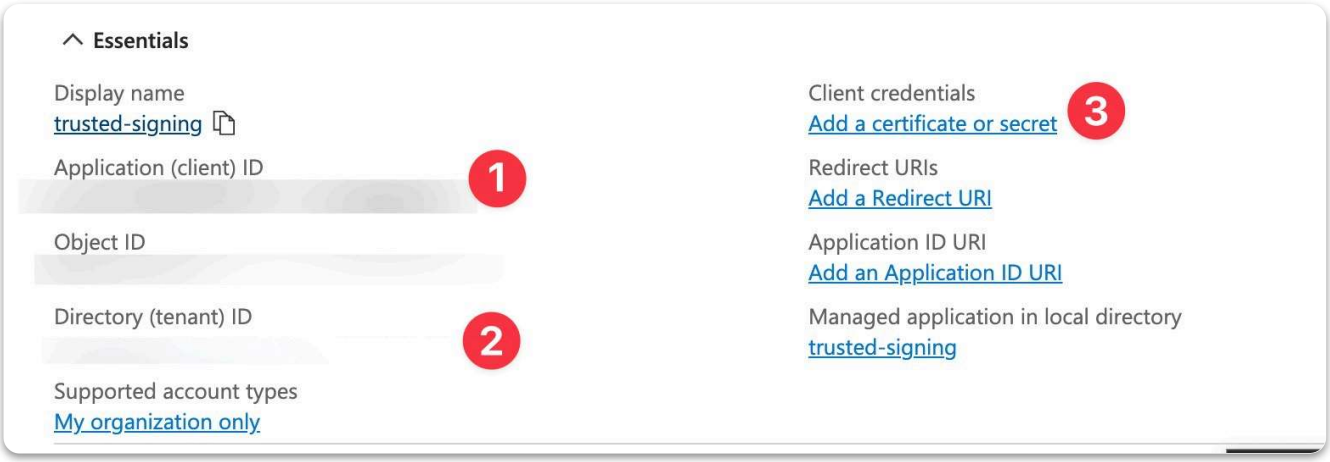
This step creates API credentials for an arbitrary "App Application" to use outside of Azure. In other words, this is how Azure will know it's you when you go to sign

your installers.

Search for `App Registrations` and create a new one.



Give it a name and keep the defaults. I called it `trusted-signing` here, but that's arbitrary.



Note the client ID (1) and the tenant ID (2) for later signing. Locally you will later set these as environment variables `AZURE_CLIENT_ID` and `AZURE_TENANT_ID`.

Then add a secret (3), setting the expiry date to 24 months.

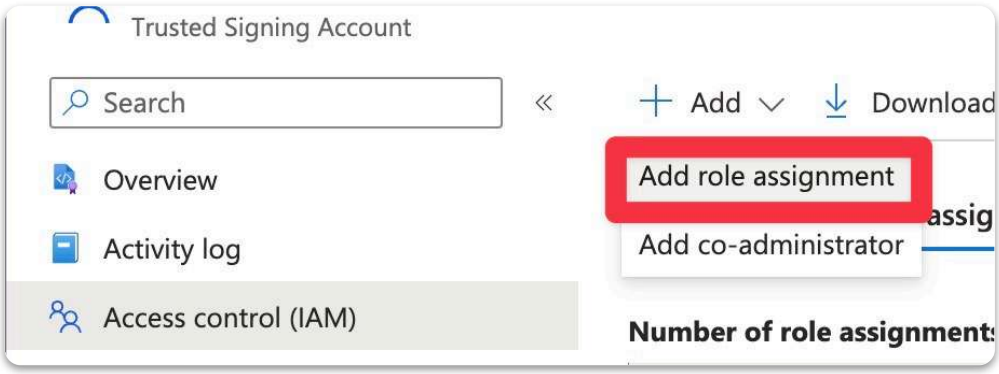
Also note the `secret value` of the created secret. You'll set this as `AZURE_CLIENT_SECRET`.

Step 5: Add "identity verifier" role to your Azure account

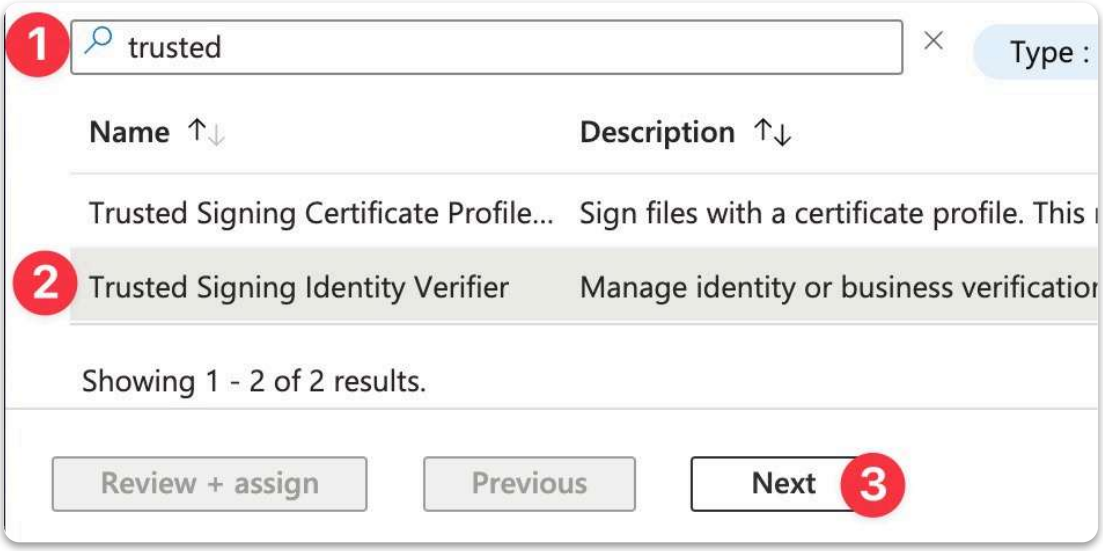
> **You'll go through this role wizard *twice*, once for your Azure user (to add the identity verifier role) and once for that "App Registration" user (to add the signing role).**

First, setup the `Trusted Signing Identity Verifier`. This is so your Azure account has *permission* to go through identity validation. This feels a bit silly and redundant for an indie dev — we're clearly the admin already on our Azure account? But it's necessary.

In the Trusted Signing Account, click `Access Control (IAM)` and then `Add role assignment`.



Search for "trusted" to bring up the role:



Yes, the light gray background means selected!

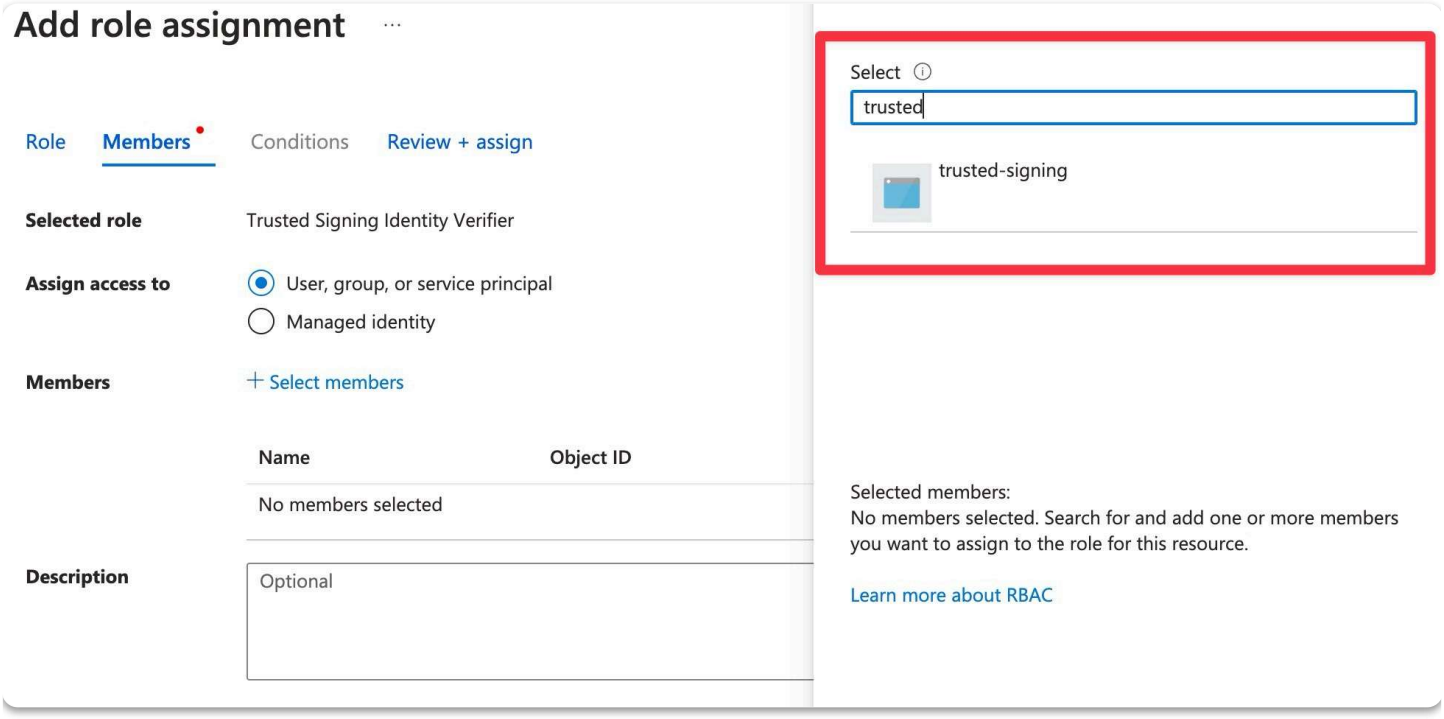
Select the `Trusted Signing Identity Verifier` role and then click through the wizard to add it to your main Azure user.

Step 5b: Add the signer role to your "App Registration" user

Next, you'll start the wizard again. You want to add a role assignment for `Trusted Signing Certificate Profile Signer`. This is a role that we're adding to the `App Registration` user that you created in Step 4. This lets us actually do the signing from the API.

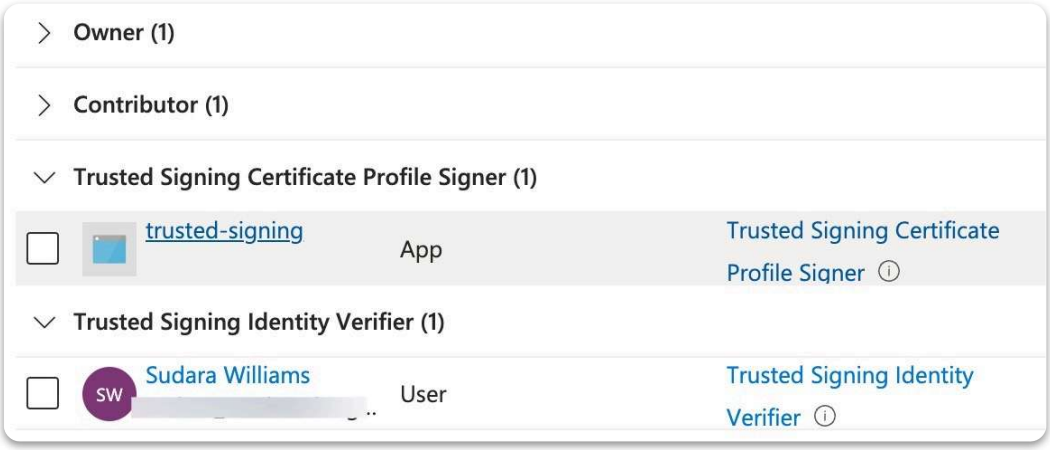
First, do a funny dance of searching for the App Registration user you setup by name. In my case, the App Registration name from Step 4 was `trusted-signing`. So I typed in `trusted` to bring up the user:

> ⚠️ Don't make my mistake of assigning the role to your main Azure user — double check you are assigning the role to the "App Registration" user you created in Step 4.



The UX on all of this is a bit rough!

To double check you did it right, go to `IAM > Role Assignments` and double check the two roles are there:



Again, `trusted-signing` is just my poorly named "App Registration" user created in the previous step!

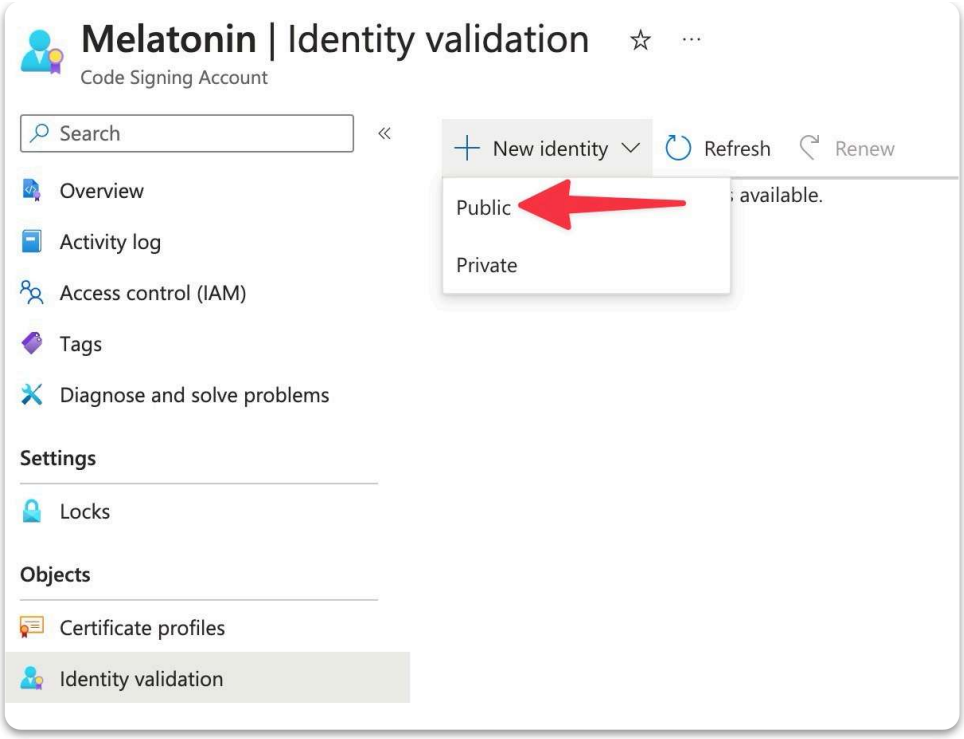
Step 6: Identity Validation

Compared to the old, crusty, third-party identity validations that can take weeks, require phone calls and physical letters, Microsoft's identity validation is *fairly* chill.

Microsoft uses an in-house, worldwide identity validation service. They claim they can validate in as little as an *hour*. This was true for several commenters below as well as a few friends, including one where it took 10 minutes.

For me (in the EU, submitted on a Saturday) it took ~12 hours to get the initial request for additional documents, another ~2 days to get back to me, and then things stalled out a bit because of a misunderstanding (more on this later) taking 10 days in total.

You'll want to select `New Identity > Public`



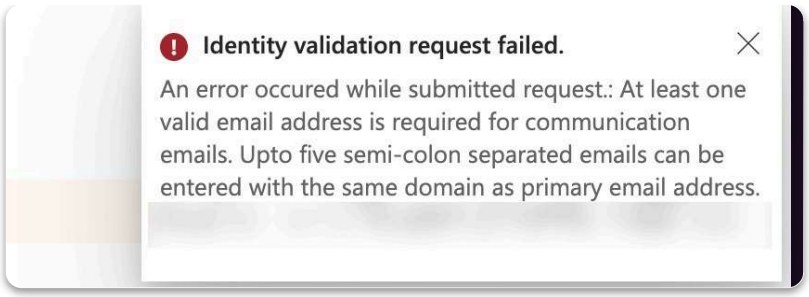
`Private` means "use a certificate chained to an opt-in trust root that your app users have to manually install" — so, yes, you want `Public`!

Now fill out the form. Use a DUNS Number if you are a US biz and have one. Otherwise a Tax ID, for example if you are in the EU with a business (like I am).

> ⚠️ Make sure you can provide proof of ownership of the domain you are submitting as your `Primary Email` (in other words, it can't be @gmail.com or whatever).

> Azure form validation sucks. It took me a few tries before pressing `Create` was possible. Some fields seem to wants numbers only, otherwise it would say things like "This is not a valid tax id."

When I finally could press `Create`, I got hit with this great popup, despite having valid primary and secondary email addresses:



The problem was (randomly) that ***the secondary email address has to be on the same domain as the primary***!

Providing identity documents

You'll probably have to provide additional documents. I got an email 12 hours after I submitted the request. They wanted proof of business and (for some reason I didn't initially understand) proof of domain name ownership?

Azure Code Signing identity validation requires additional documents to complete the request

This email is to notify you that the Azure Code Signing identity validation request for [redacted] requires additional documents for completion. Upload the relevant documents from the categories mentioned below to the Azure portal. The supported file formats are .pdf and .docx. The file size should be less than 5 MB.

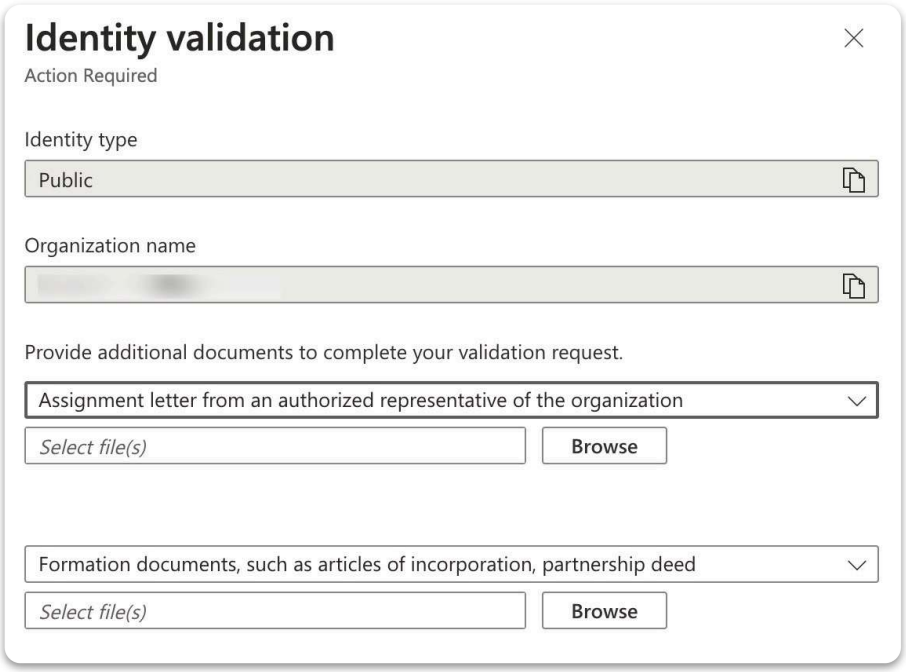
Category 1: (Provide one of the following documents.)

1. Formation documents, such as articles of incorporation, partnership deed
2. Franchise or agency appointment letters
3. Government issued letter, license, registration, or certificate
4. Lease or tenancy documents
5. Letter or statement from a financial institution or a utility company
6. Record on a Government registry website (site/link must be displayed)
7. Stock exchange filings or tax filing records

Category 2: (Provide one of the following documents.)

1. Assignment letter from an authorized representative of the organization
2. Domain ownership records, such as Whois
3. Domain purchase invoices or registry confirmation records
4. Website showing name, address, contact information, and domain of the organization

Logging back into Azure and navigating to `Trusted Signing > Identity Validation`, you'll see the status is `Action Required`. Clicking on the record brings up the document uploader:



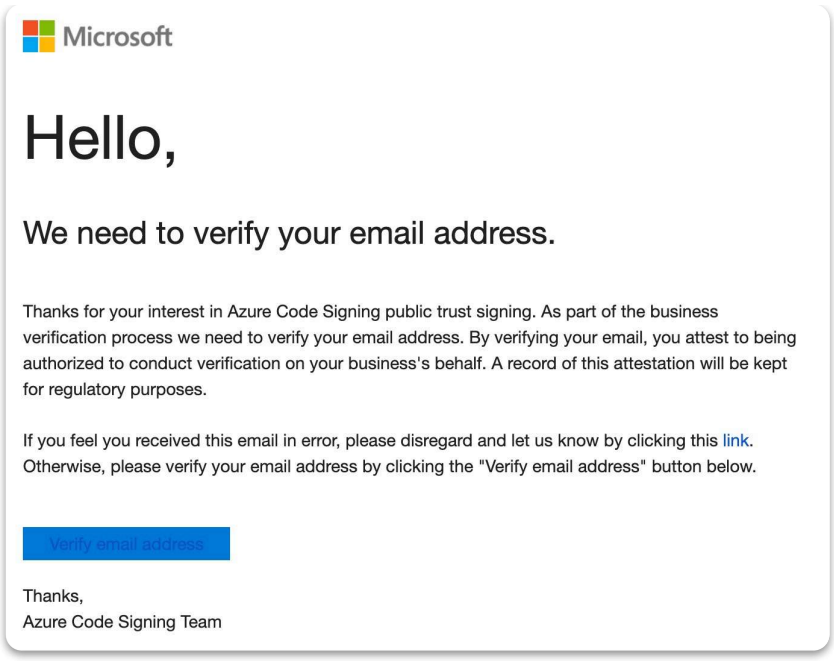
It took 2 days for my documents to be processed. At which point I got another automated request for `Domain purchase invoices or registry confirmation records`.

This was a bit strange. I provided them with a renewal receipt of my domain `melatonin.dev`, so I wasn't sure what else they wanted. I provided them the original purchase receipt and then 2 hours later got another request for the same document.

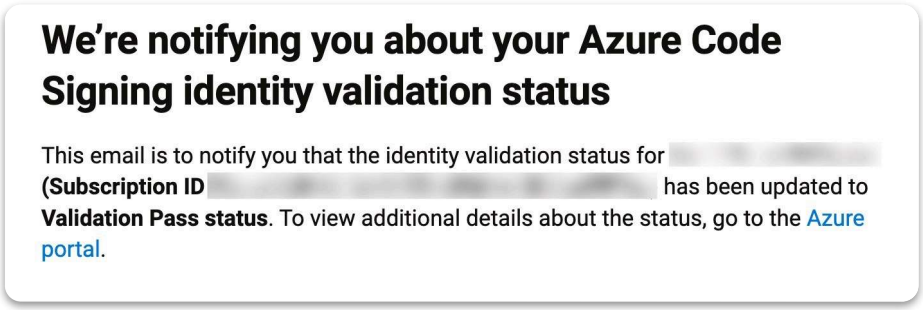
> ⚠️ Do not expect comments from a human as to what the problem is if you have one. There is no unpaid support, you'll have to just guess and wing it.

I figured out they probably wanted proof of ownership of my `Primary Email` domain (which in my case was different than my marketing domain).

About an hour later, I then got an email validation request on my Primary Email:



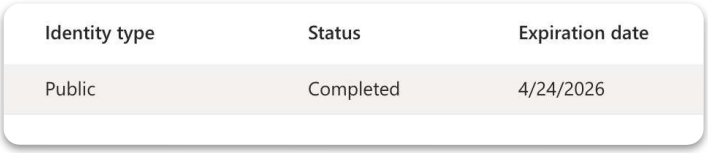
Then things stalled out for me for a few more days. Because I had the luxury of being in the private alpha, I pinged the team to ask what's up. Apparently the internal validation team was confused if I was applying as a business or as an individual. I told them I'm a business — a sole proprietorship — at which point they asked me again for my business license and then approved me. The process took a total of 10 days.



> 💡 If you end up waiting for more than a day or two after submitting documents, re-submit identity validation with a different document, for example the EIN (EU tax ID) document instead of a DUNS (this has worked better for people in the EU). Also, make sure your company name is aligned everywhere you are entering it as well as on your documents. When in doubt, re-submit a new request! Treat it like the API it is.

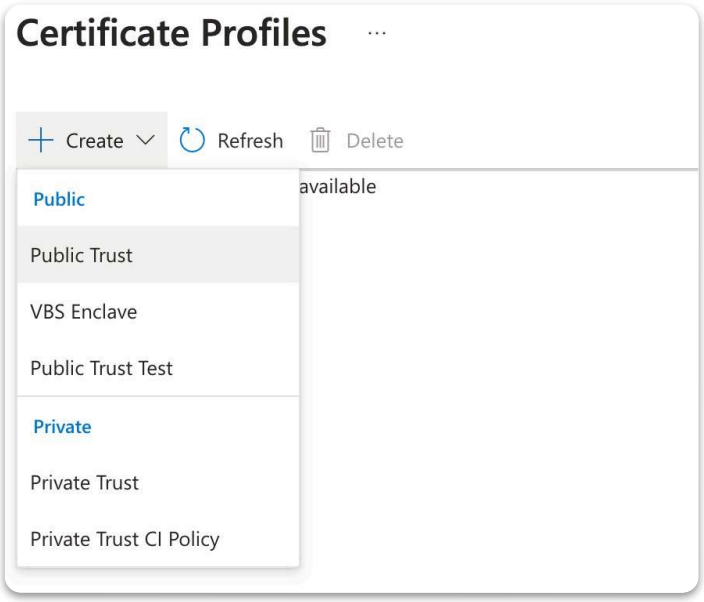
Let me know in the comments how long it took for your identity validation, would be nice to know if I'm an outlier.

Interestingly, the identity validation record expires 2 years after the request was made, better put that on your calendar.



Step 7: Create a Certificate Profile

The actual certs on Azure Trusted Signing are created and rotated daily. But you'll need to create a "profile" to access and sign with them. Create a Public Trust profile:



Pick a name.

You'll need this name later when signing...

Under `Verified CN and O` select your verified identity (from the last step).

Step 8: Signing locally

I initially skipped this step and just got things building on GitHub. I usually only create signed builds via CI (I prefer to keep that boundary hygienic, helps with debugging, etc).

To get going locally, follow Microsoft's docs about how to get started with signtool.exe. They are solid except they conveniently don't really mention authentication at all. Whups. For that, you'll probably just want to export `AZURE_CLIENT_ID`, `AZURE_CLIENT_SECRET` and `AZURE_TENANT_ID` as environment variables to get started.

You can also crib Koala DSP's guide for this part.

I don't have much to add, except:

- You aren't crazy — yes, you need to download a dlib and pass its downloaded location as a command line argument to `signtool`. Not awkward at all. If you don't want to use nuget to grab the dlib, you can click "Download Package" in the sidebar, rename the downloaded file to a `.zip` and bob's your uncle.
- You ***do*** need at least the .NET 6.0 runtime installed. Double check what runtime you have with `dotnet --list-runtimes`.
- You also need to lovingly handcraft some json to feed `signtool` — because we're having fun, ya know? Make sure that endpoint url is right, I fucked that up at first...
- `signtool` credentials give priority to the azure environment variables for "service principles" (aka your application user), but there are many methods including ManagedIdentity. You can also use `az login`. I recommend just setting `AZURE_CLIENT_ID`, `AZURE_CLIENT_SECRET` and `AZURE_TENANT_ID` as environment variables so that Things Just Work.

- You'll need at least version `10.0.2261.755` of `signtool`. If you need a new version locally, you can download it from within Visual Studio's `Feature Search` and it'll show up in `C:\Program Files (x86)\Windows Kits\10\bin`.

Step 9: Trusted Signing in CI (GitHub)

Azure publishes a trusted signing action for GitHub Actions which basically scripts inputs to the Powershell integration.

You'll need 6 pieces of information that we'll add as GitHub secrets.

The first 3 are the application client info (as in local signing): `AZURE_TENANT_ID`, `AZURE_CLIENT_ID` and `AZURE_CLIENT_SECRET`.

In addition you'll need the `AZURE_ENDPOINT` — this the url for the region you selected. You can find this labelled `Account URI` on the main Trusted Signing Account page in Azure. For me, in the EU, it's `https://weu.codesigning.azure.net/`

While you are there, note the name of your trusted signing account. You'll store that as a secret called `AZURE_CODE_SIGNING_NAME`.

Lastly, you'll need the `AZURE_CERT_PROFILE_NAME` from step 7.

> You might notice some GitHub and API stuff is still called “code signing” and not “trusted signing.” It’s because Azure renamed the service but didn’t want to break existing usage. 🤷

In total, you should have 6 GitHub secrets. You could argue some of this stuff doesn’t actually need to actually be a secret (can just be in the workflow yaml) but I have public repositories, so this is nicer.

The entire action will look something like this:

```
      - name: Azure Trusted Signing
        uses: azure/trusted-signing-action@v0.3.16
        with:
          azure-tenant-id: ${{ secrets.AZURE_TENANT_ID }}
          azure-client-id: ${{ secrets.AZURE_CLIENT_ID }}
          azure-client-secret: ${{ secrets.AZURE_CLIENT_SECRET }}
          endpoint: ${{ secrets.AZURE_ENDPOINT }}
          trusted-signing-account-name: ${{ secrets.AZURE_CODE_SIGNING_NAME }}
          certificate-profile-name: ${{ secrets.AZURE_CERT_PROFILE_NAME }}

          # Sign all exes inside the folder
```

```
          files-folder: ${{ env.ARTIFACTS_PATH }}
          files-folder-filter: exe
```

This signs all `exe` files in the named directory. I opened an issue so we can just specify a single filename.

Success looks like this:

```
Submitting digest for signing...
OperationId 9823489-2398492348-2134234: InProgress
Signing completed with status 'Succeeded' in 2.9607421s
Successfully signed: D:\a\pamplejuce\pamplejuce\Builds\Pamplejuce_artefacts
Number of files successfully Signed: 1
Number of warnings: 0
Number of errors: 0
Trusted Signing completed successfully
```

Debugging

No certificates were found

```
The following certificates were considered:
Issued to: localhost
Issued by: localhost
Expires: Fri Apr 25 16:54:32 2025
SHA1 hash: SOMEHASH

After EKU filter, 0 certs were left.
After expiry filter, 0 certs were left.
SignTool Error: No certificates were found that met all the given criteria.
```

This means that the call out to Azure wasn't invoked. There could be a couple reasons for this, double check the following:

- The `dll` wasn't found. The path should be exactly to `Azure.CodeSigning.Dlib.dll` (note that it's **not** `Azure.CodeSigning.dll` and **not** `Azure.CodeSigning.Dlib.Core.dll`)
- Make sure the `x64` and `x86` situation is aligned. Both the dll and the signtool executable need to be using the same version.
- Make sure you are on a recent enough version of signtool. As of April 2024, Microsoft support recommended `10.0.2261.755` or later.
- Make sure you are using the 64bit version of signtool and the Dlib and not the 32bit version.

Number of Errors: 1

Surely an award winning error message. Note this is with `/v` and `/debug` settings on, lol:

```
Number of files successfully Signed: 0
Number of warnings: 0
Number of errors: 1
```

I tried everything to resolve this one. In the end, the issue was the path to the thing I was signing was wrong! I was an idiot and trying to sign a *folder* (that for some reason was called `MyPlugin.exe`.)

GitHub Action 403s:

```
Azure.RequestFailedException: Service request failed.
Status: 403 (Forbidden)
...
Error information: "Error: SignerSign() failed." (-2147467259/0x80004005)
SignTool Error: An unexpected internal error has occurred.
```

This probably means your application user (client id/secret) doesn't have the `Trusted Signing Certificate Profile Signer` role, see Step 5.

Silent signtool failures

Unfortunately there can be various silent failures from signtool. Check the following:

- You have at least .NET 6.0 installed, check with `dotnet --list-runtimes`.
- Make sure you are using a 64bit version of signtool unless you are a 32 bit system.
- If you run into anything else, please check out Event Viewer and comment below…

What to do if you can't pass identity validation

- Try again.
- Submit different documents.
- Ensure company name is aligned in forms and documents.
- Check out this thread for some inspiration.
- Check out the JUCE forum thread for more edge case detail.

FAQ

> 💡 Check out Microsoft's Trusted Signing FAQ too…

Do I still need to buy a cert and put it into the account?

No. It's all managed for you, by Azure. You just interact with their API via their tools. No more buying and juggling certs.

Do I keep a cert in "Azure Key Vault" or something?

Nope. `Azure Key Vault` is a different service, for the old school manual certs.

Can I use AzureSignTool?

Again, no. That's for the old Azure Key Vault / manual certs.

Is this basically the modern equivalent to signing with an EV cert?

Yes. You get instant reputation.

How does smartscreen reputation work? Will I get instant reputation?

Your Azure trusted signing account will start off with a base level of reputation. Reputation now belongs to the code signing identity, no longer the individual certs (as the actual certs are rotated daily and are now an implementation detail).

Can I give my devs access to the account?

Yes, there's full RBAC control. As far as I'm aware there's no additional charge for additional accounts, etc.

Do I need to pay some sort of Azure "subscription"?

No, an azure subscription is a record/resource you need to setup (see step 2) so that Azure can bill you. It's just bureaucratic b.s. needed for the big enterprise bois. Just a hoop to jump through, no additional cost.

Why does my business need to be 3 years old?!

Apparently this has something to do with "Code Signing Baseline Requirements.

Microsoft is working on allowing anyone (“personal” or businesses with less years of tax history) to do signing by doing extra identity proofing. They hoped to have it out by the Public Preview. As of Sept 2024, it’s not ready, but there was a comment in Aug 2024 from the team showing it’s a current priority.

I have heard multiple accounts of people with less than 3 years of history passing validation. My advice would be to try it, but set expectations low.

Additional Resources

- KoalaDSP’s docs which include details on AAX support.
- Microsoft’s Trusted Signing docs.

Posted on April 25, 2024 by sudara
Last Updated 3 months ago

c++, juce

Comments

Kengo
February 18, 2025

Thanks a lot, Sudara! Integrated it into my CI!

Reply

Dane Fisher
February 8, 2025

Problem resolved…

I've downloaded the .msi from here: https://learn.microsoft.com/en-us/azure/trusted-signing/how-to-signing-integrations

Installed it.

Opened a command prompt and cd'd into: C:\Program Files (x86)\Windows Kits\10\bin\10.0.22621.0\x64

Setup the environment variables
SET "AZURE_CLIENT_ID=…"
SET "AZURE_TENANT_ID=…"
SET "AZURE_CLIENT_SECRET=…"
SET "ACS_DLIB=C:\Users\…
\AppData\Local\Microsoft\MicrosoftTrustedSigningClientTools\Azure.CodeSigning.Dlib.dll"
SET "ACS_JSON=…\metadata.json"

And executed:
signtool.exe sign /v /debug /fd SHA256 /tr "http://timestamp.acs.microsoft.com" /td SHA256 /dlib "%ACS_DLIB%" /dmdf "%ACS_JSON%" "D:\my file to sign.exe"

Trusted Signing

Version: 1.0.68

"Metadata": {
"Endpoint": "https://weu.codesigning.azure.net/",
"CodeSigningAccountName": "trusted-signing",
"CertificateProfileName": "public-certificate",
"ExcludeCredentials": []
}

Submitting digest for signing…

OperationId 87ed3b83-….-….-….-c48eb2e7f1d9: InProgress

Signing completed with status 'Succeeded' in 2.2490083s

Successfully signed: D:\my file to sign.exe

Number of files successfully Signed: 1
Number of warnings: 0
Number of errors: 0

I don’t know why it works when using the installer, I’ve imho had all tools in a portable fashion but I’ll disect everything in a fresh vm to find out how to do without it…

Reply

Dane Fisher
February 7, 2025

Hi,
can anyone help me out, please?

I’ve followed the tutorial to the point and got my business registered only a few hours later.
I want to sign .exe files locally.

dotnet-runtime-8.0.4-win-x64.exe is installed
signtool.exe is from v10.0.26100.1742 [SDK] and x64
Azure.CodeSigning.Dlib.dll is v1.0.60.0 and x64

my metadata.json contains:
{
“Endpoint”: “https://weu.codesigning.azure.net/”,
“CodeSigningAccountName”: “trusted-signing”,
“CertificateProfileName”: “public-certificate”
}

These environment variables are set
SET “AZURE_CLIENT_ID={Bitwarden}”
SET “AZURE_TENANT_ID={Bitwarden}”
SET “AZURE_CLIENT_SECRET={Bitwarden}”
SET “AZURE_CONFIG_DIR=D:\!Compilers\Azure-cli_x64\.azure”
SET “ACS_DLIB=D:\Tools\@Command Line Tools\signtool_x64\Azure.CodeSigning.Dlib.dll”
SET “ACS_JSON=D:\Tools\@Command Line Tools\signtool_x64\metadata.json”

{Bitwarden} just means that the values are stored in Bitwarden and are ofc set as necessary.

When I try to sign an .exe file:
signtool.exe sign /v /debug /fd SHA256 /tr “http://timestamp.acs.microsoft.com” /td SHA256 /dlib “%ACS_DLIB%” /dmdf “%ACS_JSON%” “{file}”

I'll get this:
The following certificates were considered:
After EKU filter, 0 certs were left.
After expiry filter, 0 certs were left.
SignTool Error: No certificates were found that met all the given criteria.

So for some reason the Azure call isn't initiated? Why? What am I missing?

Reply

Jean-Marc Couffin
January 30, 2025

I finally cracked it. What helped:

– align user name with IV user name
– and THIS ONE: Invalid client secret provided. Ensure the secret being sent in the request is the client secret value, not the client secret ID
– SignTool Error: This file format cannot be signed because it is not recognized. It cannot sign .nupkg

Reply

jm
January 30, 2025

Everything went pretty smoothly thanks to your article
Id validation went quick (french id, based in CZ), I had to use my pro email
I did not try the local signing yet,
Setup and triple checked the Github CI and got:
> Invalid tenant id provided. You can locate your tenant id by following the instructions listed here: https://learn.microsoft.com/partner-center/find-ids-and-domain-names (Parameter

'tenantId')
I am wondering if the issue could be related to:
Azure User is a gmail address, Id validation another one (pro)

unsolved yet

Reply

Paul
January 11, 2025

I can't get trusted signing to work with the signtool, receiving a 403 error. However I've checked, amongst other things, step 5b many times.

With this command:
======
$ "C:\Program Files (x86)\Windows Kits\10\bin\10.0.26100.0\x64\signtool.exe" sign /v /debug /fd SHA256 /tr "http://timestamp.acs.microsoft.com" /td SHA256 /dlib "C:\Utilities\Security\Trusted signing tools\Azure.CodeSigning.Dlib.dll" /dmdf "signing.json" "C:\Users\Paul\Projects\N8I\MegunoLink\Software\Visualizers\InterfacePanel\bin\Release\InterfacePanel.dll"
======

I get:
======
Trusted Signing

Version: 1.0.68

"Metadata": {
"Endpoint": "https://wus2.codesigning.azure.net",
"CodeSigningAccountName": "app-signer",
"CertificateProfileName": "mlp-certificate-profile",
"ExcludeCredentials": []
}

Submitting digest for signing...
Unhandled managed exception

Azure.RequestFailedException: Service request failed.
Status: 403 (Forbidden)

Headers:
Date: Sat, 11 Jan 2025 08:54:27 GMT
Connection: keep-alive
Strict-Transport-Security: REDACTED
x-azure-ref: REDACTED
X-Cache: REDACTED
Content-Length: 0

at Azure.CodeSigning.CertificateProfileRestClient.SignAsync(String codeSigningAccountName, String certificateProfileName, SignRequest body, String xCorrelationId, String clientVersion, CancellationToken cancellationToken)
at Azure.CodeSigning.CertificateProfileClient.StartSignAsync(String codeSigningAccountName, String certificateProfileName, SignRequest body, String xCorrelationId, String clientVersion, CancellationToken cancellationToken)
at Azure.CodeSigning.Dlib.Core.DigestSigner.SignAsync(UInt32 algorithm, Byte[] digest, SafeFileHandle safeFileHandle, CancellationToken cancellationToken)
at Azure.CodeSigning.Dlib.Core.DigestSigner.Sign(UInt32 algorithm, Byte[] digest, SafeFileHandle safeFileHandle)
at AuthenticodeDigestSignExWithFileHandleManaged(_CRYPTOAPI_BLOB* pMetadataBlob, UInt32 digestAlgId, Byte* pbToBeSignedDigest, UInt32 cbToBeSignedDigest, Void* hFile, _CRYPTOAPI_BLOB* pSignedDigest, _CERT_CONTEXT** ppSignerCert, Void* hCertChainStore)

SignTool Error: An unexpected internal error has occurred.
Error information: “Error: SignerSign() failed.” (-2147467259/0x80004005)
======

Signtool.exe version: 10.0.26100.1742, dated 2024-09-05
Azure.CodeSigning.Dlib.dll file version: 1.0.68.0; dated 2024-11-04
Trusted validation status = Completed.

I have environment variables set for AZURE_TENANT_ID, AZURE_CLIENT_ID and AZURE_CLIENT_SECRET; all checked many times. Also, I get a different error if I change any of them leaving me thinking that the app authentication isn’t what’s being forbidden. When I look in the “Trusted signing account” -> “Access control (IAM)” -> “Role assignments”, I see “app-signer” is an “App” type and has the “Trusted Signing Certificate Profile Signer” role, created in step 5b.

It looks similar to this issue (https://github.com/Azure/trusted-signing-action/issues/13), but I’m using signtool.exe locally. Still, I couldn’t see anything different in the ImageMagick instructions that weren’t already covered here.

Is there any logging on the azure end where I could see what it is forbidding? Any suggestions or comments would be much appreciated!

Reply

Paul
February 6, 2025

I resolved the 403 problem I was helping with help from Jimmy at Microsoft. I was using the name of the account created in step 4 (that is, the app registration user) for the "CodeSigningAccountName" in the json file. This is wrong. Obviously, in hind-sight, this should be the name of the code signing service account created in step 3.

Pete
January 10, 2025

You. Are. Awesome!
Without this Step by Step guide I would be lost. Even registration to Azure was painful, this is no way I would figure out on my own. Heck, even GPT would probably get lost.
P.S. didn't want to wait or risk anything with Organisation verification, so I just made Individual for now. 10 mins.

Thank you!

Reply

Faidideq

January 6, 2025

Big thanks for this article.

Other things to beware of:
– Dotnet runtime installed from vs2022 is not globally available even in the VS developer console.
– If you are using powershell and read the Koala article, the variables in the signtool call will not expand with %VARIABLE% so need to be replaced with $VARIABLE. The cryptic error associated with the paths being invalid and not expanded was the following:

For the record, if you end up with this error:

Done Adding Additional Store
SignTool Error: An unexpected internal error has occurred.
Error information: "Error: SignerSign() failed." (-2147024846/0x80070032)

Reply

Paul
January 5, 2025

Thanks; this was very helpful. Setup from NZ and identity verified, after confirming email address, in 15 minutes.

Reply

JeffL
November 27, 2024

mine continues to be denied. no reason is given. I assume its the 3 year rule. absolute garbage.

Reply

bukacdan
November 24, 2024

Has anyone come across the Error: “SignerSign() failed.”(-2147024846/0x80070032).

In the FAQ page it says “Ensure that you’re using the latest version of SignTool”, but I’m on the latest (10.0.26100).

Any ideas? It’s driving me crazy

Reply

Greg Schier
November 23, 2024

I had a company and Azure account that are both less than a month old, and got verified without issue.

Reply

Tyler Shelton

November 18, 2024

This was very helpful for signing .exe files. Our app uses click once so we need the .application file (Application Manifest) to be signed as well. It seems signtool.exe does not work for this. Has anyone figured out how to sign that .application file using a trusted signing account?

Reply

Shannon
October 26, 2024

Thank you so much for the great and neat article. I followed the steps very quickly. The identity validation process took 2 days: because I have DUNS number, the business verification passed within an hour, but I was stucked by domain ownership verification. After turning of the privacy guard of the domain, I uploaded the screen shot of the whois information on icann.org in pdf format twice, neither worked, finally after reading the support page carefully I realized they may only accept query result on whois.com, yes, it approved within 2 hours.

Reply

Justin
October 13, 2024

Superb article. Followed the steps carefully and we now have our apps code signing again! For us the approval came through in like 20 minutes. Took longer to wrangle with signtool. But again the instructions and troubleshooting there really helped. Thank you!

Reply

Maicon Saraiva
October 3, 2024

Hello. I'm Maicon Saraiva from Brazil.
Thanks to your article, I was able to unlock the official manual in one step.
Thank you very much.
I was able to validate the data in approximately 1 day (24 hours). We have had a CNPJ (Brazilian Tax ID) for over 10 years, and I used corporate emails in all fields and in the Azure account. So I think this helped.

Reply

Benoît Andrieu
September 27, 2024

Thanks, I just followed the steps.
The validation from MS was done in 30 minutes (France, Azure customer for +7 years).
I am using signtool through InstallShield 2024.

Reply

Dean Herbert
September 26, 2024

Thanks for this guide.

One thing to note is that using a newer `signtool.exe` on older windows versions will lead to a cryptic error message:

0x80070057
The parameter is incorrect.

The solution is to use the available version of signtool on the system, using something like:

“`csharp
string signtoolPath = Directory.GetDirectories(@”C:\Program Files (x86)\Windows Kits\10\bin”, “*”, SearchOption.AllDirectories)
.Select(dir => Path.Combine(dir, @”x64\signtool.exe”))
.Where(File.Exists)
.Last();
“`

or

“`powershell
Resolve-Path “C:\Program Files (x86)\Windows Kits\10\bin*\x64\signtool.exe” | Select-Object -Last 1
“`

Reply

Jerry
October 2, 2024

Hi,
By “older” windows you mean Win10 in general? How did you find it out, via the support? I ran (probably) into the same problem when trying to use the latest signtool. Simply 0x80070057 and that’s it. No matter what I enter in the metadata.json file, the result is the same, but if I specify an incorrect path of the metadata.json for the signtool, the message is “SignerSign() failed 0x80070003

However, for my region (weu), I can’t complete the registration of the Microsoft.CodeSigning resource provider. It’s in the permanent “registering” state. From what I saw someone asked the same here: https://learn.microsoft.com/en-us/answers/questions/2084447/codesigning-service-provider-stuck-in-registering

jerry
October 3, 2024

Please ignore. Apparently, it was about setting a working path to the original latest sdk/signtool path or running it in its original location. (Although the Codesigning resource provider is still in the “registering…” state.)

Viktor
November 12, 2024

MS docs say to use signtool from Microsoft.Windows.SDK.BuildTools nuget package https://learn.microsoft.com/en-us/azure/trusted-signing/how-to-signing-integrations#download-and-install-signtool

Eirik Bakke
September 14, 2024

Excellent guide! Completing the Azure setup, adding the signing step to my app’s GitHub Actions flow, and verifying that my app’s freshly signed MSI installer did not generate a SmartScreen warning when downloaded and opened took less than half a workday.

I was previously using an EV certificate from Sectigo; not sure if my app’s previous reputation played any role in avoiding the SmartScreen warning or not.

The Identity Verification step with a US DUNS number took only 16 minutes, and did not require any additional paperwork to be submitted in my case.

Thank you very much for writing this up!

Reply

Craig
September 12, 2024

Thank you for this guide.

Unfortunately for me, my numerous attempts to get past validation have all failed. Despite my company existing for over 16 years with a valid tax ID, the only thing I get back is the dreaded “your validation status has been update to ‘Validation Fail status’”. And this is before I even have the chance to submit any documentation.

No reason is ever given. I guess we’re supposed to be clairvoyant and read their minds? I know this is a “Preview” feature, but quite honestly, given the dreadful lack of feedback, this isn’t even “Alpha”.

I guess it’s time to buy one of those expensive and clunky USB tokens.

Reply

sudara
September 12, 2024

Hmm, what kind of business entity do you have? Are you submitting an EU VAT ID or more like an internal country tax number? I would try to switch things up.

Agreed that a feedback mechanism is straight up missing, which makes it especially problematic when something goes wrong and you don’t know what. Rather than retrying the same information, people have succeeded by trying variations.

David
September 12, 2024

Nice to find some good instructions. Did you use your personal Microsoft account to create an Azure subscription, or did you first create a new Microsoft account for your business.
I have a sole proprietorship (one person business) and try to not mix up personal and business email addresses, accounts, etc.

Reply

sudara
September 12, 2024

Thanks! Makes sense to be to setup a new account in your case. Won't really matter either way, you just need a way to get into Azure.

David
September 17, 2024

For the record: Validation took two working days for my company. I applied Friday morning, and validation was completed Tuesday morning.

It took me quite a few hours to get the actual signing to work though. The main problem was that I used the 'App registration' (your "trusted-signing" entity) for the "CodeSigningAccountName" where it should have been the name of the 'Trusted Signing Account'.
This results in :

Submitting digest for signing...
Unhandled managed exception
Azure.RequestFailedException: Service request failed.
Status: 403 (Forbidden)

In case anyone needs to know, it works fine from Advanced Installer as well.

So happy that I never have to purchase an expensive Code Sign certificate again!

Adrian
September 5, 2024

This guide has been absolutely invaluable!
I would never have got through the process without it.
Thank you so much!

Reply

sudara
September 12, 2024

Thank you!

Levminer
August 28, 2024

If anyone is interested I made an easy to use CLI: https://github.com/Levminer/trusted-signing-cli

Reply

Krister KNO Oksnes
August 27, 2024

Hey! Do you know if its possible to sign JAR files with this?

Reply

Steve Atkins
August 23, 2024

Wonderfully helpful article!

I've only done very lightweight github actioning before, and I'm completely new to Azure, but I just signed my binary successfully on the first attempt!

It took about 12 days to validate identity, despite having a DUNS number and adequate company history. The holdup seemed to be proving ownership of the domain name; everything else was approved almost immediately.

Reply

sudara
August 23, 2024

Awesome to hear! Nothing like that first try working out!

Corey
August 8, 2024

Great article on a thorny subject! Any chance you have contact info for someone on the MS team to share privately? All of the people who contributed to the MS doc you linked to aren't reachable via GitHub. I'd love to use this for an OSS project I run, but we just launched a new subsidiary to support the project… which is technically a new company that doesn't meet the 3yr tax minimum (but whose parent company does).

Reply

sudara
August 23, 2024

Sorry! It's been a while since I was in contact, and GitHub was the last place I had contact. I know they are working behind the scenes on OSS / personal plans, but it might still take a little while. What I do suggest is just giving it a go. I've had friends pass identity validation on the 2nd or 3rd attempt who have older businesses, they don't seem to be strict about that rule.

Dan
July 26, 2024

Thanks a lot for publishing such detailed instructions, which saved us a lot of time. However, we ran into an issue with certificates in the exe and msi files signed with signtool.

We see the following message: 'This certificate cannot be verified up to a trusted certification authority.'

Also, SmartScreen complains about the signed files.

Do you know what could be causing this?

Reply

sudara
August 23, 2024

I'm a bit unclear what you mean by "certificates in the exe" — are you signing something else beyond the exe/msi?

Paul
July 24, 2024

Thanks, very helpful guide, you saved me!
Small typo in version:
"As of April 2024, Microsoft support recommended 10.0.2261.755 or later"

Reply

burhan
July 22, 2024

Done Adding Additional Store
SignTool Error: An unexpected internal error has occurred.
Error information: “Error: SignerSign() failed.” (-2147024846/0x80070032)

Getting This error

Reply

Edijs
July 22, 2024

Great article. Do you know how to get it working with AZURE_CLIENT_CERTIFICATE_PATH instead of the client secret? I cannot seem to get certificate authentication working, client secret works fine.

Reply

sudara
July 22, 2024

Hmm, I don’t have experience with it. Maybe making sure AZURE_CLIENT_ID isn’t set, since maybe it takes priority? Maybe someone else knows…

Muhammad Samiullah Siddiqui

July 19, 2024

SignTool Error: This file format cannot be signed because it is not recognized.

file extension “app”?

Reply

Sami
July 18, 2024

Signed Success But Still File not showing Certificate?

Reply

sudara
July 18, 2024

Can you provide more details? Did you try to use /verify?

Muhammad Samiullah Siddiqui
July 18, 2024

I used the Trusted Signing GitHub Action, and it was successful. There was a message saying that one file was successfully signed. However, the file located in the GitHub repository did not reflect any changes. When I open the file again, it is not signed. Is there another method, or am I missing something?

sudara
July 19, 2024

Does that mean you are seeing the smartscreen warning when opening the executable? What kind of file is this, an installer? How are you checking if it is signed?

Martin
July 12, 2024

Thanks a lot for this guide.
I got signing setup now 🙄
Identity validation was very quick for me. Took only 15 minutes and I did not even have to upload any papers. Maybe because I have a German “UG”, which by law is required to upload annual financial papers to a public server. Maybe they took it from there?

The only problem I encountered was using signtool: I was using the lowest recommended version, from Win SDK 10.0.19041. But that did not work. Signtool could not find my Azure trusted certificate for some reason.
After updating the Win SDK to 10.0.26100 it worked fine and I could sign successfully.
EDIT: Just seeing that Pablo Diaz left a similar comment.

Reply

sudara
July 12, 2024

Thanks, I'll update the article with some more info around this stuff!

bgporter
July 8, 2024

Sudara —
Major thanks for the effort here; this was incredibly helpful. Getting the identity things in order took a few days, but I suspect that was because I started the process immediately before a major US holiday and the verification was complete when I returned, which is fine.

Only bump for my org is that we use CircleCI for builds, and I'm reaching out to them in hopes they have something like the GitHub Actions that are ready to use; if I learn anything I'll be sure to report anything useful back here.

Reply

sudara
July 12, 2024

The GitHub action is just a light wrapper around the Powershell implementation. See this forum post for a powershell script that'll work on any CI: https://forum.juce.com/t/azure-code-signing-for-plugin-developers-guide/60391/95

Norm
June 26, 2024

From the FAQ cited above:

Does Trusted Signing issue EV certificates?
No, Trusted Signing doesn't issue EV certificates, and there are no plans to issue them in the future.

So are things signed with Trusted Signing treated as OV signed? ie we have to endure the reputation check, 3000 download thing?

Reply

sudara
July 12, 2024

From my FAQ:

Is this basically the modern equivalent to signing with an EV cert?
Yes.

Trusted signing is a different paradigm from EV/OV. You get instant reputation. Read my FAQ for more info.

Gautam Jain
June 25, 2024

Thanks a lot Sudara for compiling and sharing the missing important details.

I was able to successfully sign, but the certificate expires after 3 days.

Do I have to keep signing my apps and uploading to my server every 3 days?

Reply

sudara
July 12, 2024

Instead of the certs being something you hold like the old school EV/OV, they are now just an implementation detail on Azure. A new cert is made for you every day behind the scenes. This add some features, such as allowing for time precise revocation. Note that it doesn't mean a *signed binary* is invalid after 3 days, just the certs are rotated.

Alex Voina
June 10, 2024

Thank you so much! It took us close to a month to complete this process, but I can't imagine doing it without this article.

Even after getting past Identity Validation, I went straight to Signing in Github Actions, and got an error that was mentioned at the end (403 – forbidden). I followed your suggestion and had a second look at what I've configured at step 5, and that was it. I discovered I overlooked a tiny detail – assigning the role to the app registration instead of the user.

I could not believe you have actually pointed me to the source of error that I did by not reading this same article carefully enough. THANK YOU!!

I'm happy to see you have referenced the Github issue I have opened with Azure, I hope that can provide additional help for others.

Keep doing this!

Reply

Pablo Diaz
May 31, 2024

Terrific document. You saved me so much headache, particularly the bits about authentication. I just filed a request for MSFT to include this critical in their documentation.

By the way, one thing where I stumbled was the release version of the Windows kit. Microsoft's docs are inconsistent, suggesting version "10.0.19041 or later", and then "minimum version 10.0.2261.755". The latter is correct; The older version does not work, and this took me a while to figure out.

Reply

sudara
July 12, 2024

Thanks for this, I'll update the article with some more info...

Adam
May 24, 2024

Thank you Sudara for such a comprehensive and easy to follow tutorial.

I wanted to report that I got verified within about two hours, using my UTR code for my UK limited company.

With the help of Koala DSP's signtool steps I am now able to sign Windows installers with no Smartscreen popups! Yay 🙂

Reply

sudara
May 24, 2024

Nice, Adam! Thanks for reporting back!

claes
May 24, 2024

Thanks alot for this, got it running fairly quickly. At one point it failed silently (crash) with info only available in the Event Viewer, but it turned out the .NET 6 runtime was not installed and after that it worked.

Reply

sudara
May 24, 2024

Ah good to know re: .NET. I wasn't sure how "real" that dependency was despite it being mentioned some docs. I'll update the article.

Matthew
May 18, 2024

After several weeks of frustration and lack of errors visibly being emitted by signtool, it turned out that the 32-bit version of signtool was crashing, and a fault was logged in Event Viewer:

Faulting application name: signtool.exe, version: 10.0.22621.3233, time stamp: 0xeef8b361
Faulting module name: msvcrt.dll, version: 7.0.19041.3636, time stamp: 0x7e27562f
Exception code: 0x40000015
Fault offset: 0x0003a65b
Faulting process id: 0x1bec
Faulting application start time: 0x01daa8d5d610c055
Faulting application path: C:\Program Files (x86)\Windows Kits\10\bin\10.0.22621.0\x86\signtool.exe
Faulting module path: C:\WINDOWS\System32\msvcrt.dll

Changing to the 64-bit version of signtool (and Azure.CodeSigning.Dlib.dll) makes code signing work correctly.

Reply

sudara
May 18, 2024

Yikes! Glad you got it working. When you say “lack of errors” did you not get the “No certificates were found” error? It just didn’t output anything? I updated the “Debugging” section to remind people to check 32 vs. 64 bit.

Rich
May 15, 2024

Thanks for this, helped me quite a bit. I got validated within an hour using my DUNS number.

Couple of things to note:

if you're starting with a fresh Azure account you'll automatically have a subscription, so no need to create one, but you do have to upgrade to pay-as-you-go before anything will be possible.

It wasn't so immediately clear to me from these instructions that you need to create the "Trusted Signing Certificate Profile Signer" role using the name of the App Registration from the previous step. Also the Azure UI won't show you the App Registration as on option to assign the role to without first searching for the name you gave it, that caught me out for a little while.

So glad this service exists, my traditional certs expired 3 days ago, I was dreading having to go through the process again and even more now that you can't just get a file a be done with it AND worse the price is just astronomical now! And even better I tested my first successfully signed installer and didn't see that damned blocked by SmartScreen nonsense!

Reply

sudara
May 15, 2024

Thanks for the detail here, Rich! I'll do my best to clarify those items.

Also congrats! I also didn't see SmartScreen on the first sign, was good vibes!

Brian Fritton
May 14, 2024

Thank you very much. Lots of head banging to get this working and your write up helped a lot.

One thing I encountered not found here I'd recommend adding: On our first Identity Validation attempt, I waited 10 full days with no update past the "In Progress" status. We did that with our DUNS #. I did verify the email as well. Since MS provides no unpaid support, I just submitted another Identity Validation request, this time with our company EIN (tax ID) and that one

succeeded within an hour. So… if you don't get a status past in progress in a day or two, just try another one with a different ID method.

Reply

sudara
May 15, 2024

This is great advice, thanks for sharing it. I'll pop it in the article too.

Jaxel
April 30, 2024

I came here to say, kudos, this covers a lot of territory. Wanted to add for anyone who reaches here:

– 403s can sometimes be seen when customers have multiple identities on their configuration, so ensure you are using the proper one.
– Also, for customers outside of an Azure environment there's a nagging message that might pop up (https://github.com/Azure/azure-sdk-for-net/issues/29471), you can easily avoid it by excluding managed identity credentials on your config:

{
"Endpoint": "",
"CodeSigningAccountName": "",
"CertificateProfileName": "",
"ExcludeCredentials": [
"ManagedIdentityCredential"
]
}

Or the appropiate action from the CI actions.

Reply

sudara
May 15, 2024

Thanks Jaxel!!

Leave a Comment

Your email address will not be published. Required fields are marked *

Comment *

Name *

Email *

Website

☐ Save my name, email, and website in this browser for the next time I comment.

Post Comment

Related Posts

How to code sign Windows installers with an EV cert on GitHub Actions

How to code sign and notarize macOS audio plugins in CI