



Decoder rings!
The Adobe License Decoder tool
for fun and profit



Darren Wallace

Lead Systems Engineer |

darren@datajar.co.uk

My name is Darren Wallace and I'm a Lead Systems Engineer with dataJAR. - Joking also known as an "Adobe Deployment Engineer"?

You can find me on most places, Slack, Twitter etc as Daz_Wallace.



The Adobe License Decoder tool

and Today I'm talking about a free open source tool Adobe have released called the License Decoder tool (also called ALD, as I like acronyms!)

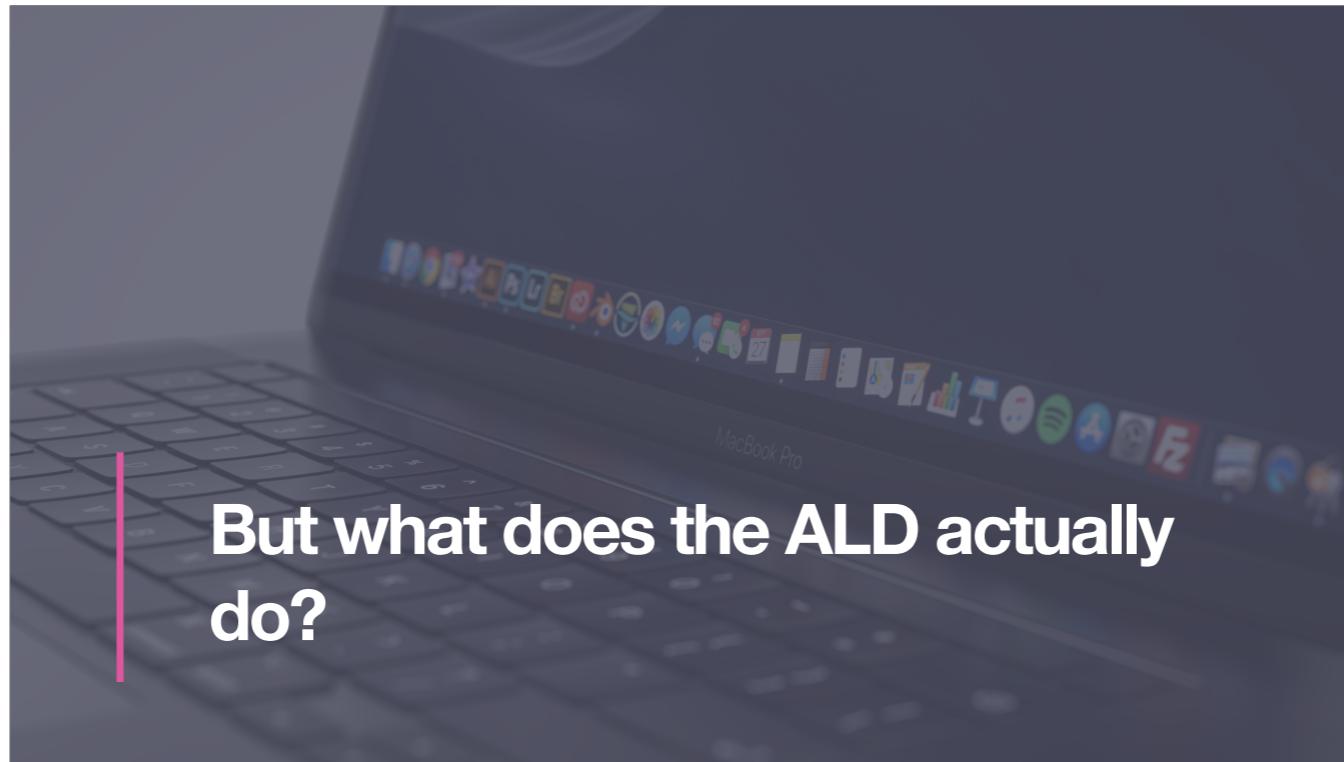


What is the ALD?

So first of all, what is the Adobe License Decoder?

The ALD was an internal tool to help the teams at Adobe understand what licenses were in use on devices and within packages. It has been written in Rust and has binaries for both Windows and macOS - with a separate build for Intel and Apple Silicon

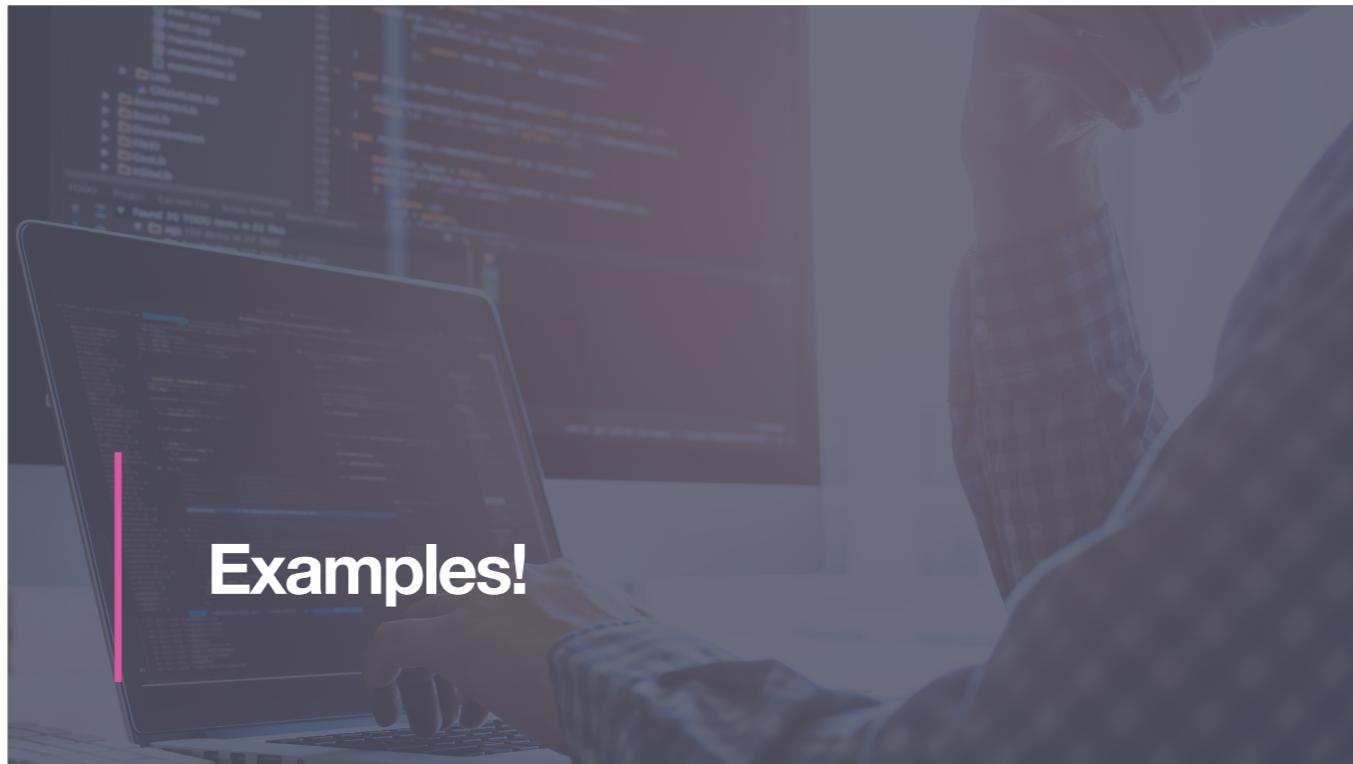
It looks like the tool was released back in November 2020, and was mentioned in the #adobe Slack channel at that time, however it wasn't until it was mentioned again last month that I spotted it!



But what does the ALD actually do?

So the ALD is a command line tool that can interrogate certain things to determine the Adobe license in use. It will only work on Shared Device Licenses (or SDLs) and Feature Restricted Licenses (or FRLs). - For those that don't know, FRLs are a specific type of device license for enterprises, typically but not always with air-gapped networks. You'd normally need to be a rather large customer to get access to it!

You can run the tool on a device with Adobe software installed, on an Adobe package you have locally, or you can grab a copy of the Operating Configs directory from a device and run it on that.



How about some examples of using the tool?



On Device

`./adobe-license-decoder`

So the first example is how to check the licenses status from the device you're currently on.
Just run the command. Simples.



On Package

```
./adobe-license-decoder \
/path/to/Adobe/Directory/
```

What about on package? Just run the command followed by the path to your Adobe package folder. Don't point this to the final package as it only reads from the .ccp file.



On Directory

```
./adobe-license-decoder \
/path/to/copied/OperatingConfigs/
```

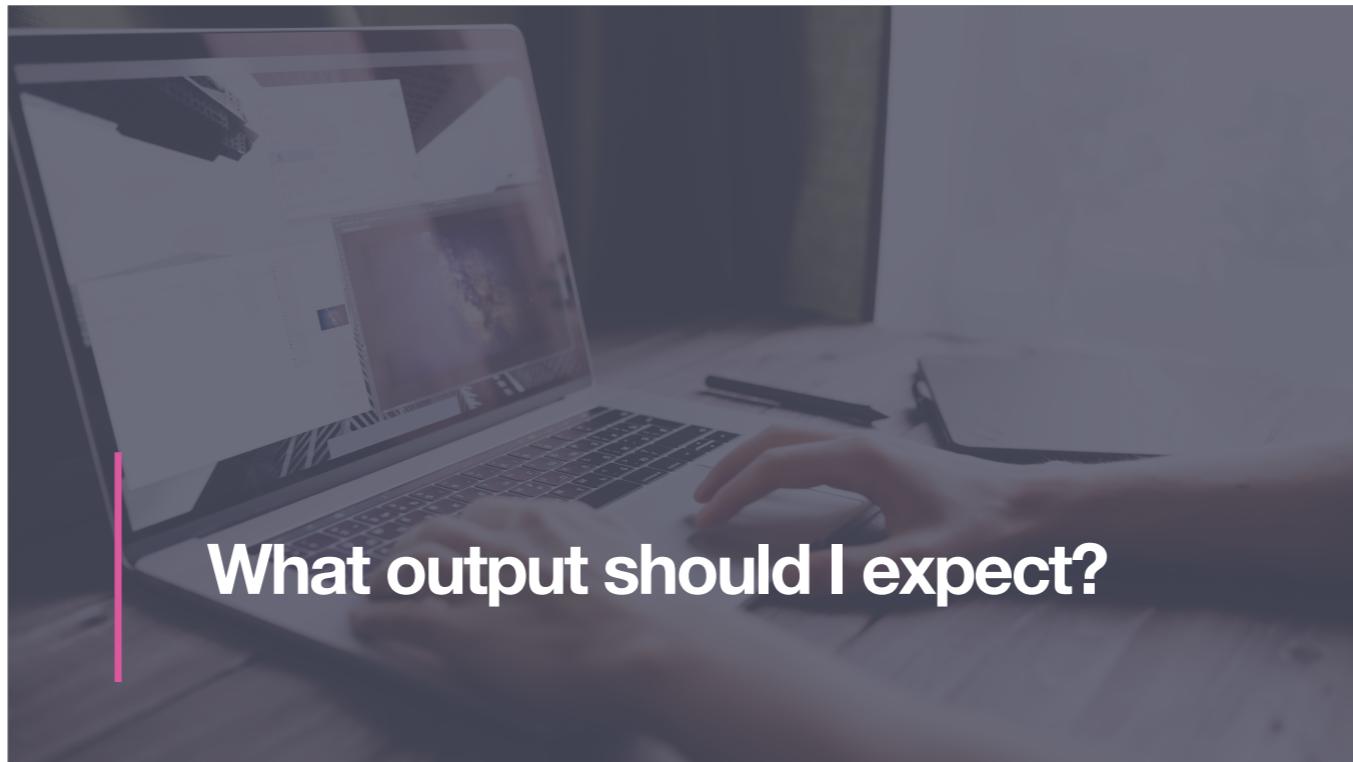
And what about if you've grabbed the Operating Configs directory from a device? Just run the command again, but followed by the path to your local copy of the directory



On Directory

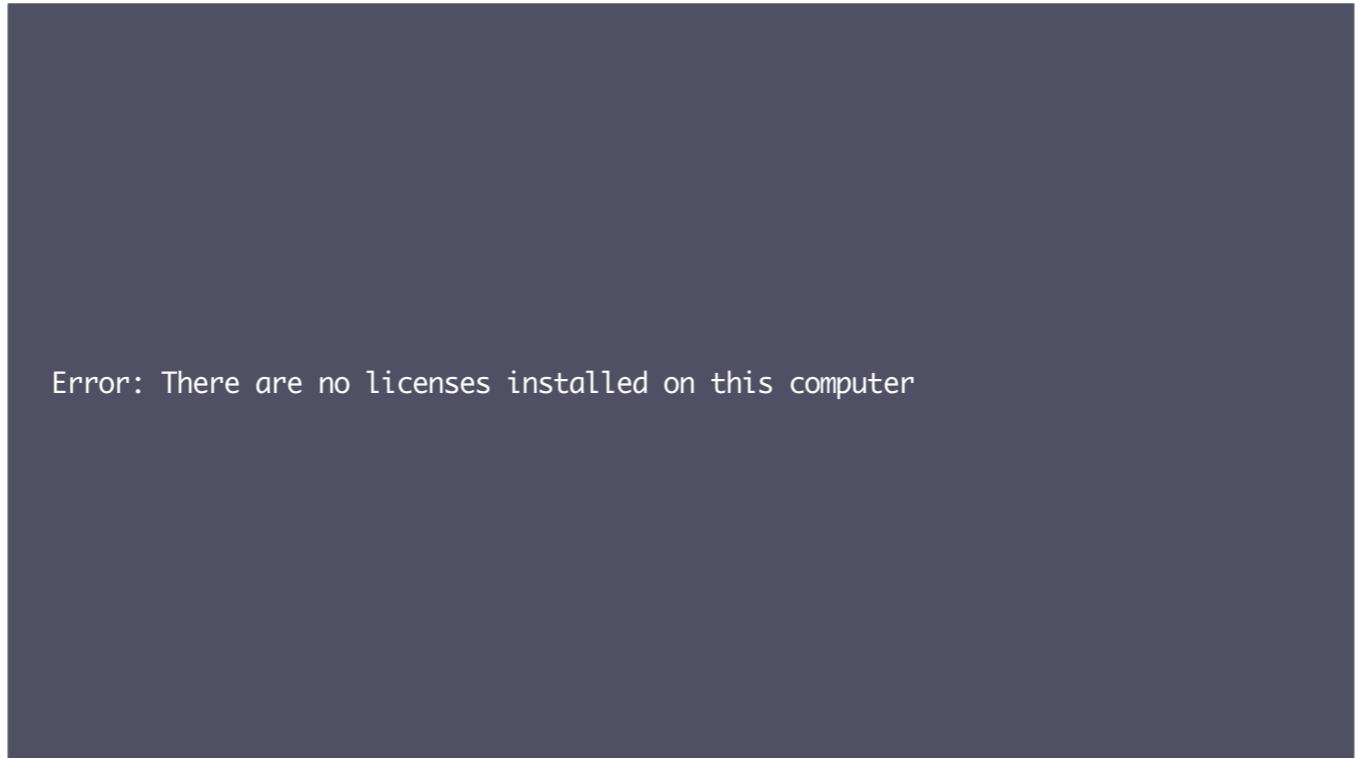
/Library/Application Support/Adobe/
OperatingConfigs

Oh you can find the OperatingConfigs directory on devices at Library > Application Support > Adobe > OperatingConfigs



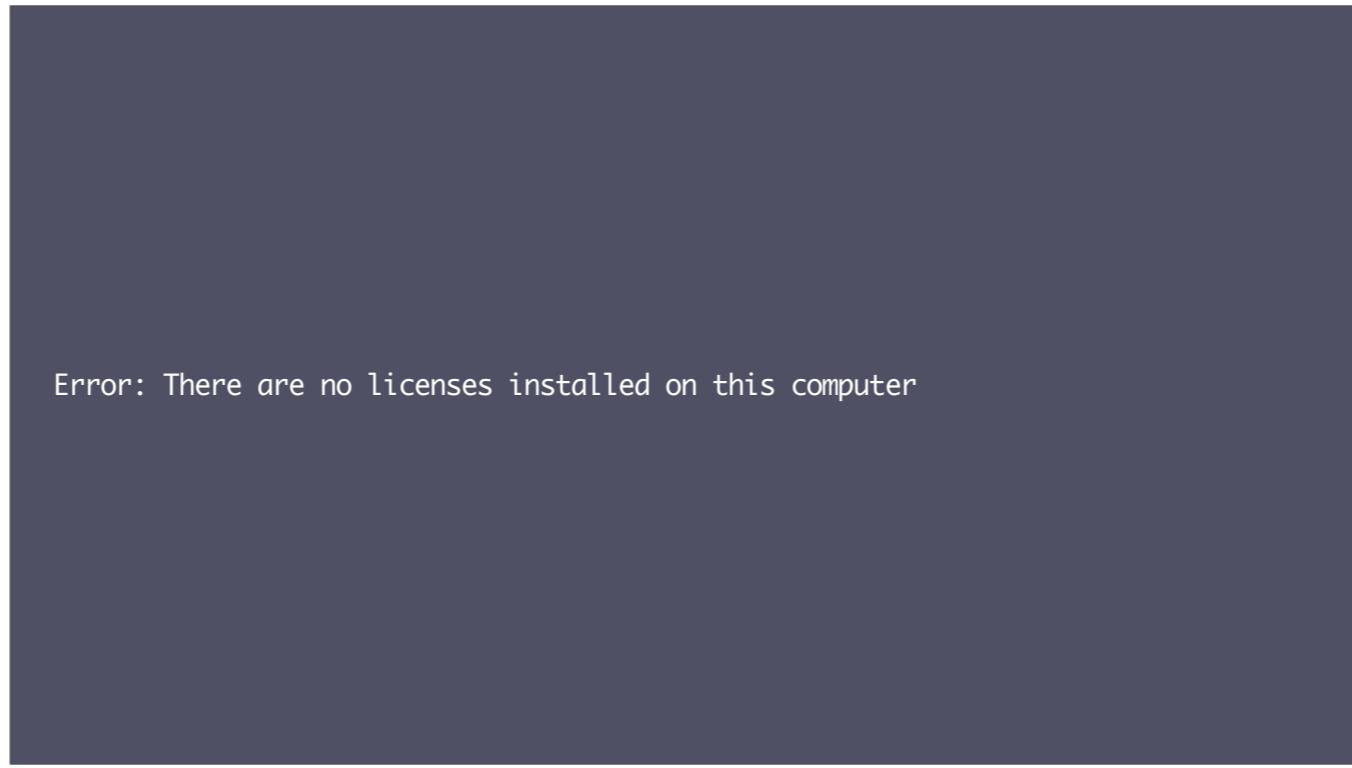
What output should I expect?

It'll be handy to give you some sample outputs from these commands, so here we go.
These have all been tested with an SDL license as I don't have access to an FRL to test with.



Error: There are no licenses installed on this computer

If you don't have any Adobe Apps installed, you'll get the above output. Previously this output was a little more...wordy but a recent update to the tool added a feature request by me to improve this!



Error: There are no licenses installed on this computer

If you have the Adobe Apps installed, but they are unlicensed (or using Name User Licensing) this is the output you'll see. You may notice that its the same as for the 'no Adobe Apps installed' output

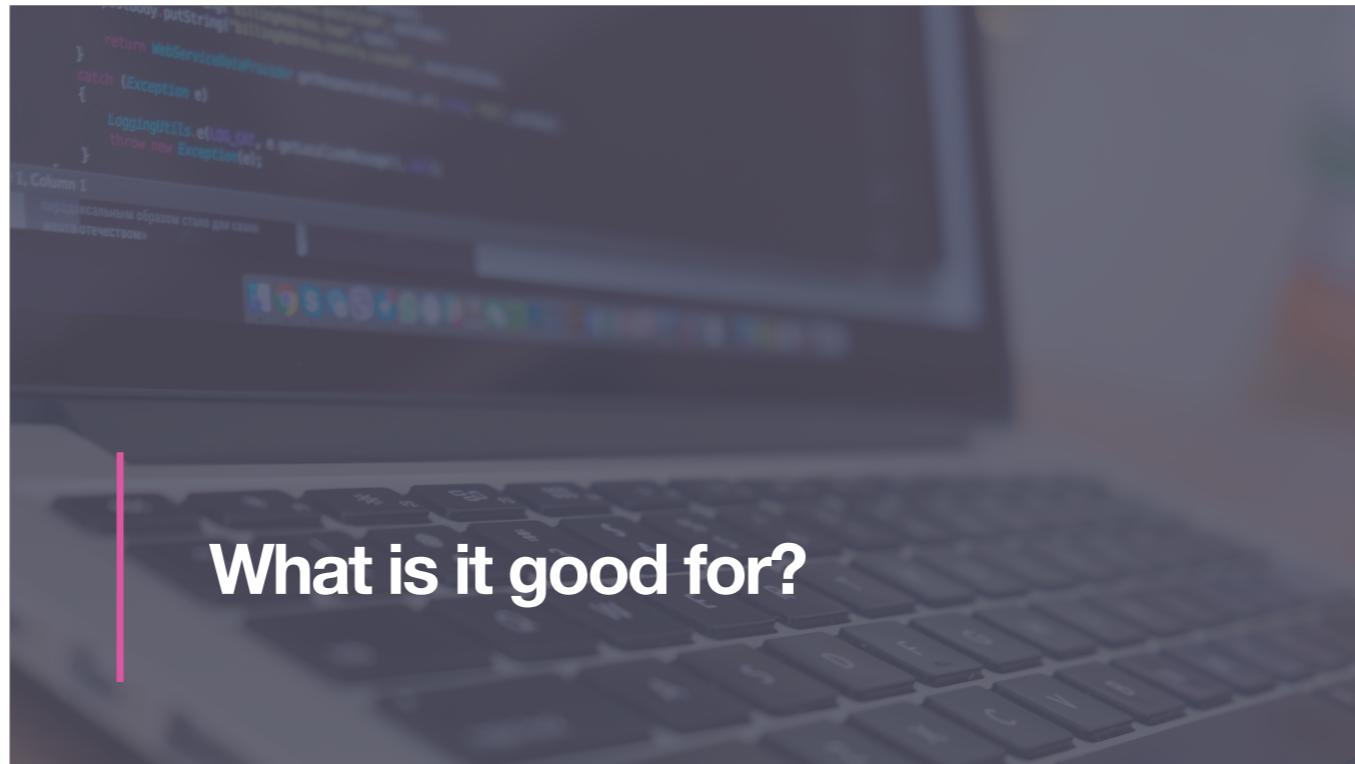
```
App ID: MediaEncoder1
Install date: 2021-03-06 15:49:41 +00:00
14: [REDACTED]...-90.operatingconfig
App ID: Nimbus1
Install date: 2021-03-06 15:49:54 +00:00
15: [REDACTED]...-90.operatingconfig
App ID: Photoshop1
Install date: 2021-03-06 15:49:36 +00:00
16: [REDACTED]...-90.operatingconfig
App ID: Prelude1
Install date: 2021-03-06 15:49:34 +00:00
17: [REDACTED]...-90.operatingconfig
App ID: PremierePro1
Install date: 2021-03-06 15:49:52 +00:00
18: [REDACTED]...-90.operatingconfig
App ID: Rush1
Install date: 2021-03-06 15:49:55 +00:00
```

And lastly, what if you have some Adobe Apps installed with an SDL? You'll get the above. It's fairly long and wordy output. I've also redacted the NPD ID incase it's confidential.

License files for npdId: [REDACTED]:
License type: SDL
License expiry date: controlled by server
Precedence: 90 (CC All Apps)

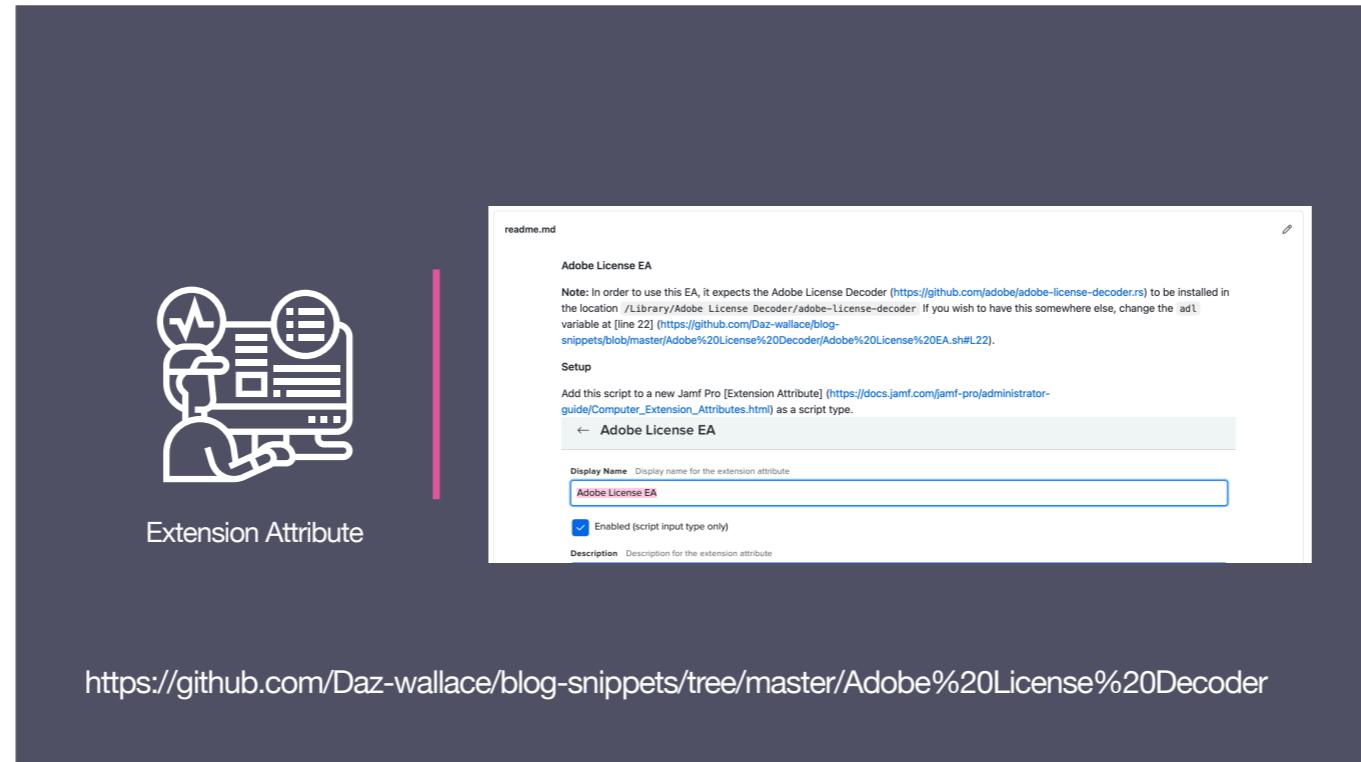
The important bit is at the top here where you'll see the type of license and the expiry date.

Pretty handy



What is it good for?

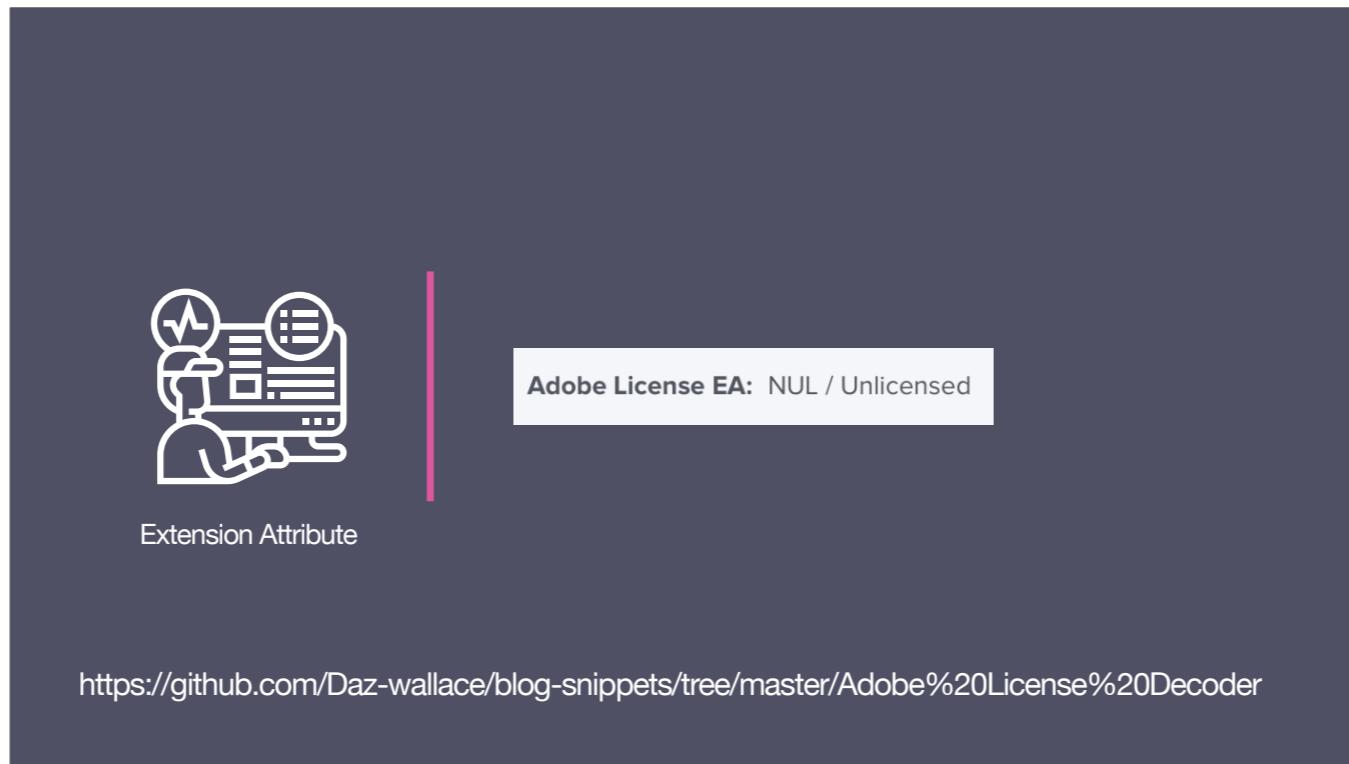
Well other than being a cool and interesting little tool, I can also see some use when troubleshooting issues with a device not appearing to have the correct license.



For you Jamf admins out there, I've also knocked up an Extension attribute that can use the ALD to give you this information from your devices. You can find the link above

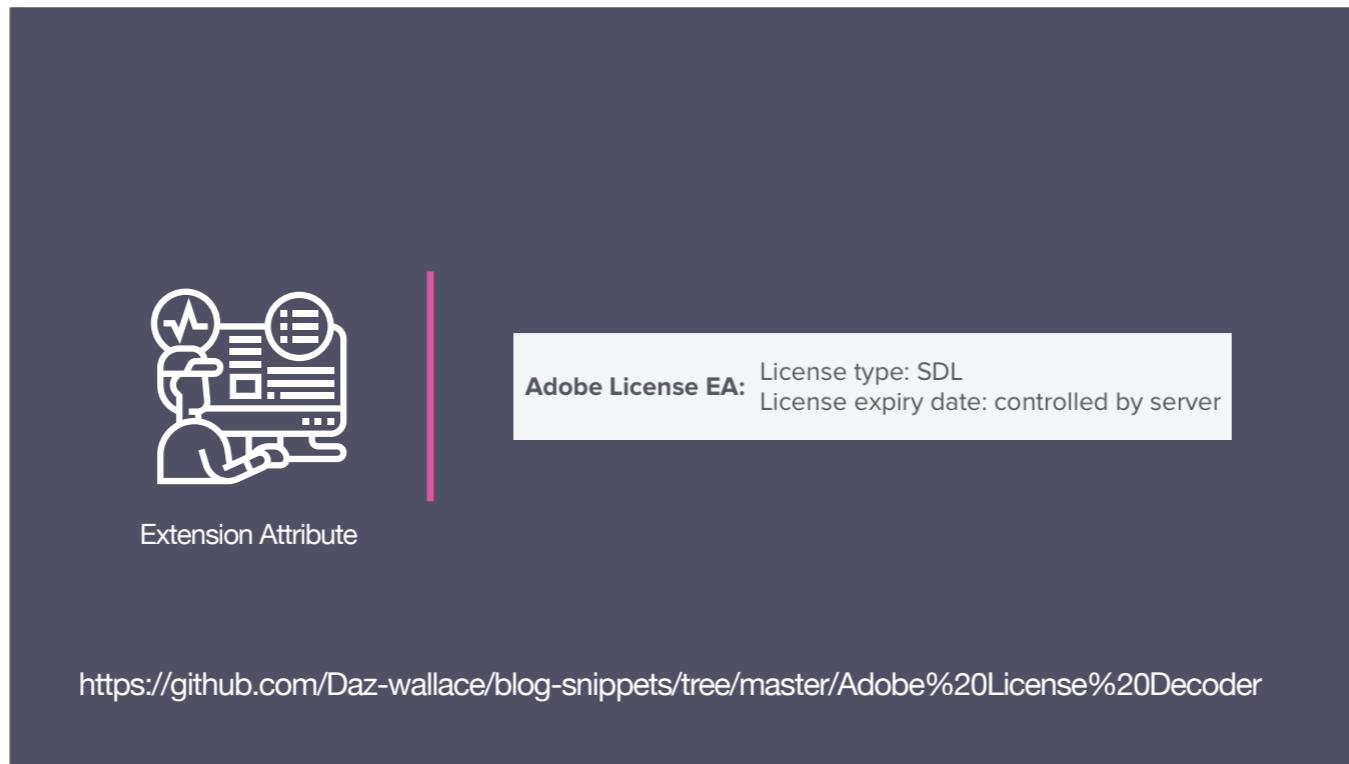


If the system is missing the ALD at the default location (/Library/Adobe License Decoder/adobe-license-decoder) you'll see this error



If the ALD is installed but the device has no Adobe titles, or isn't using SDL or FRL licenses, you'll see this output.

This is basically checking if the output from the ALD command is not 0



If the ALD is installed and the device has an SDL or FRL license, you'll see this output. This output is directly taken from the ALD output

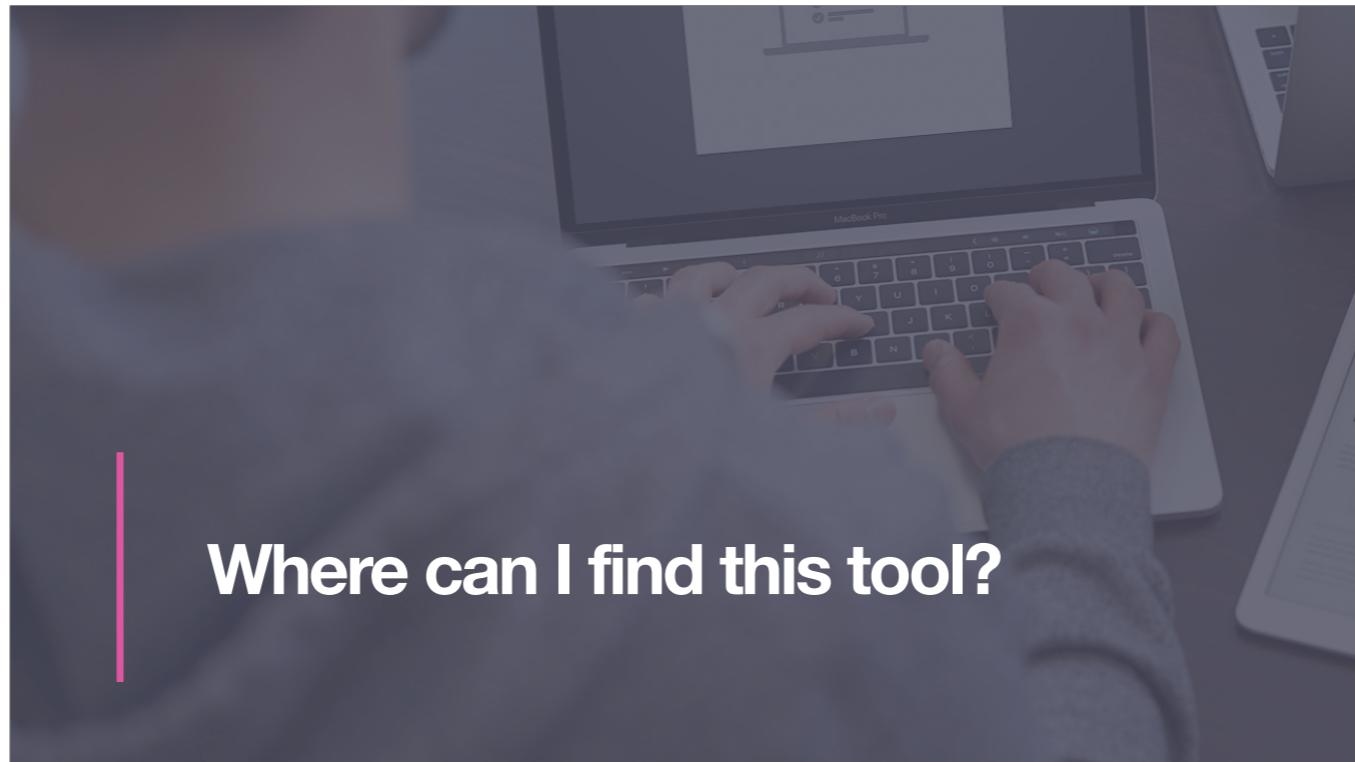


Extension Attribute

Adobe License EA: Unexpected output, please review

<https://github.com/Daz-wallace/blog-snippets/tree/master/Adobe%20License%20Decoder>

And lastly, if the script encounters an unexpected result, you'll also see this reflected as per the message above



Where can I find this tool?

So what about where you can get a copy of the tool from?



GitHub

Adobe License Decoder

Rust CI (stable) passing

Anyone who has worked with FRL or SDL licensing is familiar with the adobe-licensing-toolkit command-line tool for Mac and Windows. This tool runs on client machines in the context of a particular user account and provides information about the state of FRL and SDL licenses that are installed on the machine, including:

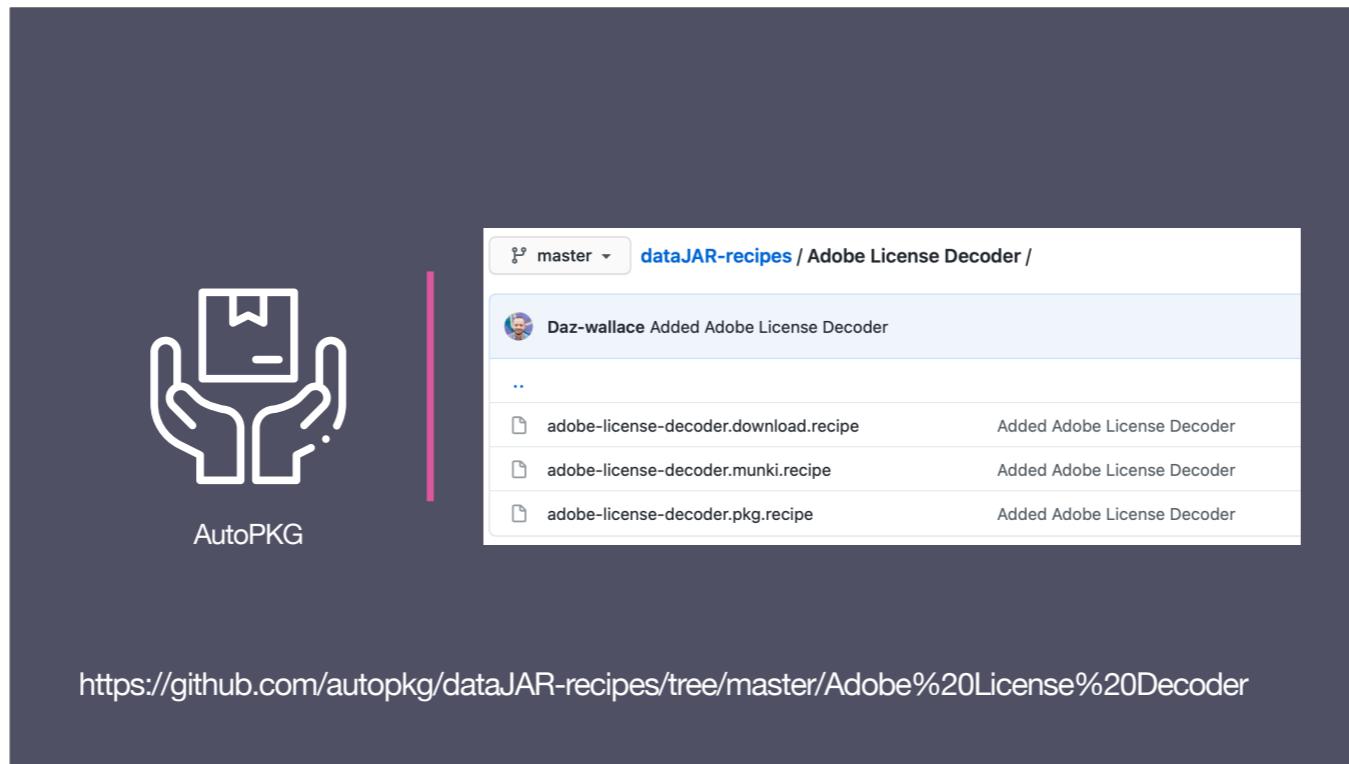
- the so-called "npdid" (also known as the "package id") of the license;
- whether the license is activated for the given user; and
- if activated, what the expiration date is of the license.

While this information is invaluable, it's specific to the user account it is run in, and it doesn't give any general information about the licenses that are installed on the machine that haven't been used.

Enter the `adobe-license-decoder`, a different command-line tool that can tell you about FRL and SDL license files both before and after installation. This tool can examine globally-installed SDL and FRL license files and tell you which apps they are for, which packages they are from, when they were installed, when they expire, and so on. It's like a "secret decoder ring" for the licenses!

<https://github.com/adobe/adobe-license-decoder.rs>

First of all (and the most obvious) is the GitHub page at the link above



Secondly we have an AutoPKG recipe in our dataJAR repo, linked above.



Blog Post

<https://dazwallace.wordpress.com/2021/03/06/adobe-license-decoder/>

Thanks for listening to me ramble on for a bit.

I've posted a link above to my blog post on the ALD, feel free to check it out as it has all the links and details mentioned in this talk.

Any questions, chuck them into the Q and A section and we'll get to them in the end.

Thanks!



One more thing...

<https://creativecloud.uservoice.com/forums/923263-apps-section-app-catalog-install-updates/suggestions/42883680-support-universal-macos-installer-that-works-on-bo>

So one last thing that's been discussed in the Adobe channel and I feel is worth raising here.

Adobe currently looks to have decided to supply separate Intel and Apple Silicon packages for their Apps (ignore the fact that most Apps are still Intel-only, the packages are still forcibly limiting the architecture type). If you need to support both Intel and Apple Silicon, this means you'll need to:

- 1) Double up on your packages that you need to store,
- 2) Add some logic into your deployment system to make sure the right items are going to the right devices.

If this sounds like a less than fun idea to you, please take a look at the link above and upvote the feature request