



Web



Signature lifecycle



WEB > GUIDES > PSPDFKIT SERVER > DIGITAL SIGNATURES > SIGNATURE LIFECYCLE

How to create a self-signed certificate for signatures



PSPDFKit Server has been deprecated and replaced by [Document Engine](#). To migrate to Document Engine and unlock advanced document processing capabilities, refer to our [migration guide](#). Learn more about these enhancements on our [blog](#).

To apply a digital signature to your document, you'll need a certificate and private key pair.

For testing purposes, you can use a self-signed certificate, but validating a document signed with a self-signed certificate may generate warnings due to the inherent lack of trust in self-signed certificates. It may result in a yellow (warning) bar when validating it with Nutrient (see our guide on how to [view and validate a digital signature](#) for more information) and third-party viewers.



For organizations seeking a hassle-free digital signing solution, leveraging cloud services is an ideal choice. [Nutrient's advanced eSignatures API](#) is a robust, cloud-based alternative that eliminates the need to implement most parts of the signing workflow. This service uses [Adobe Approved Trust List \(AATL\)](#) certificates, ensuring the highest level of trust and compliance.

Creating a self-signed certificate

The steps to create a self-signed certificate vary depending on your operating system.



ASK AI

Here's how to do it on macOS using the graphical user interface (GUI):

- 1 Search for Keychain Access in Spotlight by pressing **Command-Space bar**.
- 2 In the menu bar, click **Keychain Access**. Then choose **Certificate Assistant** and **Create a Certificate....**
- 3 Provide a name for the certificate, ensure that the **Identity Type** field is set to **Self-Signed Root**, and click **Create**.

This will generate a standard self-signed certificate using a secure 2048-RSA key.

To ensure that validation in Nutrient displays a green bar without warnings, you can create a certificate authority (CA) specifically for testing and place trust in it.

Self-signed certification authority and signing a certificate

More advanced configuration consists of creating a self-signed certificate authority and using it as a root certificate to create a signing certificate. In a system that trusts such a CA certificate, it isn't different from a certificate issued by a globally known root authority.

OpenSSL in macOS

Default OpenSSL configuration in macOS doesn't set relevant options for certification authority generation. One of the ways to solve this is by updating the system configuration.

To do this, add the following lines to `/etc/ssl/openssl.cnf`. For more information, refer to the community recommendations:

```
1 [ v3_ca ]
2 basicConstraints = critical,CA:TRUE
3 subjectKeyIdentifier = hash
4 authorityKeyIdentifier = keyid:always,issuer:always
```



Creating a certification authority

Generate a private key file named `test-ca.key` :

```
openssl genrsa -out test-ca.key 2048
```

Create and sign a certificate file named `test-ca.cert` for a CA with the common name (CN) `My Test CA v1` :

```
1 openssl req \  
2 -x509 -new -nodes -key test-ca.key \  
3 -subj "/CN=My Test CA v1" \  
4 -days 3650 -reqexts v3_req -extensions v3_ca \  
5 -out test-ca.cert
```

Creating a signing certificate

Generate a private key file named `test-signer.key` and a certificate signing request file named `test-signer.csr` with the CN `My Testing Document Signer` :

```
1 openssl req \  
2 -utf8 -nameopt oneline,utf8 -new -newkey rsa:2048 -nodes \  
3 -subj "/CN=My Testing Document Signer" \  
4 -keyout test-signer.key -out test-signer.csr
```

Create a signing certificate file from the request and name it `test-signer.cert` :

```
1 openssl x509 \  
2 -days 365 \  
3 -CA test-ca.cert -CAkey test-ca.key -CAcreateserial \  
4 -in test-signer.csr -req \  
5 -out test-signer.cert
```

Outcome

The process provides four important files:

- ⌘ `test-ca.cert` — A self-signed CA certificate (also the only component of the CA chain). This is what has to be trusted to accept child certificates.
- ⌘ `test-ca.key` — A self-signed CA private key that's necessary to sign more certificates by the same CA.
- ⌘ `test-signer.cert` and `test-signer.key` — A signer certificate and a private key used for signing in the signing service — for example, [our signing service reference implementation](#).

Obtaining a certificate from a trust service provider (TSP)

Obtaining a certificate from a trusted provider ensures the verification of the signer's identity. If you require a "trusted" certificate that's recognized worldwide, you can:

- ⌘ Use Nutrient's own [advanced eSignatures API](#) (recommended).
- ⌘ Purchase one from a trust service provider (TSP). We provide an integration with GlobalSign DSS, a qualified trust service provider that's part of the Adobe Approved Trust List and offers qualified certificates.

To choose a provider using the trusted list browser:

- 1 Visit the [trusted list browser](#).
- 2 Under **Qualified trust services**, select **Qualified certificate for electronic signature** and click **Next step**.
- 3 Choose a country and click **Search**.
- 4 Explore the listed TSPs, and by clicking on each provider's name, you can access detailed information about their services.

Note that Trusted Lists are published by each Member State, and the provided link will offer additional details about the TSP and the products they offer.

Was this helpful?



