



Web



Client authentication



WEB &gt; GUIDES &gt; PSPDFKIT SERVER &gt; CLIENT AUTHENTICATION

# Generate JWT for mobile authentication



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](#).

Our [Android](#) and [iOS](#) SDKs let you use your running PSPDFKit Server instance for converting Office documents to PDFs. This API also uses the [JWT](#) format for authentication, but it needs a different set of claims than our document API does. Keep the following in mind when generating a token for mobile conversion:

- ✧ It has to include the standard claim `"exp"`, which sets the deadline for the validity of the token. This needs to be a non-negative number using the Unix “Seconds Since the Epoch” timestamp format.
- ✧ It has to include the custom `"sha256"` claim, containing the SHA-256 of the Office file you’re planning to convert. This is used so that each token is only able to convert a single document.
- ✧ It has to be signed using an asymmetric cryptographic algorithm. PSPDFKit Server supports the algorithms RS256, RS512, ES256, and ES512. See [RFC 7518](#) for details about specific algorithms.


## Generating tokens

The following example shows the creation of a JWT in JavaScript using the `jsonwebtoken` lib

1 Create a key via `ssh-keygen` :



ASK AI




```
1 ssh-keygen -t rsa -b 4096 -f jwtRS256.key
2 # Enter your passphrase.
3
4 # Get the public key in PEM format:
5 openssl rsa -in jwtRS256.key -pubout -outform PEM -out jwtRS256_pub.pem
6
7 # If the above command fails because newer versions of `ssh-keygen` output
8 # convert the key to PEM like this and then repeat the `openssl` command.
9 ssh-keygen -p -m PEM -t rsa -b 4096 -f jwtRS256.key
10 openssl rsa -in jwtRS256.key -pubout -outform PEM -out jwtRS256_pub.pem
```

The private key ( `jwtRS256.key` ) is used to sign the tokens you hand out to the clients.

The public key ( `jwtRS256_pub.pem` ) needs to be added as a `JWT_PUBLIC_KEY` in the server's configuration so that the server will be able to validate the tokens' signatures but won't be able to create valid signatures. This example assumes you chose the `RS256` algorithm as the `JWT_ALGORITHM` in the server's configuration.


**i Note:** If you want to quickly test PSPDFKit for Web with your application, you can also use the key from our [example apps](#) (passphrase: `secret`). Make sure to change to a self-generated key before going into production.

- 2 Install the `jsonwebtoken` dependency:



```
npm install --save jsonwebtoken
```

- 3 Read the private key so that it can be used to sign JWTs. In the claims, pass the SHA-256 of the Office file you're planning to convert and an expiration. You can then use the produced token in your application:



```
1 const fs = require("fs");
2 const jwt = require("jsonwebtoken");
3 const key = fs.readFileSync("./jwtRS256.key");
4 const token = jwt.sign({sha256: "<office_file_sha>"}, key, {
5   algorithm: "RS256",
6   expiresIn: 60 * 60 // 1hr, this will set the `exp` claim for us.
7 });
```

---

Was this helpful?

✓ YES

✗ NO

Questions? [Contact us](#)

