



Web



Client authentication



WEB > GUIDES > PSPDFKIT SERVER > CLIENT AUTHENTICATION

Generate JSON Web Tokens for Document Engine



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](#).

JSON Web Tokens (JWTs) used for authentication by PSPDFKit Server can be generated with one of the many open source libraries that are available and listed on [jwt.io](#).

Token requirements

- ✧ 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 user-defined claims `"document_id"` and `"permissions"`.
 - ✧ `"document_id"` is the identifier of a document that has been previously uploaded to the server. This field has to be a string value.
 - ✧ `"permissions"` is a list of permission names that define which features will be accessible to the holder of this token. See the [Permissions](#) section below for more details.
- ✧ It may include the `"user_id"` claim, which will be stored on any annotation created, deleted by the user.



- ✧ It may include the `"layer"` claim, which defines the Instant layer that all changes will be persisted to.
- ✧ It may include the `"collaboration_permissions"` claim, which defines fine-grained permissions for actions allowed by individual users when multiple users are working on the same document. See the Collaboration Permissions guide for more details.
- ✧ It may include the `"default_group"` claim to control the group of created annotations, comments, and form fields.
- ✧ It may include the `"password"` claim, which defines the password to be used to open password-protected PDFs.
- ✧ It may include the `"creator_name"` claim, which ensures all annotations and comments are created with the specified creator name.
- ✧ 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.

Permissions

Available permissions:

- ✧ `"read-document"` — Required for viewing a document and its annotations. Without this permission, it won't be possible for the user to load the document and perform any operations on it.
- ✧ `"write"` — Required for creating, updating, and deleting annotations in a document. If this permission isn't present, PSPDFKit for Web will always be in read-only mode.
- ✧ `"download"` — Required for downloading and printing a document's PDF file.
- ✧ `"cover-image"` — Required for accessing the `/documents/cover` endpoint.

Any combination of the above permissions can be included in the `permissions` list when generating a JWT. Apart from that, the `permissions` field of a JWT may have one of the following special values:

- ✧ `"all-2017.3"` will enable all permissions available in the 2017.3 release, namely `"read-document"`, `"write"`, and `"download"`.
- ✧ `"all-2017.9"` will enable all permissions available in the 2017.9 release, namely `"read-document"`, `"write"`, `"download"`, and `"cover-image"`.
- ✧ `"all"` will enable all permissions supported by the running version of the server.

Generating tokens

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

- 1 Create a key via `ssh-keygen` :

```
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.

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 document ID, the set of permissions you want to have, 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 permissions = ["read-document", "write"];
5  const token = jwt.sign({document_id: "abc", permissions: permissions}, key, {
6    algorithm: "RS256",
7    expiresIn: 60 * 60 // 1hr, this will set the `exp` claim for us.
8  });
```

Was this helpful?

 YES

 NO

Questions? [Contact us](#)

