



Web



Migrate existing documents



WEB > GUIDES > PSPDFKIT SERVER > MIGRATE EXISTING DOCUMENTS

Migrate documents from Amazon S3 to 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](#).

You can migrate documents stored on S3 into PSPDFKit for Web Server-Backed using the remote document URL API. This enables Server to fetch the documents directly from S3 when required, which means you won't incur any additional storage costs.

For this to work correctly, you cannot use [signed URLs](#), since the URL needs to stay valid for the entire existence of the document. Furthermore, the file returned by the URL needs to match exactly with the file that's returned on the initial upload. If, at any point, the URL for a document becomes invalid, you'll have to delete the document and reupload it. There's no way to update the URL for an existing document.

If you're using signed URLs at the moment, we recommend the approaches outlined below.

Changing the S3 bucket policy

If you're using S3 with signed URLs, instead of providing PSPDFKit Server URLs with signed tokens and an expiration date, we recommend you change the [S3 bucket policy](#). By allowing the IP where Server is hosted, you won't need any signed URLs for extra protection, since PSPDFKit Server doesn't expose these URLs.



ASK AI

Having an internal endpoint that redirects

When you still want to use signed URLs or need to add credentials to the URLs that you provide to PSPDFKit Server, we recommend you add an internal endpoint on your backend that redirects the URLs to the signed URLs. In the following code snippet, we have a Node.js server that receives requests at `/documents/:document_id`, where `:document_id` is the identifier you use in your application. It then generates the signed URL or URLs with any other credentials and redirects to this URL.

In this example, you'd provide PSPDFKit Server the URL

`https://yourapp.com/documents/myDocumentIdentifier1`, and this endpoint would redirect to any signed URL, like `s3.amazonaws.com/myDocument.pdf?token=123455`:

```
1 // Catch the document endpoint.
2 app.get('/documents/:documentId', function (req, res) {
3   // Here we generate the signed URL for Amazon S3 for the unique document ID
4   // by PSPDFKit Server.
5   const preSignedUrl = generatePreSignedUrlForDocument(req.params.documentId);
6   res.redirect(preSignedUrl);
7 });
```

The `generatePreSignedUrlForDocument` function used in the snippet above will generate the signed S3 URL with the expiry parameters. This will look similar to the function in the [Java example](#) from the AWS guides.

Finally, you can either upload the documents on demand as described below, or run through all your documents one time and upload them all to Server in one go.

Uploading all documents

To upload all your documents, use the [API to add a document from a URL](#). You can call this with each of your documents' URLs to let Server know about all your documents. If your documents already have IDs your application knows about, you can also supply them here so you have one consistent ID everywhere:

HTTP

CURL

```
1 POST /api/documents
2 Content-Type: application/json
3 Authorization: Token token="<secret token>"
```

```
4
5 {
6   "url": "http://file.example.com/sample.pdf",
7   "document_id": "my_document_id_1"
8 }
```

Uploading documents on demand

When a user requests one of the documents, you can check on PSPDFKit Server to see if a document with this ID already exists. If this isn't the case, you can upload the document with the same ID your user used for the document. Otherwise, you can serve them the document immediately.

As an example, let's say you have a route like `/documents/:id` and your user requests a specific document with the ID `my_document_id_1`, which you internally have mapped to a URL.

Now you can use the [document info endpoint](#) on PSPDFKit Server, `GET /api/documents/my_document_id_1/document_info`, and see if the document already exists. If this isn't the case, you'll receive a 404 error and call the [adding a document from a URL endpoint](#):

HTTP

CURL

```
1 POST /api/documents
2 Content-Type: application/json
3 Authorization: Token token="<secret token>"
4
5 {
6   "url": "https://s3.amazon.com/my-bucket/sample.pdf",
7   "document_id": "my_document_id_1"
8 }
```



Was this helpful?

✓ YES

✗ NO

