




Java



From local storage

[HOME](#)  [GUIDES](#)  [JAVA](#)  [FEATURES](#)  [FROM LOCAL STORAGE](#)

# Open local PDF files using Java



COPY PAGE



This guide covers how to open and save PDFs with Nutrient Java SDK. We will cover some simple examples and give an overview of saving options.

## Opening PDFs

The simplest input source for opening a document is a file path. To open a PdfDocument from a file path, first you will require a Java File. In order for PdfDocument to read from the File, you're required to wrap the File in a FileDataProvider, and the resulting object can then be passed:

```
final File file = new File("document.pdf");  
PdfDocument document = new PdfDocument(new FileDataProvider(file));
```

To ensure maximum flexibility of document sources, Nutrient comes with an extensible class to provide a custom data provider.

Writing a data provider inheriting from DataProvider will allow you to provide data from any source possible (e.g. cloud providers, in-memory, encrypted sources).



ASK AI

# Saving PDFs

Saving a document is as simple as calling `save`, but there are more options for finer-grained control:

```
document.save(new DocumentSaveOptions.Builder().build());
```

## Incremental saving

By default, Nutrient saves to files incrementally. This is a feature of PDFs and means that when saving, the initial content of the original document is not altered and all changes are appended to the file. This can result in a significantly faster save when working with a large file, as changes are typically quite small.

However, you should bear in mind that since the changes are always appended, the file will constantly increase in size with every save, regardless of the changes being made. Since this is sometimes undesirable, you can specify that the PDF should be completely rewritten upon saving.

Here is an example of saving incrementally:

```
document.save(new DocumentSaveOptions.Builder().incrementalSave(true).build());
```

Here is how to rewrite the PDF when saving:

```
document.save(new DocumentSaveOptions.Builder().incrementalSave(false).build());
```

Again, to ensure maximum flexibility of document sources, Nutrient comes with an extensible class to provide a custom data provider that is writable.

Writing a data provider inheriting from `WriteableDataProvider` will allow you to sink data to any source possible (e.g. cloud providers, in-memory, encrypted sources).

---

## Was this helpful?

✓ YES

✗ NO



Copyright 2025 Nutrient. All rights reserved.