

# OCR and Document Understanding

## Document OCR processor

The Document OCR (Optical Character Recognition) processor, powered by our latest OCR model `mistral-ocr-latest`, enables you to extract text and structured content from PDF documents.

### Key features:

- Extracts text content while maintaining document structure and hierarchy
- Preserves formatting like headers, paragraphs, lists and tables
- Returns results in markdown format for easy parsing and rendering
- Handles complex layouts including multi-column text and mixed content
- Processes documents at scale with high accuracy
- Supports multiple document formats including PDF, images, and uploaded documents

The OCR processor returns both the extracted text content and metadata about the document structure, making it easy to work with the recognized content programmatically.

## OCR with PDF

**python**   **typescript**   **curl**

```
import { Mistral } from '@mistralai/mistralai';

const apiKey = process.env.MISTRAL_API_KEY;
const client = new Mistral({apiKey: apiKey});

const ocrResponse = await client.ocr.process({
  model: "mistral-ocr-latest",
  document: {
    type: "document_url",
    documentUrl: "https://arxiv.org/pdf/2201.04234"
  },
  include_image_base64: true
});
```

► **Example output:**

## OCR with uploaded PDF

You can also upload a PDF file and get the OCR results from the uploaded PDF.

### Upload a file

**python**   **typescript**   **curl**

```
import { Mistral } from '@mistralai/mistralai';
import fs from 'fs';

const apiKey = process.env.MISTRAL_API_KEY;

const client = new Mistral({apiKey: apiKey});

const uploaded_file = fs.readFileSync('uploaded_file.pdf');
const uploaded_pdf = await client.files.upload({
  file: {
    fileName: "uploaded_file.pdf",
    content: uploaded_file,
  },
  purpose: "ocr"
});
```

### Retrieve File

**python**   **typescript**   **curl**

```
await client.files.retrieve({
  fileId: uploaded_pdf.id
});
```

```
id='00edaf84-95b0-45db-8f83-f71138491f23' object='file' size_bytes=3749788
created_at=1741023462 filename='uploaded_file.pdf' purpose='ocr'
sample_type='ocr_input' source='upload' deleted=False num_lines=None
```

### Get signed URL

**python**   **typescript**   **curl**

```
signed_url = client.files.get_signed_url(file_id=uploaded_pdf.id)
```

## Get OCR results

**python**   **typescript**   **curl**

```
import { Mistral } from '@mistralai/mistralai';

const apiKey = process.env.MISTRAL_API_KEY;
const client = new Mistral({apiKey: apiKey});

const ocrResponse = await client.ocr.process({
  model: "mistral-ocr-latest",
  document: {
    type: "document_url",
    documentUrl: signedUrl.url,
  }
});
```

## OCR with image

**python**   **typescript**   **curl**

```
import { Mistral } from '@mistralai/mistralai';

const apiKey = process.env.MISTRAL_API_KEY;
const client = new Mistral({apiKey: apiKey});

const ocrResponse = await client.ocr.process({
  model: "mistral-ocr-latest",
  document: {
    type: "image_url",
    imageUrl: "https://media-cldnry.s-nbcnews.com/image/upload/t_fit-560w,f_avif,q_auto:eco,dpr_2/rockcms/2023-11/short-quotes-sw1-231117-02-33d404.jpg",
  }
});
```

# Document understanding

The Document understanding capability combines OCR with large language model capabilities to enable natural language interaction with document content. This allows you to extract information and insights from documents by asking questions in natural language.

## The workflow consists of two main steps:

1. Document Processing: OCR extracts text, structure, and formatting, creating a machine-readable version of the document.
2. Language Model Understanding: The extracted document content is analyzed by a large language model. You can ask questions or request information in natural language. The model understands context and relationships within the document and can provide relevant answers based on the document content.

## Key capabilities:

- Question answering about specific document content
- Information extraction and summarization
- Document analysis and insights
- Multi-document queries and comparisons
- Context-aware responses that consider the full document

## Common use cases:

- Analyzing research papers and technical documents
- Extracting information from business documents
- Processing legal documents and contracts
- Building document Q&A applications
- Automating document-based workflows

The examples below show how to interact with a PDF document using natural language:

**python**    **typescript**    **curl**

```
import { Mistral } from "@mistralai/mistralai";

const apiKey = process.env["MISTRAL_API_KEY"];
```

```
const client = new Mistral({
  apiKey: apiKey,
});

const chatResponse = await client.chat.complete({
  model: "mistral-small-latest",
  messages: [
    {
      role: "user",
      content: [
        {
          type: "text",
          text: "what is the last sentence in the document",
        },
        {
          type: "document_url",
          documentUrl: "https://arxiv.org/pdf/1805.04770",
        },
      ],
    },
  ],
});

console.log("JSON:", chatResponse.choices[0].message.content);
```

## Cookbooks

For more information and guides on how to make use of OCR and leverage document understanding, we have the following cookbooks:

- [Tool Use and Document Understanding](#)
- [Batch OCR](#)
- [Structured OCR](#)