Documentation: Custom PDF Generation REST API

**Summary**

**DOCX to PDF REST API – Project Overview**

This document describes the design, technology stack, and architecture of a REST-based service for converting Microsoft Word .docx files to PDF format in a local development environment, accessible externally via ngrok.

**Goal:** Build a lightweight, REST API for converting DOCX to PDF using native tools on macOS, which can also be accessed externally for integration testing (e.g. with ServiceNow).

**Why:**

* Many enterprise systems (e.g. ServiceNow) generate or handle DOCX files
* Converting to PDF ensures consistency, security, and read-only distribution

**Key Benefits:**

* Fully API-driven and minimalistic
* Easy to set up on any environment (example – macOS) with Python and LibreOffice
* No need for cloud hosting or Docker to run and test
* Externally accessible using ngrok
* Fast and reliable
* No file storage
* Easy to integrate into ServiceNow workflows

**Technology Stack**

**Detailed Overview**

The solution has been built with a carefully selected technology stack that ensures reliability, performance, maintainability, and security. Here's a breakdown of each component and its purpose:

**Python 3 + Flask**

* Python is a powerful, high-level programming language known for its readability and wide adoption in enterprise automation, scripting, and backend services.
* Flask is a lightweight, minimalistic web framework for Python that’s ideal for building fast and efficient REST APIs.
* Flask’s makes this perfect for fast local dev cycles
* Flask is used to expose the /convert/docx/to/pdf POST endpoint
* Accepts binary .docx content and streams back PDF output
* Used to handle:
  + File upload via POST method
  + Interfacing with the document conversion layer
  + Streaming PDF output as a response
* Flask keeps the application lean and avoids unnecessary complexity or overhead

**LibreOffice (Headless Mode)**

* LibreOffice is a powerful open-source office suite that supports a wide variety of document formats.
* Installed natively on environment (example – macOS) - (/Applications/LibreOffice.app/...)
* In this implementation, LibreOffice is run in headless mode, which means:
  + No graphical user interface (GUI) is launched
  + All conversions are performed via command line (libreoffice --headless)
  + Ideal for server environments where no desktop is available
* It ensures high-accuracy and quality conversion of .docx documents to .pdf with layout, fonts, images, and formatting preserved — without requiring Microsoft Word or any software.

**Ngrok**

* Lightweight tunneling tool to expose local server via public HTTPS endpoint
* Enables testing integrations (like ServiceNow) without public hosting
* Used via: ngrok http http://127.0.0.1:5000

**Summary of Technology Stack**

|  |  |
| --- | --- |
| **Component** | **Role Architecture** |
| Python + Flask | Handles file uploads and REST interface |
| LibreOffice | Performs DOCX to PDF conversion in headless mode |
| ngrok | Exposes local Flask server to the internet for external access |
| Local environmnrt (example – macOS) | Local development environment |

**Solution Architecture**

**Request Flow:**

* Receives a .docx file via POST method (Content-Type: application/octet-stream)
* The file is temporarily saved on disk (in /uploads)
* Uses LibreOffice (headless mode) to convert .docx → .pdf
* Streams the PDF back to the client (via send\_file)
* Deletes both files immediately after conversion from the disk (in /uploads) using a finally block

**Files involved:**

* app.py: Flask app handling the logic
* requirements.txt: Python dependencies (Flask)

**Security & Data Privacy**

This API is designed for sensitive documents and enterprise use cases:

* Files are not saved and stored permanently
* All files are removed in a finally block after processing
* No content is logged or retained
* No third-party dependencies or logging services are used

**Integration with ServiceNow**

**How it's used:**

* ServiceNow sends .docx file via REST (using sn\_ws.RESTMessageV2)
* The public ngrok HTTPS URL is used as the API endpoint
* API returns the converted .pdf file
* PDF is attached back to the target record (e.g. incident, HR case)

**Use cases:**

* Outbound REST Messages
* Business Rules
* Script Include
* Scheduled Jobs
* Flow Designer custom steps
* UI Actions or Workflows

**Local Development & Execution**

**Prerequisites:**

* Python 3.7+
* LibreOffice installed on environment (example – macOS)
* brew for installing ngrok
* Postman or curl for testing

**Setup & Execution:**

# Create new project folder

mkdir -p ~/my-api-localhost && cd ~/my-api-localhost

# Create virtual environment

python3 -m venv venv

source venv/bin/activate

# Create requirements.txt and install Flask

echo "Flask" > requirements.txt

pip install -r requirements.txt

# Create uploads folder

mkdir uploads

# Run the Flask API

python app.py

**Exposing the API with ngrok**

brew install ngrok

ngrok config add-authtoken <your\_token>

ngrok http <http://127.0.0.1:5000>

This gives you a URL like:

<https://abcd1234.ngrok-free.app>

Use this in Postman or from ServiceNow REST calls.