Script Pad User Guide (Chinese Only)

Script Pad is an API-Based, Optical Character Recognition (OCR) and language translation pipeline capable of performing the following tasks:

- Process One or More Images *(PNGs, JPEGs, JPG)*
- Pre-Process Assets to Improve the Quality of OCR
- Identify Boxes of Foreign Language Text
- Translate All Identified Foreign Language Text

The platform uses the latest HuggingFace Machine Learning Translation Models for **Chinese Simplified** and is readily compatible with other languages.

Requirements

- Python 3.11+
- docker-compose
- docker.io

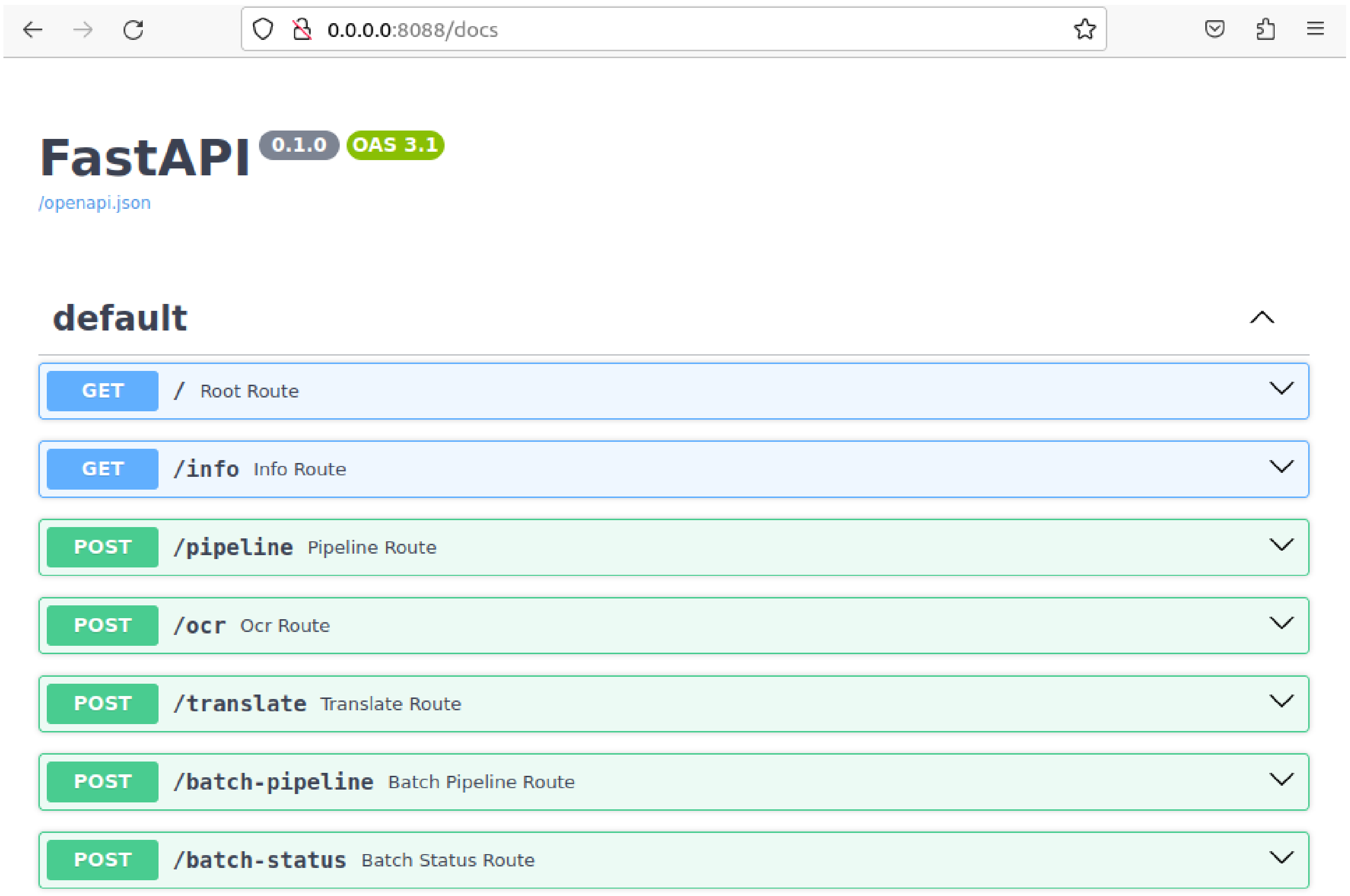
Table of Contents

Quick Setup

On any platform with `docker-compose` installed, open the terminal:

```
tar -xzvf script-pad_v1.0.0.tar.gz
tar -xzvf models.tar.gz
docker load --input svc-prod_online.tar.gz
docker-compose up
```

If start-up was successful, the API will instantly be active in the background with documentation hosted locally **offline** and viewable at the generated /docs endpoint.


0.0.0.0:8088/docs

FastAPI 0.1.0 OAS 3.1
/openapi.json

default

GET / Root Route
GET /info Info Route
POST /pipeline Pipeline Route
POST /ocr Ocr Route
POST /translate Translate Route
POST /batch-pipeline Batch Pipeline Route
POST /batch-status Batch Status Route

Endpoint Descriptions

Documentation of expected data types and returns are viewable at the generated /docs endpoint. However, as a quick reference the expected HTTP POST Request formats for each endpoint can be found in the Appendix.

The following describes the purpose and use-case of each endpoint.

User Note

As an input, the current version of the API by default expects images *(PNGs, JPEGs, JPG)* in base64 format.

To support PDFs, users need to use their API client-side script to split PDFs into individual images. This is readily supported in the open domain with Python libraries such as `pdf2image`. If you need assistance, contact the application developers.

/pipeline

Asynchronous endpoint capable of receiving simultaneous requests and processing them concurrently to perform **OCR** and **Translation**. Each individual incoming request is expected to have **only one document or image** provided.

Target Use Case: A user has a large volume of media that must be processed **concurrently** to expedite OCR and translation speeds. The user writes an API client script to send multiple files in individual requests.

Constraint: Only limited by the available CPU cores of the API server and the number of workers (defined in the `docker-compose.yml`, see Configuration) to maximize those cores. Requires API clients capable of handling asynchronous requests and responses (*can be provided upon request*). Users need to be able to handle scenarios where all workers are busy.

/translate

Synchronous endpoint which only translates raw foreign language text and does not OCR.

Target Use Case: A user possesses foreign language in a plaintext form and wishes to quickly receive a rapid offline translation.

Constraint: N/A

/batch-status

An informational endpoint which returns the status of jobs being worked on by the `/batch-pipeline` endpoint

/ocr

Synchronous endpoint which **only performs OCR** of media and **does not translate** content.

Target Use Case: (a) Users who are fluent in a foreign language or (b) Users who wish to leverage OCR text extraction for their own translation pipeline(s).

Constraint: N/A

/batch-pipeline

Asynchronous endpoint which will accept a single request with multiple files, processing them quietly in the background **one at a time**, keeping track of completions, and writing the generated translations of OCR'd content to the `/data` directory (defined in the `docker-compose.yml`, see Configuration). Uses a SQLite database to track job status.

Target Use Case: A user has a large volume of media to process in a "set and forget" manner (i.e., not sensitive to time). The user wishes to periodically check-in on progress and receive files as they're completed.

Constraint: Extends processing times but is friendlier to simpler API clients (*can be provided upon request*), reducing development burden on the user. Constrained by memory available to the host system.

Informational Endpoints

The root `/` endpoint is used to validate the API is running successfully while the `/info` endpoint is used to provide information on the currently loaded translation models.

Configuration

The `docker-compose.yml` provides defines a number of environmental variables, see below for definitions. Only `WORKER_COUNT` (and optionally `INIT_PORT`) should typically ever require modification.

- `MODEL_DIR`: Input directory containing the translation models
- `DATA_DIR`: Output directory used by the /batch-pipeline endpoint to write JSON results from OCR and translation
- `INIT_LANG`: The foreign langauage the API should expect (corresponds to the languages supported by the translation models)
- `INIT_PORT`: The local network port to run the API
- `WORKER_COUNT`: The number of concurrent workers to stand-up. Used by the `/pipeline` endpoint to distribute multiple requests for parallel processing of documents. Leverage this setting to vertically scale the API.

> The recommended maximum number of workers for `WORKER_COUNT` is generally (Number of Cores x 2 + 1)

```
version: "3"
services:
  .build:
    image: svc-prod:online
    build:
      context: .
      dockerfile: Dockerfile

  svc-chi:
    image: svc-prod:online
    container_name: svc-chi
    environment:
      MODEL_DIR: "/models"
      DATA_DIR: "/data"
      INIT_LANG: "chinese"
      INIT_PORT: 8088
      WORKER_COUNT: 8
    volumes:
      - ./models:/models:ro
    ports:
      - 8088:8088
    command: "python3 /opt/app/src/scriptpad/script_pad.py"
    depends_on:
      - .build
```

Appendix

Snapshots of the expected HTTP POST Request formats for each endpoint in Endpoint Descriptions.

Source: `models/__init__.py`

/pipeline

```
{
  "name": "string",
  "b64data": "string",
  "src_lang": "string",
  "image_type": "string",
  "metadata": {
    "additionalProp1": "string",
    "additionalProp2": "string",
    "additionalProp3": "string"
  },
  "box_scale": 0,
  "density_scale": 0,
  "internal_id": "string",
  "dst_lang": "string",
  "overlay": "False" // Always Set to False
}
```

/batch-pipeline

```
{
  "images": [
    {
      "name": "string",
      "b64data": "string",
      "src_lang": "string",
      "image_type": "string",
      "metadata": {
        "additionalProp1": "string",
        "additionalProp2": "string",
        "additionalProp3": "string"
      },
      "box_scale": 0,
      "density_scale": 0,
      "internal_id": "string",
      "dst_lang": "string",
      "overlay": "False" // Always Set to False
    }
  ]
}
```

/batch-status

```
{
  "internal_id": "string"
}
```

/translate

```
{
  "text": "string",
  "src_lang": "string",
  "dst_lang": "string"
}
```

/ocr

```
{
  "name": "string",
  "b64data": "string",
  "src_lang": "string",
  "image_type": "string",
  "metadata": {
    "additionalProp1": "string",
    "additionalProp2": "string",
    "additionalProp3": "string"
  },
  "box_scale": 0,
  "density_scale": 0,
  "internal_id": "string"
}
```