DiffRhythm Music Generation Server API Documentation

Version: 1.0
Date: January 2025
Server Port: Default 8000 (configurable)
Base URL: `http://localhost:8000` (or your deployed address)

This is a simple, single-concurrent-job HTTP server for music generation using a diffusion-based model (DiffRhythm / ASLP-lab style).
Only **one generation** can run at a time — additional requests are rejected with 503.

Endpoints

Method	Endpoint	Description	Response Codes
`POST`	`/generate`	Start a new music generation job	202, 400, 503
`GET`	`/output/{filename}.wav`	Get generated audio file or status information	200, 404
`HEAD`	`/output/{filename}.wav`	Same as GET but only headers (status + metadata)	200, 404

1. Start Generation

`POST /generate`

Multipart/form-data request to start music generation.

Form Fields (all optional except where noted)

Field	Type	Required	Description	Example / Format
`lrc`	text	No	Lyrics in LRC format (or plain text)	`[00:00.00] Hello world...`
`ref_prompt`	text	*	Text description of desired style / genre	`dreamy synthwave, 80s retro, warm pads`
`ref_audio`	file	*	Reference audio file (.wav) to extract style from	Upload .wav file

Field	Type	Required	Description	Example / Format
`audio_length`	integer	Yes	Desired song length in seconds (95 or 96–285)	`120`
`chunked`	boolean	No	Use chunked decoding (default: false)	`true` / `false`
`edit`	boolean	No	Enable edit mode (requires `ref_song` + `edit_segments`)	`true` / `false`
`ref_song`	text	No*	Reference song identifier/path for editing (edit mode)	Path or identifier
`edit_segments`	text	No*	JSON-like segments to edit: `[[start1,end1], [start2,end2]]` (-1 = start/end)	`[[-1,30], [90,-1]]`
`batch_infer_num`	integer	No	How many variants to generate internally (only one is saved)	`1` – `4` (default: 1)
`output_dir`	text	No	Custom output directory (default: `infer/example/output`)	`/path/to/folder`

Note: Either `ref_prompt` **or** `ref_audio` file must be provided (not both).

Success Response (202 Accepted)

```json
{
  "status": "accepted",
  "url": "/output/8f4b9c2a7d1e3f5a.wav",
  "job_id": "a1b2c3d4e5f6g7h8i9j0",
  "message": "Generation started. Check the URL later for the result."
}
```

Error Responses

- **400 Bad Request** — Invalid parameters, missing required fields, wrong Content-Type
- **503 Service Unavailable** — Server is busy (another generation is running)

2. Get Audio or Status

`GET /output/{filename}.wav`

`HEAD /output/{filename}.wav`

Returns the generated WAV file if ready, or **JSON status** information if still processing.

When file exists (ready)

- **GET**: Returns the audio/wav file
- **HEAD**: Returns headers only (Content-Type: audio/wav, Content-Length, etc.)

When file does not exist yet (but job is running)

Both GET and HEAD return **200 OK** with JSON:

```
JSON

  {
    "status": "processing",
    "job_id": "a1b2c3d4e5f6g7h8i9j0",
    "file": "8f4b9c2a7d1e3f5a.wav",
    "elapsed_time": "42.3 seconds",
    "estimated_remaining": "68.7 seconds",
    "progress_percent": 38,
    "parameters": {
      "lrc": "[00:00.00] Hello...\n...",
      "ref_prompt": "dreamy synthwave...",
      "audio_length": 120,
      ...
    },
    "server_busy": true,
    "other_goodies": {
      "device": "cuda",
      "max_frames": 6144,
      "current_time": "2026-01-15 22:45:12 UTC",
      "uptime": "14523.4 seconds"
    }
  }
```

When file does not exist and no active job for it

```
JSON

  {
    "status": "not_found",
    "message": "No active job for this file or file does not exist."
  }
```

→ HTTP 404 in this case

Quick Client Example (Python)

```
Python

  import requests
  import time

  # Start generation
  files = {"_dummy": (None, "")}  # forces multipart/form-data
  data = {
      "ref_prompt": "upbeat electronic dance",
      "audio_length": "120",
      "lrc": "[00:00.00] Let's dance..."
  }

  r = requests.post("http://localhost:8000/generate", data=data, files=files)
  info = r.json()
  url = info["url"]

  # Poll status
  while True:
      resp = requests.get(f"http://localhost:8000{url}")
      if resp.headers.get("content-type") == "audio/wav":
          with open("my_song.wav", "wb") as f:
              f.write(resp.content)
          print("Downloaded!")
          break
      else:
          status = resp.json()
          print(f"Progress: {status.get('progress_percent', '?')}%")
          time.sleep(5)
```

Features Summary

- Single concurrent generation (others get 503)

- Immediate 202 Accepted with future download URL
- Real-time progress via GET/HEAD on the same URL
- Supports both text style prompt and reference audio upload
- Basic edit mode support
- GPU-aware (cuda/mps/cpu)
- Clean shutdown on Ctrl+C

Happy generating! 🎵