# Blog-to-Podcast API Reference

## Base URL

```
http://localhost:8000
```

## Authentication

Currently, no authentication is required for local development. For production, implement token-based authentication.

## Endpoints

### 1. Create Conversion Task

**Endpoint:** `POST /convert/`

**Description:** Initiates a new blog-to-podcast conversion task.

**Request Body:**

```
{
 "url": "https://example.com/blog-post"
}
```

**Response:** `201 Created`

```
{
 "task_id": "550e8400-e29b-41d4-a716-446655440000",
 "status": "PENDING",
 "url": "https://example.com/blog-post",
 "created_at": "2025-12-01T01:00:00Z"
}
```

**Error Responses:** - `400 Bad Request`: Invalid URL format - `500 Internal Server Error`: Server error

**Example:**

```
curl -X POST http://localhost:8000/convert/ \
 -H "Content-Type: application/json" \
 -d '{"url": "https://example.com/blog-post"}'
```

---

## 2. Get Task Status

**Endpoint:** `GET /status/{task_id}/`

**Description:** Retrieves the current status and progress of a conversion task.

**Path Parameters:** - `task_id` (UUID): The unique identifier of the task

**Response:** `200 OK`

```
{
 "task_id": "550e8400-e29b-41d4-a716-446655440000",
 "status": "PROCESSING",
 "progress": 65,
 "current_step": "Generating multi-speaker audio...",
 "url": "https://example.com/blog-post",
 "audio_file": null,
 "video_file": null,
 "error_message": null,
 "created_at": "2025-12-01T01:00:00Z"
}
```

**Status Values:** - `PENDING`: Task queued, not started - `PROCESSING`: Currently being processed - `COMPLETED`: Successfully completed - `FAILED`: Processing failed

**Completed Task Response:**

```
{
 "task_id": "550e8400-e29b-41d4-a716-446655440000",
 "status": "COMPLETED",
 "progress": 100,
 "current_step": "Completed",
 "url": "https://example.com/blog-post",
 "audio_file": "/media/podcast_550e8400-e29b-41d4-a716-446655440000.mp3",
 "video_file": "/media/podcast_550e8400-e29b-41d4-a716-446655440000.mp4",
 "script": "Host A: Welcome to today's podcast...",
 "error_message": null,
 "created_at": "2025-12-01T01:00:00Z"
}
```

**Error Responses:** - `404 Not Found`: Task ID doesn't exist

**Example:**

```
curl http://localhost:8000/status/550e8400-e29b-41d4-a716-446655440000/
```

---

## 3. Get Task Logs

**Endpoint:** `GET /logs/{task_id}/`

**Description:** Retrieves detailed logs for a conversion task.

**Path Parameters:** - `task_id` (UUID): The unique identifier of the task

**Response:** `200 OK`

```
{
 "task_id": "550e8400-e29b-41d4-a716-446655440000",
 "logs": "[10:30:15] Extracting content from blog...\n[10:30:45] Generating podcast script with
}
```

**Example:**

```
curl http://localhost:8000/logs/550e8400-e29b-41d4-a716-446655440000/
```

---

## 4. Download Media Files

**Endpoint:** `GET /media/{filename}`

**Description:** Downloads generated audio or video files.

**Path Parameters:** - `filename`: The media file name (from task response)

**Response:** `200 OK` (Binary file)

**Example:**

```
# Download audio
curl -O http://localhost:8000/media/podcast_550e8400.mp3
```

```
# Download video
curl -O http://localhost:8000/media/podcast_550e8400.mp4
```

---

# Data Models

## ConversionTask

```
{
 "id": UUID,
 "url": String,
 "status": Enum["PENDING", "PROCESSING", "COMPLETED", "FAILED"],
 "progress": Integer (0-100),
 "current_step": String,
 "script": String (nullable),
 "audio_file": String (nullable),
 "video_file": String (nullable),
 "error_message": String (nullable),
 "logs": String,
 "timing_map": Array (nullable),
 "created_at": DateTime
}
```

## Timing Map Entry

```
{
 "start": Float,      # Start time in seconds
 "end": Float,        # End time in seconds
 "text": String,      # Dialogue text
 "speaker": String    # "Host A" or "Host B"
}
```

---

# Error Handling

All errors follow this format:

```
{
 "error": "Error message description",
 "status_code": 400
```

```
}
```

**Common Error Codes:** - `400`: Bad Request (invalid input) - `404`: Not Found (resource doesn't exist) - `500`: Internal Server Error (processing failure)

---

## Rate Limiting

**Current Limits:** - Max concurrent tasks: 5 - No per-user rate limiting (local development)

**Production Recommendations:** - Implement per-IP rate limiting - Add authentication tokens - Set max requests per hour/day

---

## Webhooks (Future)

Planned feature for production:

```
{
 "webhook_url": "https://your-app.com/webhook",
 "events": ["task.completed", "task.failed"]
}
```

---

## Code Examples

### Python

```
import requests

# Create task
response = requests.post(
   'http://localhost:8000/convert/',
   json={'url': 'https://example.com/blog'}
)
task = response.json()
task_id = task['task_id']

# Poll status
import time
```

```
while True:
   status = requests.get(f'http://localhost:8000/status/{task_id}/').json()
   if status['status'] in ['COMPLETED', 'FAILED']:
       break
   print(f"Progress: {status['progress']}%")
   time.sleep(5)

# Download files
if status['status'] == 'COMPLETED':
   audio = requests.get(f"http://localhost:8000{status['audio_file']}")
   with open('podcast.mp3', 'wb') as f:
       f.write(audio.content)
```

### JavaScript

```
// Create task
const response = await fetch('http://localhost:8000/convert/', {
 method: 'POST',
 headers: {'Content-Type': 'application/json'},
 body: JSON.stringify({url: 'https://example.com/blog'})
});
const task = await response.json();

// Poll status
const checkStatus = async () => {
 const status = await fetch(`http://localhost:8000/status/${task.task_id}/`);
 const data = await status.json();

 if (data.status === 'COMPLETED') {
   console.log('Download:', data.audio_file);
 } else if (data.status === 'PROCESSING') {
   setTimeout(checkStatus, 5000);
 }
};
checkStatus();
```

---

**Version**: 1.0

**Last Updated**: December 2025