COPLAND

기술 문서

COPLAND TTS

API 개발자 매뉴얼

HTTP API를 직접 호출하는 경우의 문서입니다. 요청 필드, 스트리밍, 읽기 규칙, SSML을 예제와 함께 설명합니다.

적용 버전

COPLAND TTS 0.3.4(게이트웨이)
Java SDK 0.2.1

문서 갱신일

2026-10-12

copland.ai/tts

차례

1. 시작하기
   1.1 기본 주소
   1.2 첫 호출 (curl)
   1.3 OpenAI 호환
   1.4 API 한눈에 보기
2. 인증
   2.1 키 보내기
   2.2 키 받기
   2.3 키가 필요 없는 경로
3. 음성 합성 — POST /v1/audio/speech
   3.1 요청 필드
   3.2 화자와 모델
   3.3 응답 형식
   3.4 응답 헤더
   3.5 스트리밍 WAV
   3.6 한 번에 받기 (stream: false)
   3.7 시드와 결과 고정
   3.8 오류가 나는 때
4. 긴 글과 재생
   4.1 문장 분할
   4.2 첫 소리 시간
   4.3 문장 사이 쉼
5. 읽기 규칙 (정규화)
   5.1 적용 순서
   5.2 규칙표
   5.3 사용자 사전
   5.4 규칙이 맞지 않을 때
6. SSML
   6.1 지원 태그
   6.2 예시
   6.3 SSML 오류
7. 그 밖의 엔드포인트
   7.1 POST /v1/normalize — 읽기 미리보기
   7.2 GET /v1/voices — 화자 목록

7.3 GET /v1/models — 모델 목록
7.4 GET /v1/dictionaries — 사전 목록
7.5 GET /v1/license — 라이선스와 채널
7.6 GET /v1/usage — 키별 사용량
7.7 POST /v1/usage/reset — 사용량 초기화
7.8 POST /v1/dictionary/reload — 사전 다시 읽기
7.9 POST /v1/keys/reload — API 키 다시 읽기
7.10 GET /health — 상태 확인
7.11 GET /metrics — 운영 지표
7.12 GET /openapi.json — OpenAPI 명세
8. 오류
8.1 형식
8.2 오류 코드 전체
8.3 검사 순서
9. 한도와 남용 방지
9.1 동시 채널 (라이선스)
9.2 과부하
9.3 키별 하루 한도
9.4 입력 길이
10. 재시도와 장애 전환
10.1 다시 보내는 경우
10.2 다시 보내지 않는 경우
10.3 소리를 받기 시작한 뒤에는 다시 보내지 않습니다
10.4 openai 패키지를 쓸 때
11. 언어별 예제
11.1 curl
11.2 Python (httpx, 스트리밍 재생·저장)
11.3 JavaScript (브라우저, Web Audio로 PCM 스트리밍 재생)
11.4 Node.js (서버에서 중계)
11.5 Java
12. 변경 내역 (API 관련)

- 적용 버전: **COPLAND TTS 0.3.4**(게이트웨이), Java SDK 0.2.1
- 문서 갱신일: 2026-10-12
- 독자: COPLAND TTS 게이트웨이를 부르는 웹 서버·애플리케이션 개발자

이 문서는 게이트웨이의 HTTP API 전체를 설명합니다. 언어에 상관없이 이 문서만으로 연동을 끝낼 수 있게 썼습니다. Java를 쓰면 Java SDK 매뉴얼의 SDK를 쓰는 편이 쉽습니다. SDK가 장애 전환과 재시도를 대신 처리합니다.

이 문서의 응답 예시는 모두 실제 게이트웨이 0.3.4에서 받은 응답입니다. API 키는 `ctts_xxxxxxxx…` 같은 가짜 값으로 바꿔 적었습니다. 고객사 이름, 장비키, 파일 경로처럼 설치마다 다른 값도 예시 값으로 바꿨습니다.

1. 시작하기

1.1 기본 주소

용도	주소	API 키	비고
고객사 설치 서버	`http://<서버>:9300`	필요	포트는 설치 때 정합니다 (`TTS_PORT`, 기본 9300)
시험 엔드포인트	`https://copland.ai/tts/sandbox`	필요 없음	IP당 하루 100회, 공개 API만

- 모든 API 경로는 기본 주소 뒤에 붙입니다. 예: `http://10.0.0.11:9300/v1/audio/speech`, `https://copland.ai/tts/sandbox/v1/audio/speech`
- **시험 엔드포인트**는 자사 서버를 설치하기 전에 연동을 미리 시험하는 곳입니다.
  - 쓸 수 있는 경로: `/v1/audio/speech`, `/v1/normalize`, `/v1/voices`, `/v1/models`, `/health`. 나머지 경로는 404입니다.
  - 합성(`/v1/audio/speech`)은 접속 IP마다 하루 100회입니다. 짧은 시간에 몰아 보내는 요청도 막힐 수 있습니다.
  - 시연용 소형 장비(AMX 없음)라서 첫 소리 시간과 동시 처리 성능은 실제 설치 장비와 다릅니다. 성능은 설치한 서버에서 잽니다.
  - 입력한 문장은 저장하지 않습니다. 그래도 실제 고객 정보는 넣지 마세요.
  - 서버 간 호출(curl, Python, Java, Node.js)용입니다. CORS 헤더를 보내지 않으므로 다른 사이트의 브라우저 스크립트에서는 부를 수 없습니다.
- 온라인 데모는 https://copland.ai/tts/demo/ 입니다. 같은 시험 엔드포인트를 브라우저에서 부릅니다.

1.2 첫 호출 (curl)

1. 시험 엔드포인트로 WAV 파일을 받습니다.

```
curl -s https://copland.ai/tts/sandbox/v1/audio/speech \
  -H 'Content-Type: application/json' \
  -d '{"input":"안녕하세요. COPLAND TTS 시험 엔드포인트입니다.","voice":"F1"}' \
  -D - -o hello.wav
```

2. 확인 방법: 응답 헤더가 `200 OK` 이고 `hello.wav` 가 재생되면 됩니다.

   예상 출력(헤더 일부):

```
HTTP/1.1 200 OK
Content-Type: audio/wav
x-sample-rate: 32000
x-sentences: 2
x-quota-limit: 100
x-quota-remaining: 96
x-request-id: a6881973eb474ee1
```

3. 자사 서버에서는 주소를 바꾸고 API 키를 붙입니다.

```
export COPLAND_TTS_KEY=ctts_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
curl -s http://<서버>:9300/v1/audio/speech \
  -H "Authorization: Bearer $COPLAND_TTS_KEY" \
  -H 'Content-Type: application/json' \
  -d '{"input":"안녕하세요. 상담원 연결 전에 안내 말씀 드립니다."}' \
  -o hello.wav
```

1.3 OpenAI 호환

`POST /v1/audio/speech` 는 OpenAI의 음성 합성 API와 같은 모양입니다. OpenAI 클라이언트 라이브러리를 쓰고 있다면 **`base_url` 만 바꾸면** 됩니다.

다른 점은 다음과 같습니다.

항목	OpenAI	COPLAND TTS
`model`	`tts-1` 등	`copland-cpu-tts` , `tts-1` , `tts-1-hd` 모두 같은 모델
`voice`	`alloy` 등	`F1` ~ `F5` (여성), `M1` ~ `M5` (남성). OpenAI 화자 이름은 400 `invalid_voice`
`response_format`	mp3·opus·aac·flac·wav·pcm	`wav` (기본), `pcm` 만. 그 밖은 400 `invalid_parameter`
`speed`	0.25~4.0	0.67~1.9
`instructions`	말투 지시	받지만 무시합니다
확장 필드	–	`sample_rate` , `volume` , `normalize` , `stream` , `seed` , `input_type` , `sentence_pause_ms` , `dictionary`
오류 본문	`{"error":{...}}`	같은 형식

- 모르는 필드는 오류 없이 무시합니다.
- OpenAI 클라이언트에서 확장 필드는 `extra_body` 로 넘깁니다.

Python `openai` 패키지(2.x) 예입니다. 이 예는 openai 2.24.0으로 시험 엔드포인트와 설치 서버에서 실제로 돌려 확인했습니다.

```
import os
import openai
from openai import OpenAI

client = OpenAI(base_url="http://<서버>:9300/v1",            # 시험: https://copland.ai/tts/
          sandbox/v1
                api_key=os.environ["COPLAND_TTS_KEY"])        # 시험 엔드포인트는 아무 문자열이나 넣
          습니다

# 스트리밍으로 받아 파일에 쓴다
with client.audio.speech.with_streaming_response.create(
    model="tts-1",                # copland-cpu-tts · tts-1 · tts-1-hd 모두 같은 모델
    voice="F1",                   # F1~F5, M1~M5
    input="안녕하세요. 고객님의 이번 달 청구 금액은 32,500원입니다.",
    response_format="wav",
    speed=1.0,
    extra_body={"sample_rate": 16000},   # OpenAI 에 없는 확장 필드
) as resp:
    print(resp.status_code, resp.headers["x-sample-rate"], resp.headers["x-sentences"],
          resp.headers["x-request-id"])
    resp.stream_to_file("hello.wav")

# 오류는 openai 예외로 온다
try:
    client.audio.speech.create(model="tts-1", voice="alloy", input="안녕")
except openai.BadRequestError as e:
    print(e.status_code, e.body)
```

예상 출력:

```
200 16000 2 fc69c7ee3d904cf2
400 {'message': "voice 는 ['F1', 'F2', 'F3', 'F4', 'F5', 'M1', 'M2', 'M3', 'M4', 'M5'] 중 하나",
'type': 'invalid_request_error', 'code': 'invalid_voice'}
```

- `stream_to_file` 로 저장한 WAV는 헤더의 길이 칸이 `0xFFFFFFFF` 입니다. 이유와 고치는 법은 3.5 스트리밍 WAV에 있습니다.
- `openai` 패키지는 기본으로 429·5xx·연결 실패를 2번까지 다시 보냅니다(`max_retries`). 하루 한도 초과(`daily_quota`)는 다시 보내도 소용이 없습니다. 재시도 방침은 10장을 따르고, 필요하면 `max_retries=0` 으로 끕니다.

1.4 API 한눈에 보기

메서드	경로	권한	하는 일
POST	`/v1/audio/speech`	user	음성 합성 (WAV·PCM 스트리밍)
POST	`/v1/normalize`	user	읽기 미리보기 (사전·정규화 결과)
GET	`/v1/voices`	user	화자 목록
GET	`/v1/models`	user	모델 목록 (OpenAI 형식)
GET	`/v1/dictionaries`	user	사전 목록
GET	`/v1/license`	admin	라이선스와 사용 중인 채널
GET	`/v1/usage`	admin	API 키별 누적 사용량
POST	`/v1/usage/reset`	admin	사용량 초기화
POST	`/v1/dictionary/reload`	admin	사전 다시 읽기
POST	`/v1/keys/reload`	admin	API 키 파일 다시 읽기
GET	`/health`	없음	상태 확인 (부하 분산기용)
GET	`/metrics`	admin	운영 지표 (JSON·Prometheus)
GET	`/openapi.json`	없음	OpenAPI 명세

- **user**는 일반 키와 관리자 키 모두 됩니다. **admin**은 관리자 키만 됩니다.
- 허용 IP 대역(`TTS_ALLOW`)은 모든 경로에 적용됩니다. 권한이 "없음"인 경로도 허용되지 않은 주소에서는 403입니다.
  - 서버 자신(루프백 `127.0.0.1` · `::1`)에서의 접속은 늘 허용합니다.
- 다른 키의 사용량, 장비키·고객명, 운영 지표를 보호하기 위해 `/v1/usage` · `/v1/license` · `/metrics` 는 관리자 키 전용입니다.

2. 인증

2.1 키 보내기

API 키는 두 방법 중 하나로 보냅니다.

```
Authorization: Bearer ctts_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
X-API-Key: ctts_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

- 둘 다 보내면 `Authorization: Bearer` 를 씁니다.
- 키는 `ctts_` 로 시작하는 문자열입니다.
- 키가 없거나 틀리면 401 `invalid_api_key` 입니다.

```
{"error":{"message":"API 키가 없거나 맞지 않
          음","type":"authentication_error","code":"invalid_api_key"}}
```

- 401 응답에는 `WWW-Authenticate` 헤더를 넣지 않습니다. WAS의 전역 인증 처리기가 인증을 다시 시도하지 않게 하려는 것입니다.

2.2 키 받기

- 키는 게이트웨이 서버 운영자가 발급합니다. 웹 서버 개발자는 운영자에게 키를 받습니다.
- 키에는 역할이 있습니다.
  - `user` : 합성, 읽기 미리보기, 화자·모델·사전 목록
  - `admin` : `user` 권한 + 사용량·라이선스·운영 지표 조회, 사용량 초기화, 사전·키 다시 읽기
- 웹 서버에는 user 키를 씁니다. 관리자 키는 운영자와 지표 수집기(Prometheus)에만 둡니다.
- 키마다 이름(예: `web01`)이 붙습니다. 사용량(`/v1/usage`)은 이 이름별로 셉니다. 부서나 서비스마다 키를 따로 받으면 사용량을 나눠 볼 수 있습니다.
- 키에 하루 한도(`daily_per_ip`)가 걸려 있을 수 있습니다. 9.3을 봅니다.
- 키는 웹 서버(백엔드)에만 둡니다. 브라우저·앱에 넣지 않습니다. 브라우저에서 부를 때는 웹 서버가 중계합니다(11.4).

2.3 키가 필요 없는 경로

- `/health` , `/openapi.json` 은 키 없이 부를 수 있습니다.
- 서버를 키 파일 없이 띄우면 인증이 꺼집니다. 이것은 개발용입니다. 납품 설치는 늘 키 파일을 씁니다.

3. 음성 합성 — POST /v1/audio/speech

- 메서드·경로: `POST /v1/audio/speech`
- 권한: user
- 요청 본문: JSON (`Content-Type: application/json`)
- 응답: `audio/wav` 또는 `audio/pcm;rate=<표본률>;channels=1;format=s16le`

3.1 요청 필드

이름	형식	기본	범위·허용값	설명
`input`	문자열	(필수)	1~5000자	읽을 글. SSML이면 태그까지 5000자. 넘으면 413 `input_too_long`
`model`	문자열	`copland-cpu-tts`	`copland-cpu-tts`, `tts-1`, `tts-1-hd`	셋 다 같은 모델입니다. OpenAI 클라이언트의 값을 그대로 받기 위한 별칭입니다
`voice`	문자열	`F1`	`F1` ~ `F5`, `M1` ~ `M5`	화자. 목록은 `GET /v1/voices`
`response_format`	문자열	`wav`	`wav`, `pcm`	`pcm` 은 헤더 없는 16비트 모노
`speed`	실수	`1.0`	0.67~1.9	말 빠르기. 1.0이 보통
`sample_rate`	정수	`32000`	16000, 22050, 24000, 32000, 44100	출력 표본률(Hz). 그 밖은 400 `invalid_sample_rate`
`volume`	실수	`1.0`	0.1~2.0	출력 음량 배율. 1.0이 원음. 넘치는 표본은 잘립니다
`normalize`	참/거짓	`true`	–	`false` 면 사전·숫자 읽기 규칙을 건너뜁니다. SSML 태그는 처리합니다
`stream`	참/거짓	`true`	–	`false` 면 끝까지 만든 뒤 길이가 정확한 WAV를 한 번에 보냅니다
`seed`	정수	없음	0~4294967295	소리를 정하는 시드. 없으면 화자와 문장으로 정한 고정 시드
`input_type`	문자열	`auto`	`auto`, `text`, `ssml`	`auto` 는 `<speak` 또는 `<?xml` 로 시작하면 SSML로 봅니다. `ssml` 은 `<speak>` 가 없으면 감싸 줍니다
`sentence_pause_ms`	정수	없음(서버 기본 300)	0~5000	문장 사이 쉼(ms)
`dictionary`	문자열	없음(기본 사전)	이름 있는 사전 이름	목록은 `GET /v1/dictionaries`. 없는 이름은 400 `invalid_dictionary`

- `speed` 와 SSML `prosody rate` 를 함께 쓰면 두 값을 곱합니다. 곱한 값은 0.5~2.0 안으로 자릅니다.

- `volume` 과 SSML `prosody volume` 도 곱합니다.
- `sentence_pause_ms` 의 서버 기본값은 운영자가 바꿀 수 있습니다(`TTS_SENTENCE_PAUSE_MS`).
- `normalize: false` 는 이미 한글 읽기로 바꿔 보낼 때 씁니다. 이때 숫자·영문은 엔진에 그대로 들어가 잘못 읽힐 수 있습니다.
- `input_type: "text"` 로 보내면 `<speak>` 같은 태그도 글자로 읽습니다.
- `input_type: "ssml"` 로 보내면 `<speak>` 로 시작하지 않아도 `<speak>...</speak>` 로 감싸 SSML로 읽습니다. 예: `안녕하세요.<break time="1s"/>잠시만요.`

3.2 화자와 모델

- 화자는 여성 5명(`F1` ~ `F5`), 남성 5명(`M1` ~ `M5`)입니다.
- 모델은 하나입니다. `copland-cpu-tts` 를 쓰기를 권합니다.

3.3 응답 형식

response_format	Content-Type	본문
`wav`	`audio/wav`	44바이트 WAV 헤더 + PCM
`pcm`	`audio/pcm;rate=32000;channels=1;format=s16le` (`rate` 는 요청 표본률)	PCM만

PCM 형식은 다음과 같습니다.

- 부호 있는 16비트 정수, 리틀 엔디언(`s16le`)
- 모노(1채널)
- 표본률 = 요청 `sample_rate` (응답 헤더 `X-Sample-Rate` 에도 있습니다)
- 1초 = `sample_rate` × `2` 바이트. 32kHz면 64,000바이트입니다.
- 0.3.1까지는 `Content-Type` 이 `audio/L16;rate=...;channels=1` 이었습니다. 이 값으로 형식을 판별하던 코드는 `audio/pcm` 도 받게 고칩니다.

PCM 파일을 WAV로 바꾸는 명령입니다.

```
ffmpeg -f s16le -ar 32000 -ac 1 -i a.pcm a.wav
sox -t raw -r 32000 -e signed -b 16 -c 1 a.pcm a.wav
```

3.4 응답 헤더

헤더	언제	값
`X-Sample-Rate`	늘	출력 표본률. 예: `32000`
`X-Sentences`	늘	합성 단위(문장) 수. 예: `2`
`X-Request-Id`	늘 (모든 API, 오류 포함)	요청 번호. 요청에 `X-Request-Id` 를 보내면 그 값을 그대로 돌려줍니다. 없으면 16자리 16진수를 만듭니다
`X-License-Warning`	라이선스 유예 기간에만	예: `expired 2027-12-31; grace until +14d`
`X-Quota-Limit`	하루 한도가 걸린 키	하루 한도. 예: `100`
`X-Quota-Remaining`	하루 한도가 걸린 키	오늘 남은 횟수. 예: `96`
`Transfer-Encoding: chunked`	`stream: true`	길이를 모르고 보냄
`Content-Length`	`stream: false`	전체 길이

- 문제를 문의할 때는 `X-Request-Id` 를 알려 주세요. 게이트웨이 로그에서 그 요청을 찾습니다.
- 자사 요청 번호를 `X-Request-Id` 로 보내면 웹 서버 로그와 게이트웨이 로그를 같은 번호로 맞춰 볼 수 있습니다.

```
curl -s -o /dev/null -D - http://<서버>:9300/v1/voices -H "Authorization: Bearer
          $COPLAND_TTS_KEY" -H "X-Request-Id: web01-20261012-0001" | grep -i x-request
```

예상 출력:

```
x-request-id: web01-20261012-0001
```

- `X-License-Warning` 이 보이면 운영자에게 알립니다. 유예 기간(기본 14일)이 지나면 합성이 403 `license_expired` 로 거부됩니다.

3.5 스트리밍 WAV

`stream: true` (기본)이면 첫 문장이 합성되자마자 보내기 시작합니다. 이때 전체 길이를 아직 모릅니다. 그래서 WAV 헤더의 두 크기 칸에 최댓값 `0xFFFFFFFF` 를 넣습니다. OpenAI도 같은 방식입니다.

스트리밍 WAV 의 앞 44바이트(헤더)와 소리

오프셋	내용
0	"RIFF"
4	RIFF 크기 = **0xFFFFFFFF** — 스트리밍이라 아직 모름
8	"WAVEfmt "
16~35	형식: PCM, 모노, 표본률, 바이트율, 블록 2, 16비트
36	"data"
40	data 크기 = **0xFFFFFFFF** — 스트리밍이라 아직 모름
44~	**소리 (PCM)** 16비트 모노, 리틀 엔디언 — 첫 문장이 합성되자마자 이어서 옵니다

스트리밍 WAV 헤더 구조

실제로 받은 헤더(32kHz)입니다.

```
00000000: 5249 4646 ffff ffff 5741 5645 666d 7420  RIFF....WAVEfmt
00000010: 1000 0000 0100 0100 007d 0000 00fa 0000  .........}......
00000020: 0200 1000 6461 7461 ffff ffff 0000 0000  ....data........
```

- 브라우저 `<audio>`, ffmpeg, sox, 대부분의 재생기는 이 파일을 끝까지 바르게 재생합니다.
- 길이를 헤더에서 읽는 프로그램은 틀린 값을 냅니다. 예를 들어 Python `wave` 모듈은 6.4초 파일을 134,217초로 읽습니다.
- **파일로 저장해 둘 거라면** 다음 중 하나를 씁니다.

1. 처음부터 `stream: false` 로 받습니다. 헤더 길이가 정확합니다.
2. `response_format: "pcm"` 으로 받아 직접 WAV 헤더를 붙입니다(11.2 예제는 Python `wave` 모듈로 씁니다).
3. 받은 뒤 헤더를 고칩니다. 게이트웨이 WAV 헤더는 늘 44바이트이므로 두 칸만 바꾸면 됩니다.

```
import struct

def fix_wav_header(path):
    """스트리밍으로 받은 WAV(크기 칸 0xFFFFFFFF)의 RIFF·data 크기를 실제 값으로 고친다."""
    with open(path, "r+b") as f:
        f.seek(0, 2)
        data = f.tell() - 44                # 게이트웨이 WAV 헤더는 늘 44바이트
        f.seek(4);  f.write(struct.pack("<I", 36 + data))
        f.seek(40); f.write(struct.pack("<I", data))
```

명령줄 도구로도 고칠 수 있습니다.

```
ffmpeg -i stream.wav -c copy fixed.wav
sox --ignore-length stream.wav fixed.wav
```

확인 방법: `ffprobe -v error -show_entries format=duration -of csv=p=0 fixed.wav` 가 실제 길이(예: `6.402250`)를 보입니다. Python `wave.open("fixed.wav").getnframes()` 도 바른 값(예: 16kHz 6.4초 → `102436`)을 냅니다.

3.6 한 번에 받기 (stream: false)

- `stream: false` 면 모든 문장을 만든 뒤 보냅니다. `Content-Length` 가 붙고 WAV 헤더 길이가 정확합니다.
- 파일 저장, 메일 첨부, 길이가 필요한 처리에 씁니다.
- 대신 첫 소리가 늦습니다. 전체를 다 만들 때까지 아무것도 오지 않습니다. 바로 재생하려면 `stream: true` 를 씁니다.
- PCM 내용은 `stream: true` 와 바이트까지 같습니다. 헤더의 크기 칸만 다릅니다.

```
curl -s http://<서버>:9300/v1/audio/speech -H "Authorization: Bearer $COPLAND_TTS_KEY" -H
          'Content-Type: application/json' \
  -d '{"input":"안녕하세요. 상담원 연결 전에 안내 말씀 드립니다.","stream":false}' -D - -o b.wav
```

예상 출력(헤더):

```
HTTP/1.1 200 OK
x-sample-rate: 32000
x-sentences: 2
content-length: 336164
content-type: audio/wav
x-request-id: 0069690f6d6a48b9
```

3.7 시드와 결과 고정

- 같은 요청은 늘 같은 소리입니다. 화자·문장·빠르기·표본률·시드가 같으면 같은 PCM이 나옵니다.
- `seed` 를 주지 않으면 화자와 문장으로 시드를 정합니다. 그래서 시드를 주지 않아도 결과가 고정됩니다.

- `seed` 를 주면 같은 문장도 다른 억양으로 나옵니다. 마음에 드는 억양의 시드를 찾아 고정해 쓸 수 있습니다.
- 서버는 문장 단위로 결과를 **캐시**합니다. 같은 문장을 다시 요청하면 합성하지 않고 바로 보냅니다.
  - 같은 문장이 동시에 여러 번 들어와도 한 번만 합성해 나눠 줍니다.
  - 자주 쓰는 고지문은 운영자가 예열 목록에 넣어 둘 수 있습니다. 서버를 다시 띄워도 첫 요청부터 바로 나갑니다.
  - 사전을 다시 읽으면(`/v1/dictionary/reload`) 캐시를 비웁니다.

실제 확인(같은 요청 두 번, 본문의 MD5 앞 8자리):

```
seed 없음   3f18b953 3f18b953
seed 7      ace5f976 ace5f976
voice F2    7370b59f
```

3.8 오류가 나는 때

- 요청 값이 틀리면 소리를 보내기 전에 HTTP 오류로 답합니다. 형식은 8장에 있습니다.
- **응답 상태(200)는 첫 문장을 합성한 뒤에 정해집니다.** 첫 문장 합성에 실패하면 HTTP 오류(예: 504)로 받습니다.
- 소리를 보내기 시작한 뒤(두 번째 문장부터) 실패하면 HTTP 상태를 바꿀 수 없습니다. 이때 서버는 연결을 끊습니다.
  - 클라이언트에는 청크 전송이 덜 끝난 오류로 보입니다(예: curl 18번 오류, httpx `RemoteProtocolError`).
  - 이 경우 처리는 10장을 봅니다.
- 클라이언트가 연결을 끊으면 서버는 남은 문장의 합성을 멈추고 채널을 바로 돌려줍니다. 재생을 멈출 때는 연결을 끊으면 됩니다.

4. 긴 글과 재생

4.1 문장 분할

게이트웨이는 입력을 읽기 규칙으로 바꾼 뒤 문장으로 나눕니다. 문장 하나가 합성 1번이고 캐시 1칸입니다.

1. `.` `?` `!` `。` 뒤에 공백이 오면 끊습니다. 줄바꿈에서도 끊습니다.
2. 한 문장이 **120자**를 넘으면 120자 안의 마지막 `,` 또는 공백에서 끊습니다.
3. 끊을 곳이 앞쪽 40자 안에만 있으면 120자에서 그냥 자릅니다.
4. SSML의 `<p>`, `<s>`, `<break>`, `<prosody>` 경계에서도 끊습니다.

나뉜 결과는 `POST /v1/normalize`의 `sentences`로 미리 볼 수 있습니다. 147자짜리 문장 하나를 넣은 예입니다.

```
117 고객님께서 가입하신 이 상품은 예금자보호법에 따라 … 상품 설명서와 약관을 반드시 읽어
29 보시고 궁금한 점은 상담원에게 문의해 주시기 바랍니다
```

- 긴 문장은 어색한 곳에서 끊길 수 있습니다. 쉼표나 마침표로 직접 나눠 주면 더 자연스럽습니다.

4.2 첫 소리 시간

- 첫 소리까지 걸리는 시간은 **첫 문장 하나**를 합성하는 시간입니다. 글 전체 길이와 상관없습니다.
- 서버는 첫 문장을 보내는 동안 다음 문장을 미리 합성합니다. 재생보다 늘 한 문장 앞섭니다.
- 시험 서버(Intel Xeon w5-2445, 물리 10코어, AMX)에서 잰 값입니다.
  - 물리 4코어, 150채널: 끊김 0, 첫 소리 p95 368ms
  - 물리 6코어, 150채널: 끊김 0, 첫 소리 p95 247ms
  - 권장 사양은 AMX 물리 6코어 + 1코어입니다. AMX가 없는 서버는 약 3배의 코어가 필요합니다.
- 캐시에 있는 문장은 합성하지 않으므로 첫 소리가 수 ms입니다.
- 첫 소리를 빠르게 하려면 다음과 같이 합니다.
  - 첫 문장을 짧게 씁니다(예: "안녕하세요.").
  - 자주 쓰는 고지문은 운영자에게 예열 목록에 넣어 달라고 합니다.
  - `stream: true`로 받아 받는 대로 재생합니다.

4.3 문장 사이 쉼

- 문장 사이에는 무음을 넣습니다. 기본 300ms입니다.
- 요청마다 `sentence_pause_ms` (0~5000)로 바꿀 수 있습니다.
- 한 곳만 쉼을 바꾸려면 SSML `<break>`를 씁니다. `<break>`가 연달아 오면 쉼을 더합니다(최대 5초).
- 마지막 문장 뒤에는 무음을 붙이지 않습니다.

5. 읽기 규칙 (정규화)

COPLAND TTS는 숫자·기호·영문 약어를 한글 읽기로 바꾼 뒤 합성합니다. 이 바꾸기를 정규화라고 합니다. 결과는 `POST /v1/normalize` 로 미리 볼 수 있습니다.

5.1 적용 순서

앞 단계가 뒤 단계를 이깁니다.

1. 사용자 사전 (이름 있는 사전 → 고객사 사전 → 기본 사전, 긴 표기부터)
2. 날짜, 시각, 전화번호, 계좌·카드번호
3. 범위(`~` , 뒤에 단위가 붙으면 앞뒤 수를 모두 그 단위로 읽음), 퍼센트(`%`)
4. 숫자 + 단위
5. 남은 숫자 → 한자어 수 (숫자에 붙은 쉼표는 천 단위일 때만 숫자로 봄)
6. 영문 대문자 약어 → 알파벳 이름
7. `&` → "앤", 남은 `~` → "에서"

5.2 규칙표

아래 예시는 모두 `/v1/normalize` 로 실제로 돌린 결과입니다. 사전에 `KB → 케이비` 같은 항목이 있는 상태입니다.

종류	규칙	입력	읽기
대표번호	15xx~19xx-xxxx, 한 자리씩, 0 = 공	`문의는 1588-1234로 주세요.`	문의는 일오팔팔 일이삼사로 주세요.
전화번호	0으로 시작, `-` `.` 공백으로 3묶음	`02-123-4567`	공이 일이삼 사오육칠
전화번호	괄호 지역번호	`(02) 123-4567`	공이 일이삼 사오육칠
휴대전화	하이픈 있음	`010-1234-5678`	공일공 일이삼사 오육칠팔
휴대전화	붙여 쓴 11자리 (010·011·016~019)	`01012345678`	공일공 일이삼사 오육칠팔
전화번호	점으로 구분	`031.123.4567`	공삼일 일이삼 사오육칠
계좌번호	하이픈 3묶음 이상, 한 자리씩	`계좌 110-123-456789`	계좌 일일공 일이삼 사오육칠팔구
카드번호	하이픈 4묶음	`카드 1234-5678-9012-3456`	카드 일이삼사 오육칠팔 구공일이 삼사오육
날짜	`YYYY-MM-DD`	`2026-10-12`	이천이십육 년 시월 십이 일
날짜	`YYYY.M.D.`	`2026.6.5.`	이천이십육 년 유월 오 일
날짜	`YYYY/MM/DD`	`2026/10/25`	이천이십육 년 시월 이십오 일
시각	1~12시는 고유어	`9:05`	아홉 시 오 분
시각	0분은 생략	`오후 3:00`	오후 세 시
시각	13~24시는 한자어	`14:30`	십사 시 삼십 분
금액	쉼표 무시, 만 단위로 띄움	`32,500원`	삼만 이천오백 원
금액	1만은 "만"	`10000원`	만 원
금액	큰 수	`1,000,000원`	백만 원
고유어 단위	1~99 + 개·명·살·시간 등	`3개`	세 개
고유어 단위	20 = 스무	`20살`	스무 살
고유어 단위	21 = 스물한	`21명`	스물한 명
고유어 단위	100 이상은 한자어	`100명`	백 명
고유어 단위	시간	`3시간`	세 시간
한자어 단위	개월·원·년·월·일·세 등	`5개월`	오 개월
한자어 단위	나이(세)	`만 65세`	만 육십오 세
월	6월 = 유월, 10월 = 시월	`6월 10월`	유월 시월
외화	달러·엔·유로	`1,000달러`	천 달러

종류	규칙	입력	읽기
소수	소수점 아래는 한 자리씩	`3.14`	삼 점 일 사
소수 + 단위		`1,234.5원`	천이백삼십사 점 오 원
퍼센트	`%` → 퍼센트	`1.5%`	일 점 오 퍼센트
퍼센트		`10%`	십 퍼센트
범위 + 한자어 단위	앞뒤 모두 단위를 붙여 읽음	`3~5일`	삼 일에서 오 일
범위 + 고유어 단위		`3~5개`	세 개에서 다섯 개
시각 범위		`오전 9시~10시`	오전 아홉 시에서 열 시
범위 (단위 없음)	`~` → 에서	`3~5`	삼에서 오
쉼표	천 단위가 아닌 쉼표는 문장 쉼표로 남김	`잔액은 1000, 이자는 20입니다.`	잔액은 천, 이자는 이십입니다.
기타 숫자	한자어	`제12조`	제십이조
영문 약어	대문자 묶음 → 알파벳 이름	`ATM과 ARS`	에이티엠과 에이알에스
영문 약어	사전 우선	`KB국민카드`	케이비국민카드
영문 약어	`&` → 앤	`AT&T`	에이티 앤 티
기본 사전	통신·단위	`Wi-Fi 연결`	와이파이 연결
기본 사전	단위 기호	`2.5km`	이 점 오 킬로미터
대소문자 섞인 영문	바꾸지 않음 → 사전 등록 후보	`iPhone 15`	iPhone 십오 (`unknown_latin: ["iPhone"]`)
사용자 사전	이름 있는 사전 `card`	`리볼빙 신청` (`dictionary: "card"`)	일부결제금액이월약정 신청

단위 목록은 다음과 같습니다.

- **고유어로 읽는 단위**(1~99만): 시간, 가지, 마리, 번째, 개, 명, 살, 장, 권, 잔, 곳, 벌, 대, 통, 건, 시, 배, 달
- **한자어로 읽는 단위**: 개월, 주년, 퍼센트, 원, 년, 월, 일, 분, 초, 층, 호, 회, 차, 등, 위, 번, 주, 세, 점, 킬로, 미터, 그램, 리터, 달러, 엔, 유로, 포인트, 인분

5.3 사용자 사전

- 사전은 서버에 있는 `표기<TAB>읽기` 줄 목록입니다. 운영자가 관리합니다.
- 우선순위: 이름 있는 사전(요청 `dictionary`) → 고객사 사전 → 기본 사전입니다.
- 긴 표기부터 바꿉니다. `KB국민` 이 `KB` 보다 먼저 맞습니다.
- 영문 표기는 단어 경계에서만 맞춥니다. `app` 이 `apple` 을 바꾸지 않습니다.
- 사전은 다른 모든 규칙보다 먼저 적용됩니다. 숫자 규칙이 틀리는 표기도 사전에 넣으면 고칠 수 있습니다.
- `/v1/normalize` 응답의 `unknown_latin` 은 정규화 뒤에도 남은 영문 단어입니다. 사전에 넣을 후보입니다.

5.4 규칙이 맞지 않을 때

규칙이 원하는 대로 읽지 않으면 다음 순서로 대처합니다.

1. **사전에 등록합니다.** 자주 나오는 표기(상품명, 약어, 고유명사)는 운영자에게 사전 등록을 요청합니다. 모든 요청에 적용됩니다.
2. **SSML로 그 자리만 바꿉니다.**
   - 한 자리씩 읽기: `<say-as interpret-as="digits">502013</say-as>` → 오공이공일삼
   - 읽기 직접 지정: `<sub alias="일부결제금액이월약정">리볼빙</sub>`
3. **직접 한글로 바꿔 보냅니다.** 웹 서버가 이미 한글 읽기로 바꿨다면 `normalize: false` 로 보냅니다.

자주 생기는 예입니다.

입력	결과	원하는 읽기	대처
`1588 1234` (공백)	천오백팔십팔 천이백삼십사	일오팔팔 일이삼사	`1588-1234` 로 쓰거나 `say-as telephone`
`502013` (인증번호)	오십만 이천십삼	오공이공일삼	`say-as digits`
`iPhone`	그대로 남음	아이폰	사전 등록 또는 `sub`
`3번째`	세 번째	(맞음)	`say-as` 로 감싸지 않습니다(아래 SSML 참고)

6. SSML

`input` 이 `<speak` (또는 `<?xml`)로 시작하면 SSML로 봅니다. `input_type: "ssml"` 로 명시할 수도 있습니다. 이때 `<speak>` 가 없으면 게이트웨이가 `<speak>...</speak>` 로 감쌉니다.

6.1 지원 태그

태그	속성	값	동작
`speak`	`xmlns` · `version` · `xml:lang` 등은 무시	–	최상위 태그. 없으면 오류
`break`	`time`	`500ms` , `1s` , `1.5s`	쉼. 최대 5초. 여기서 문장을 끊습니다
`break`	`strength`	`none` 0, `x-weak` 100, `weak` 200, `medium` 300, `strong` 600, `x-strong` 1000 (ms)	`time` 이 없을 때. 둘 다 없으면 `medium`
`say-as`	`interpret-as`	`digits` , `telephone`	숫자를 한 자리씩, 0 = 공. 숫자가 아닌 글자에서 묶음을 띄웁니다
`say-as`	`interpret-as`	`characters` , `spell-out`	한 글자씩: 숫자 → 일·이…, 영문 → 에이·비…, `-` `_` `.` 공백은 뺍니다
`say-as`	`interpret-as`	`cardinal` , `number`	한자어 수. 숫자 형식이 아니면 일반 규칙으로 읽습니다
`say-as`	`interpret-as`	`date`	날짜 규칙. 형식이 맞지 않으면 일반 규칙으로 읽습니다
`say-as`	`interpret-as`	그 밖(`ordinal` 등)	지원하지 않습니다. 내용을 일반 규칙으로 읽습니다
`sub`	`alias` (필수)	읽을 말	내용 대신 `alias` 를 읽습니다. 없으면 오류
`prosody`	`rate`	`x-slow` 0.7, `slow` 0.85, `medium` · `default` 1.0, `fast` 1.2, `x-fast` 1.4, `80%` , `+10%` , `-10%` , `1.2`	빠르기 배율. 0.5~2.0으로 자릅니다
`prosody`	`volume`	`silent` 0.1, `x-soft` 0.5, `soft` 0.7, `medium` · `default` 1.0, `loud` 1.3, `x-loud` 1.6, `+6dB` , `-3dB` , `50%` , `+20%`	음량 배율. 0.1~2.0으로 자릅니다
`p` , `s`	–	–	문단·문장 경계
`audio`	–	–	내용째 뺍니다(소리 파일을 재생하지 않습니다)
`voice` , `lang` , `mark` , `emphasis` , `token` 등	–	–	태그만 무시하고 내용은 읽습니다

- `say-as` 와 `sub` 의 결과는 이미 한글 읽기이므로 사전·숫자 규칙을 다시 타지 않습니다.

- 태그 앞뒤의 띄어쓰기는 원문대로 지킵니다. 태그 바로 뒤에 붙여 쓴 조사는 붙여 읽습니다. 예: `<say-as interpret-as="digits">1234</say-as>입니다` → 일이삼사입니다
- `prosody` 의 `rate` · `volume` 은 요청의 `speed` · `volume` 과 곱합니다. `prosody` 안에 `prosody` 를 넣으면 배율이 곱해집니다.
- `prosody` 앞뒤에서 문장을 끊고, 그 경계에는 쉼을 넣지 않습니다(0ms). 말투가 바뀌는 곳을 끊김 없이 이어 붙이려는 의도한 동작입니다. 쉼이 필요하면 `<break>` 를 넣습니다.
- `%` · `+` · `-` 값: `80%` 는 0.8배, `+10%` 는 1.1배, `-10%` 는 0.9배입니다. `dB` 는 `volume` 에만 씁니다(`+6dB` ≈ 2배).
- SSML은 XML입니다. 본문의 `<` 는 `<` , `&` 는 `&` 로 씁니다.
- `<!DOCTYPE>` · `<!ENTITY>` 는 받지 않습니다.

6.2 예시

모두 `/v1/normalize` 로 확인한 결과입니다.

```
{"input":"<speak>문의는 <say-as interpret-as=\"telephone\">1588-1234</say-as>입니다.</speak>"}
```

```
문의는 일오팔팔 일이삼사입니다.
```

```
{"input":"<speak>코드 <say-as interpret-as=\"characters\">AB-12</say-as></speak>"}
```

```
코드 에이 비 일 이
```

```
{"input":"<speak><sub alias=\"일부결제금액이월약정\">리볼빙</sub> 안내입니다.</speak>"}
```

```
일부결제금액이월약정 안내입니다.
```

쉼 예시입니다. `units` 의 `pause_ms` 가 각 문장 뒤의 쉼입니다. 연달아 온 `strong` (600)과 `500ms` 는 더해 1100ms가 됩니다.

```
{"input":"<speak>안녕하세요.<break time=\"1s\"/>잠시만요.<break strength=\"strong\"/><break
          time=\"500ms\"/>다음입니다.</speak>"}
```

```
"units":[{"text":"안녕하세요.","rate":1.0,"volume":1.0,"pause_ms":1000},
         {"text":"잠시만요.","rate":1.0,"volume":1.0,"pause_ms":1100},
         {"text":"다음입니다.","rate":1.0,"volume":1.0,"pause_ms":300}]
```

말투 예시입니다.

```
{"input":"<speak><prosody rate=\"slow\"
          volume=\"loud\">중요한 안내입니다.</prosody> 이어서 <prosody rate=\"120%\"
          volume=\"+6dB\">빠르게</prosody></speak>"}
```

```
"units":[{"text":"중요한 안내입니다.","rate":0.85,"volume":1.3,"pause_ms":0},
         {"text":"이어서","rate":1.0,"volume":1.0,"pause_ms":0},
         {"text":"빠르게","rate":1.2,"volume":1.995,"pause_ms":0}]
```

6.3 SSML 오류

SSML이 잘못되면 400 `invalid_ssml` 입니다. `message` 에 원인이 있습니다.

입력	message
`<speak>안녕`	`SSML 구문 오류: no element found: line 1, column 9`
`<speak>a < b</speak>`	`SSML 구문 오류: not well-formed (invalid token): line 1, column 10`
`<?xml version="1.0"?><foo>안녕</foo>`	`최상위는 <speak>`
`<speak><sub>KB</sub></speak>`	`sub 에 alias 가 없음`
`<break time="1분"/>`	`break time 형식: 1분 (예: 500ms, 1s)`
`<prosody rate="빠르게">`	`알 수 없는 값: 빠르게`
`<!DOCTYPE ...>` 포함	`DOCTYPE·ENTITY 는 받지 않음`

응답 예:

```
{"error":{"message":"break time 형식: 1분 (예: 500ms,
          1s)","type":"invalid_request_error","code":"invalid_ssml"}}
```

- SSML 안에 읽을 내용이 없으면(예: `<audio>` 만 있음) 400 `empty_input` 입니다.

7. 그 밖의 엔드포인트

7.1 POST /v1/normalize — 읽기 미리보기

- 권한: user
- 하는 일: 엔진에 들어갈 읽기를 미리 봅니다. 합성하지 않습니다. 하루 한도(`daily_per_ip`)를 세지 않습니다.
- 용도: 사전 작업, 화면에 "실제로 읽는 문장" 표시, 정규화 결과 확인

이름	형식	기본	범위·허용값	설명
`input`	문자열	(필수)	1~5000자	읽을 글
`input_type`	문자열	`auto`	`auto` , `text` , `ssml`	합성과 같음
`dictionary`	문자열	없음	이름 있는 사전	합성과 같음

응답 필드:

필드	뜻
`normalized`	읽기 전체 (문장을 공백으로 이음)
`sentences`	합성 단위별 문장
`unknown_latin`	남은 영문 단어 (사전 등록 후보)
`units`	합성 단위별 `text` · `rate` · `volume` · `pause_ms`

```
curl -s http://<서버>:9300/v1/normalize -H "Authorization: Bearer $COPLAND_TTS_KEY" -H 'Content-
        Type: application/json' \
  -d '{"input":"KB국민카드 32,500원. 문의는 1588-1234로 주세요. Wi-Fi 확인"}'
```

예상 출력:

```
{"normalized":"케이비국민카드 삼만 이천오백 원. 문의는 일오팔팔 일이삼사로 주세요. 와이파이 확
        인","sentences":["케이비국민카드 삼만 이천오백 원.","문의는 일오팔팔 일이삼사로 주세요.","와
        이파이 확인"],"unknown_latin":[],"units":[{"text":"케이비국민카드 삼만 이천오백
        원.","rate":1.0,"volume":1.0,"pause_ms":300},{"text":"문의는 일오팔팔 일이삼사로 주세
        요.","rate":1.0,"volume":1.0,"pause_ms":300},{"text":"와이파이 확인","rate":1.0,"volume":
        1.0,"pause_ms":300}]}
```

7.2 GET /v1/voices — 화자 목록

- 권한: user
- 요청 필드: 없음

```
curl -s http://<서버>:9300/v1/voices -H "Authorization: Bearer $COPLAND_TTS_KEY"
```

```
{"voices":["F1","F2","F3","F4","F5","M1","M2","M3","M4","M5"]}
```

7.3 GET /v1/models — 모델 목록

- 권한: user
- 요청 필드: 없음
- OpenAI `GET /v1/models` 와 같은 형식입니다.

```
{"object":"list","data":[{"id":"copland-cpu-tts","object":"model","owned_by":"copland"},
          {"id":"tts-1","object":"model","owned_by":"copland"},{"id":"tts-1-
          hd","object":"model","owned_by":"copland"}]}
```

7.4 GET /v1/dictionaries — 사전 목록

- 권한: user
- 요청 필드: 없음
- `name` 이 `default` 인 항목이 기본 사전(고객사 사전 + 기본 사전)입니다. 나머지가 요청 `dictionary` 에 쓸 수 있는 이름입니다.
- `entries` 는 표기 개수, `files` 는 읽은 파일(앞이 우선)입니다.

```
{"dictionaries":[{"name":"default","entries":33,"files":["/opt/copland-tts/conf/customer.tsv","/
          app/gateway/dicts/base.tsv"]},{"name":"card","entries":34,"files":["/opt/copland-tts/
          conf/dicts/card.tsv","/opt/copland-tts/conf/customer.tsv","/app/gateway/dicts/
          base.tsv"]}]}
```

(파일 경로는 설치 기본값으로 바꿔 적었습니다.)

7.5 GET /v1/license — 라이선스와 채널

- 권한: admin (user 키는 403 `admin_required`)
- 요청 필드: 없음

필드	뜻
`license_id`	라이선스 번호
`entitlements.channels`	허가 동시 채널 수
`expires`	만료일
`grace_days`	만료 뒤 유예 일수
`state`	`valid` (유효), `grace` (유예 중), `expired` (만료, 합성 거부)
`days_left`	만료까지 남은 날
`machine_key`	이 장비의 장비키
`customer`	고객사 이름
`active_channels`	지금 쓰는 채널 수 (재생이 끝나지 않은 요청)
`inflight`	지금 보내는 중인 요청 수


```
{"license_id":"L-9000001","product":"tts","product_name":"COPLAND TTS","entitlements":{"channels":
          20},"expires":"2027-12-31","grace_days":14,"state":"valid","days_left":445,"machine_key":"ABCD-
          EFGH","customer":"예시은행","active_channels":0,"inflight":0}
```


(`machine_key` · `customer` 는 예시 값으로 바꿨습니다.)

user 키로 부르면 403입니다. `/v1/usage` · `/metrics` 도 같습니다.

```
HTTP/1.1 403 Forbidden
{"error":{"message":"관리자 키가 필요함","type":"permission_error","code":"admin_required"}}
```

7.6 GET /v1/usage — 키별 사용량

- 권한: admin (user 키는 403 `admin_required`)
- 요청 필드: 없음
- 서버 기동 이후, 또는 마지막 초기화 이후의 누적입니다. 서버를 다시 띄우면 0이 됩니다.
- `keys` 의 키는 API 키 이름입니다. `requests` 는 합성 요청 수, `chars` 는 입력 글자 수, `audio_sec` 는 보낸 음성 길이(초)입니다.
- 거절된 요청(400·429 등)은 세지 않습니다.
- `/metrics` 의 `audio_sec` 와 같은 기준(문장 사이 쉼 포함)입니다.


```
{"since":"2026-10-12T23:40:21","keys":{"web01":{"requests":19,"chars":632,"audio_sec":
          106.6},"trial":{"requests":6,"chars":42,"audio_sec":9.7}}}
```

7.7 POST /v1/usage/reset — 사용량 초기화

- 권한: admin
- 요청 본문: 없음
- 초기화 **직전**의 사용량을 돌려주고 0으로 비웁니다. 월말 정산처럼 "읽고 비우기"에 씁니다.

```
curl -s -X POST http://<서버>:9300/v1/usage/reset -H "Authorization: Bearer
          $COPLAND_TTS_ADMIN_KEY"
```

```
{"since":"2026-10-12T23:40:21","keys":{"web01":{"requests":19,"chars":632,"audio_sec":
          106.6},"trial":{"requests":6,"chars":42,"audio_sec":9.7}}}
```

이어서 `GET /v1/usage` 를 부르면 비어 있습니다.

```
{"since":"2026-10-12T23:41:46","keys":{}}
```

user 키로 부르면 403입니다.

```
{"error":{"message":"관리자 키가 필요함","type":"permission_error","code":"admin_required"}}
```

7.8 POST /v1/dictionary/reload — 사전 다시 읽기

- 권한: admin
- 요청 본문: 없음
- 사전 파일을 모두 다시 읽습니다. 서버를 다시 띄우지 않아도 됩니다.
- 모두 읽혀야 바꿉니다. 하나라도 못 읽으면 400 `dictionary_error` 이고 이전 사전을 그대로 씁니다.
- 문장 캐시를 비우고, 예열 목록을 다시 합성합니다. 잠시 첫 소리가 느려질 수 있습니다.
- 응답은 `GET /v1/dictionaries` 와 같습니다.

```
curl -s -X POST http://<서버>:9300/v1/dictionary/reload -H "Authorization: Bearer
          $COPLAND_TTS_ADMIN_KEY"
```

7.9 POST /v1/keys/reload — API 키 다시 읽기

- 권한: admin
- 요청 본문: 없음
- API 키 파일을 다시 읽습니다. 키 추가·삭제를 서버 재기동 없이 반영합니다.
- 파일에 틀린 줄이 있으면 400 `keys_error` 이고 이전 키를 그대로 씁니다.
- 응답 `keys` 는 읽은 키 개수입니다.

```
{"keys":3}
```

7.10 GET /health — 상태 확인

- 권한: 없음 (허용 IP 대역은 적용)
- 요청 필드: 없음
- 부하 분산기의 상태 검사에 씁니다. 준비되면 200입니다. 합성 워커 재기동 중이거나 라이선스가 만료(유예 기간도 지남)되면 503입니다. 503이면 부하 분산기와 Java SDK가 다른 서버로 넘깁니다.

```
curl -s http://<서버>:9300/health
```

```
{"status":"ok","version":"0.3.4","commit":"dev","workers":2,"license":
          {"license_id":"L-9000001","state":"valid","expires":"2027-12-31","channels":20}}
```

- `status` : `ok` (200), `restarting` (503), `license_expired` (503)
- `commit` : 설치 패키지의 빌드 커밋. 문의할 때 함께 알려 주세요.

7.11 GET /metrics — 운영 지표

- 권한: admin (user 키는 403 `admin_required`)
- 쿼리: `format=json` (기본) 또는 `format=prometheus` . 그 밖은 400 `invalid_parameter`
- 주요 필드:

필드	뜻
`requests` , `errors` , `rejected` , `timeouts` , `aborted` , `auth_failed`	기동 이후 누적: 합성 요청, 오류, 429 거절, 합성 시간 초과, 클라이언트가 끊음, 인증 실패
`sentences` , `audio_sec`	기동 이후 누적: 보낸 문장 수, 보낸 음성 길이(초, 문장 사이 쉼 포함 — `/v1/usage` 와 같은 기준)
`inflight` , `pool_queued` , `est_wait_ms`	지금 보내는 요청 수, 합성 대기·실행 문장 수, 새 요청의 예상 대기 (ms)
`license_channels` , `license_active_channels`	허가 채널, 사용 중 채널
`cache`	`hits` · `joined` · `misses` · `entries` · `pinned` (예열)· `mb`
`ttfb_ms`	첫 소리 시간 p50·p95·p99 (최근 2000건)
`sentence_ms`	문장 합성 시간 p50·p95
`prewarm`	예열 진행 (`total` · `done` · `failed` · `running`)

JSON 예(일부):

{"version":"0.3.4","uptime_sec":61.9,"workers":2,"worker_restarts":0,"requests":1,"inflight":0,"errors":0,"rej
{"hits":0,"joined":0,"misses":3,"entries":2,"pinned":1,"mb":0.7},"prewarm":{"total":1,"done":1,"fail
{"p50":406.9,"p95":406.9}, …}

Prometheus 예(일부):

```
# HELP copland_tts_requests_total 요청 수
# TYPE copland_tts_requests_total counter
copland_tts_requests_total 0
# HELP copland_tts_rejected_total 과부하 거절(429)
# TYPE copland_tts_rejected_total counter
copland_tts_rejected_total 0
```

- 관리자 키 전용입니다. user 키로 부르면 403 `admin_required` 입니다.
- Prometheus 수집기에는 관리자 키를 헤더로 붙입니다(`authorization` 설정).

7.12 GET /openapi.json — OpenAPI 명세

- 권한: 없음
- 모든 경로와 요청 필드의 OpenAPI 3 명세입니다. 코드 생성기나 Postman에 불러 쓸 수 있습니다.
- 고객망에서 외부 CDN을 쓸 수 없는 경우가 많아 Swagger UI 화면은 넣지 않았습니다.

```
curl -s http://<서버>:9300/openapi.json -o copland-tts-openapi.json
```

8. 오류

8.1 형식

모든 오류는 OpenAI와 같은 JSON입니다. 응답에 `X-Request-Id` 가 붙습니다.

```
{"error":{"message":"사람이 읽는 설명","type":"invalid_request_error","code":"invalid_voice"}}
```

- `code` 로 분기합니다. `message` 는 바뀔 수 있으므로 화면 표시와 로그에만 씁니다.
- `type` 은 HTTP 상태로 정해집니다.

HTTP	type
400, 405, 413	`invalid_request_error`
401	`authentication_error`
403	`permission_error`
404	`not_found_error`
429	`rate_limit_error`
503, 504	`server_error`

- 예상하지 못한 내부 오류는 500으로 올 수 있습니다. 이때는 본문이 위 형식이 아닐 수 있습니다.

8.2 오류 코드 전체

"다시 보내기"는 같은 요청을 다시 보내도 되는지입니다. "다른 서버"는 게이트웨이가 여러 대일 때 다른 서버로 보내는 것입니다.

code	HTTP	뜻	다시 보내기	대처
`invalid_json`	400	본문이 올바른 JSON이 아님	아니오	JSON 문법과 문자 인코딩(UTF-8)을 확인합니다
`invalid_parameter`	400	필드 형식·범위가 틀림, `input` 없음, `Content-Type: application/json` 빠짐	아니오	`message` 의 필드 이름 (예: `speed: Input should be less than or equal to 1.9`)을 보고 고칩니다
`invalid_model`	400	모르는 `model`	아니오	`copland-cpu-tts` 를 씁니다
`invalid_voice`	400	모르는 `voice`	아니오	`GET /v1/voices` 목록에서 고릅니다
`invalid_sample_rate`	400	허용하지 않는 표본률	아니오	16000·22050·24000·32000·44100 중에서 고릅니다
`invalid_dictionary`	400	없는 사전 이름	아니오	`GET /v1/dictionaries` 를 봅니다
`invalid_ssml`	400	SSML 구문·값 오류	아니오	6.3을 봅니다
`empty_input`	400	정규화 뒤 읽을 내용이 없음 (공백만, `<audio>` 만 등)	아니오	입력을 확인합니다
`dictionary_error`	400	사전 파일을 읽지 못함 (`/v1/dictionary/reload`)	아니오	사전 파일을 고친 뒤 다시 부릅니다. 이전 사전은 유지됩니다
`keys_error`	400	키 파일을 읽지 못함 (`/v1/keys/reload`)	아니오	키 파일을 고친 뒤 다시 부릅니다. 이전 키는 유지됩니다
`invalid_api_key`	401	키가 없거나 틀림	아니오	키와 헤더 이름을 확인합니다
`admin_required`	403	관리자 전용 경로(`/v1/usage` , `/v1/usage/reset` , `/v1/license` , `/metrics` , `/v1/dictionary/reload` , `/v1/keys/reload`)에 user 키	아니오	관리자 키를 씁니다
`ip_not_allowed`	403	허용 IP 대역 밖에서 접속	아니오	운영자에게 웹 서버 주소를 허용 대역에 넣어 달라고 합니다
`license_expired`	403	라이선스 만료, 유예 기간도 지남	아니오	운영자에게 알립니다

code	HTTP	뜻	다시 보내기	대처
`not_found`	404	없는 경로	아니오	경로를 확인합니다. 시험 엔드포인트는 공개 API 만 엽니다
`method_not_allowed`	405	메서드가 틀림 (예: `GET /v1/audio/speech`)	아니오	메서드를 확인합니다
`input_too_long`	413	`input` 이 5000자를 넘음	아니오	글을 나눠 보냅니다
`daily_quota`	429	이 키의 오늘 한도(호출 IP별)를 다 씀	아니오	`Retry-After` (한국 자정까지 초) 뒤에 씁니다
`overloaded`	429	합성 대기열이 참 (첫 소리가 1초 넘게 늦어질 상황)	예, 다른 서버 우선	`Retry-After: 1` . 다른 서버로 바로, 또는 1초 뒤
`channel_limit`	429	라이선스 동시 채널을 모두 쓰는 중	예, 다른 서버 우선	`Retry-After: 1` . 다른 서버로 바로, 또는 1초 뒤
`restarting`	503	합성 워커 재기동 중	예, 다른 서버 우선	`Retry-After: 5`
`synthesis_timeout`	504	첫 문장 합성이 30초 (`TTS_SYNTH_TIMEOUT`)를 넘음	예, 다른 서버	계속되면 운영자에게 알립니다
`error`	그 밖	위에 없는 HTTP 오류	상태에 따라	`X-Request-Id` 와 함께 문의합니다

실제 응답 예입니다.

```
HTTP/1.1 400 Bad Request
{"error":{"message":"요청 본문이 올바른 JSON 이 아님 (Content-Type: application/
json)","type":"invalid_request_error","code":"invalid_json"}}

HTTP/1.1 400 Bad Request
{"error":{"message":"body: Input should be a valid dictionary or object to extract fields
from","type":"invalid_request_error","code":"invalid_parameter"}}

HTTP/1.1 400 Bad Request
{"error":{"message":"sample_rate 는 (16000, 22050, 24000, 32000, 44100) 중 하
나","type":"invalid_request_error","code":"invalid_sample_rate"}}

HTTP/1.1 413 Request Entity Too Large
{"error":{"message":"input 은 5000자 이하","type":"invalid_request_error","code":"input_too_long"}}

HTTP/1.1 429 Too Many Requests
retry-after: 1
{"error":{"message":"합성 대기열이 가득 참","type":"rate_limit_error","code":"overloaded"}}

HTTP/1.1 429 Too Many Requests
retry-after: 1
{"error":{"message":"라이선스 동시 채널 20개를 모두 쓰는
중","type":"rate_limit_error","code":"channel_limit"}}

HTTP/1.1 429 Too Many Requests
retry-after: 1094
x-quota-remaining: 0
{"error":{"message":"오늘 이 주소의 사용 한도(5회)를 모두 썼습니다. 내일 다시 이용해 주세
요.","type":"rate_limit_error","code":"daily_quota"}}
```

`restarting`, `synthesis_timeout`, `license_expired`, `dictionary_error`, `keys_error`는 정상 운영 중에는 일부러 일으킬 수 없어 코드의 문구를 옮겼습니다.

code	message
`restarting`	`합성 워커 재기동 중`
`synthesis_timeout`	`문장 합성이 30.0초를 넘김`
`license_expired`	`라이선스 만료 (2027-12-31, 유예 14일 지남)`

8.3 검사 순서

합성 요청은 다음 순서로 검사합니다. 앞에서 걸리면 뒤는 보지 않습니다.

합성 요청을 검사하는 순서

위에서부터 확인하고, 걸리면 그 오류를 돌려줍니다. 뒤 단계는 보지 않습니다.

1 접근

- 허용 IP — `403 ip_not_allowed`
- API 키 — `401 invalid_api_key`

2 요청 값

- 필드 형식 — `400 invalid_parameter · invalid_json` `413 input_too_long`
- model — `400 invalid_model`
- voice — `400 invalid_voice`
- sample_rate — `400 invalid_sample_rate`
- dictionary — `400 invalid_dictionary`
- SSML — `400 invalid_ssml`
- 읽을 내용이 있는가 — `400 empty_input`

3 서버 상태

- 라이선스 만료 — `403 license_expired`
- 워커 재기동 중 — `503 restarting`
- 대기열이 가득 참 — `429 overloaded`
- 동시 채널 상한 — `429 channel_limit`
- 키별 하루 한도 — `429 daily_quota`

4 합성

- 첫 문장 합성 — `504 synthesis_timeout`

↓

200 — 첫 문장부터 소리를 흘려 보냅니다

합성 요청 검사 순서

9. 한도와 남용 방지

9.1 동시 채널 (라이선스)

- 라이선스마다 **동시 채널 수**가 정해져 있습니다. `GET /v1/license` 의 `entitlements.channels` 입니다.
- 합성 요청 하나가 채널 하나를 씁니다. 채널은 **클라이언트가 소리를 다 재생할 때까지** 차지한 것으로 셉니다.
  - 서버는 실시간보다 빨리 보냅니다. 그래도 채널은 "첫 소리 시각 + 음성 길이"까지 잡혀 있습니다.
  - `stream: false` 면 "응답을 다 보낸 시각 + 음성 길이"까지입니다.
  - 클라이언트가 중간에 끊거나 오류가 나면 채널을 바로 돌려줍니다.
- 채널이 다 차면 429 `channel_limit` 입니다(`Retry-After: 1`).

9.2 과부하

- 새 요청의 첫 소리가 1초(`TTS_MAX_WAIT`)를 넘게 늦어질 것 같으면 바로 429 `overloaded` 로 거절합니다.
- 재생 중인 다른 요청이 끊길 위험이 보여도 새 요청을 거절합니다. 이미 재생 중인 고객을 먼저 지키려는 것입니다.
- 거절은 즉시 옵니다. 클라이언트는 기다리지 않고 다른 서버로 보낼 수 있습니다.
- 캐시에 있는 문장으로 시작하는 요청은 과부하 검사를 받지 않습니다.

9.3 키별 하루 한도

- 운영자가 키에 `daily_per_ip=N` 옵션을 걸면, 그 키는 **호출 IP마다 하루 N회**까지 합성할 수 있습니다. 공개 데모처럼 불특정 다수가 쓰는 중계용입니다.
- **받아들인** `/v1/audio/speech` 요청만 셉니다. 한도 검사는 다른 모든 검사 뒤에 합니다(8.3 검사 순서).
  - 입력 오류(400), 재기동(503 `restarting`), 과부하(429 `overloaded`), 채널 상한(429 `channel_limit`)으로 거절된 요청은 세지 않습니다.
  - `/v1/normalize` ·조회 API도 세지 않습니다.
- 응답 헤더 `X-Quota-Limit` · `X-Quota-Remaining` 으로 한도와 남은 횟수를 알려 줍니다.
- 한도를 넘으면 429 `daily_quota` 이고, `Retry-After` 는 한국 시각(KST) 자정까지 남은 초입니다.
- 하루는 서버·컨테이너 시간대와 관계없이 한국 자정에 초기화됩니다.
- 횟수는 서버 메모리에 둡니다. 서버를 다시 띄우면 0부터 셉니다. 서버가 여러 대면 서버마다 따로 셉니다.
- 게이트웨이 앞에 프록시를 두면, 운영자가 그 프록시를 믿을 주소(`TTS_TRUST_PROXY`)로 지정해야 실제 호출 IP로 셉니다. 지정하지 않으면 프록시 주소 하나로 셉니다.
  - 게이트웨이는 믿는 프록시가 넘긴 `X-Real-IP` 만 씁니다. `X-Forwarded-For` 는 꾸밀 수 있어 쓰지 않습니다.
  - nginx 설정 예: `proxy_set_header X-Real-IP $remote_addr;`
- 시험 엔드포인트는 이 방식으로 IP당 하루 100회입니다.

실제 확인(한도 5회 키로 6번 호출):

```
HTTP/1.1 200 OK   x-quota-limit: 5  x-quota-remaining: 4
HTTP/1.1 200 OK   x-quota-limit: 5  x-quota-remaining: 3
HTTP/1.1 200 OK   x-quota-limit: 5  x-quota-remaining: 2
HTTP/1.1 200 OK   x-quota-limit: 5  x-quota-remaining: 1
HTTP/1.1 200 OK   x-quota-limit: 5  x-quota-remaining: 0
HTTP/1.1 429 Too Many Requests   retry-after: 1094  x-quota-remaining: 0
```

9.4 입력 길이

- `input` 은 5000자까지입니다. SSML이면 태그까지 셉니다.
- 넘으면 413 `input_too_long` 입니다. 긴 문서는 문단 단위로 나눠 여러 번 보냅니다.

10. 재시도와 장애 전환

게이트웨이를 2대 이상 두면 한 대가 바쁘거나 멈춰도 서비스가 이어집니다. 클라이언트는 다음 규칙으로 다시 보냅니다. Java SDK가 쓰는 규칙과 같습니다.

10.1 다시 보내는 경우

상황	하는 일
연결 실패 (거부, 연결 시간 초과, 이름 해석 실패, 응답 전에 연결이 끊김)	바로 다음 서버로. 연결 단계 실패면 그 서버는 5초쯤 건너뜁니다
429 `overloaded` , 429 `channel_limit`	다음 서버로. 모두 거절하면 `Retry-After` (1초)만큼 쉬고 다시
503 `restarting`	다음 서버로. 같은 서버는 `Retry-After` (5초) 뒤
504 `synthesis_timeout`	다음 서버로
502 (프록시가 낸 오류)	다음 서버로

- 시도 횟수는 서버 수만큼(최소 2번) 정도로 제한합니다.
- 연결 시간 제한은 짧게(예: 2초) 둡니다. 죽은 서버를 빨리 건너뜁니다.
- 읽기 시간 제한은 "바이트 사이 최대 간격"으로 30초쯤 둡니다. 첫 문장 합성 상한이 30초이기 때문입니다.

10.2 다시 보내지 않는 경우

- 400, 401, 403, 404, 405, 413: 요청이나 설정이 틀린 것입니다. 다시 보내도 같은 결과입니다.
- 429 `daily_quota` : `Retry-After` 가 몇 시간일 수 있습니다. 다시 보내지 않고 사용자에게 알립니다.
- **응답 시간 초과:** 연결은 됐는데 읽기 시간 제한 안에 응답이 오지 않은 경우입니다. 서버가 아직 합성 중일 수 있어, 다른 서버로 보내면 같은 글을 두 번 합성하게 됩니다. 다시 보내지 않고, 그 서버를 건너뛰지도 않습니다.
- Java SDK 0.2.1은 이 규칙을 따릅니다. `daily_quota` 와 응답 시간 초과(`response_timeout`)는 `isRetryable()` 이 false입니다.

10.3 소리를 받기 시작한 뒤에는 다시 보내지 않습니다

- 200 응답을 받고 소리를 재생하기 시작했다면, 도중에 끊겨도 **자동으로 다시 보내지 않습니다.**
- 다시 보내면 처음부터 다시 읽습니다. 고객은 앞부분을 두 번 듣게 됩니다. 대면 녹취에서는 녹취 내용도 어긋납니다.
- 대신 다음과 같이 합니다.
  - 화면에 "음성 재생이 중단되었습니다"를 보이고 "다시 듣기" 버튼을 둡니다.
  - 로그에 `X-Request-Id` 를 남깁니다.

- 응답 상태는 첫 문장을 만든 뒤에 옵니다. 200을 받기 전의 실패는 10.1·10.2의 표대로 처리합니다.

10.4 openai 패키지를 쓸 때

- `openai` 패키지는 기본으로 429·5xx·연결 실패를 같은 주소로 2번 다시 보냅니다.
- 서버가 1대면 이 동작으로 충분합니다. 다만 `daily_quota` 도 다시 보내므로, 공개 서비스에서는 예외를 잡아 사용자에게 알립니다.
- 서버가 여러 대면 `max_retries=0` 으로 끄고, 위 규칙대로 서버를 바꿔 가며 직접 부릅니다.

11. 언어별 예제

각 예제는 설치 서버(0.3.4)나 시험 엔드포인트에서 실제로 돌려 확인했습니다.

11.1 curl

```
URL=http://<서버>:9300            # 시험: https://copland.ai/tts/sandbox (키 헤더 빼도 됨)
KEY=ctts_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

# 상태 확인 (키 불필요)
curl -s $URL/health

# 읽기 미리보기
curl -s $URL/v1/normalize -H "Authorization: Bearer $KEY" -H 'Content-Type: application/json' \
  -d '{"input":"문의는 1588-1234로 주세요."}'

# 스트리밍 WAV 저장 (헤더 크기 칸이 0xFFFFFFFF)
curl -s $URL/v1/audio/speech -H "Authorization: Bearer $KEY" -H 'Content-Type: application/json' \
  -d '{"input":"안녕하세요. 상담원 연결 전에 안내 말씀 드립니다."}' -o stream.wav

# 길이가 정확한 WAV (stream=false)
curl -s $URL/v1/audio/speech -H "Authorization: Bearer $KEY" -H 'Content-Type: application/json' \
  -d '{"input":"안녕하세요.","stream":false,"voice":"M1","sample_rate":16000}' -o hello16k.wav

# PCM 받아 WAV 로 바꾸기
curl -s $URL/v1/audio/speech -H "Authorization: Bearer $KEY" -H 'Content-Type: application/json' \
  -d '{"input":"안녕하세요.","response_format":"pcm","sample_rate":32000}' -o a.pcm
ffmpeg -v error -f s16le -ar 32000 -ac 1 -i a.pcm a.wav

# 첫 바이트 시간 재기
curl -s -o /dev/null $URL/v1/audio/speech -H "Authorization: Bearer $KEY" -H 'Content-Type:
         application/json' \
  -d '{"input":"오늘 기준 포인트 잔액은 12,340점입니다. 감사합니다."}' -w 'ttfb=%{time_starttransfer}
         s total=%{time_total}s\n'
```

확인 방법: `ffprobe -v error -show_entries format=duration -of csv=p=0 a.wav` 가 음성 길이(초)를 보입니다. 마지막 명령은 캐시에 없는 문장이면 첫 문장 합성 시간이, 두 번째부터는 수 ms가 찍힙니다.

11.2 Python (httpx, 스트리밍 재생·저장)

받는 대로 재생하면서 WAV 파일로 저장합니다. PCM으로 받아 Python `wave` 모듈로 쓰므로 헤더 길이가 정확합니다. 재생은 `ffplay` (FFmpeg)를 씁니다.

```python
"""COPLAND TTS — httpx 로 스트리밍 받아 재생하면서 WAV 로 저장한다.
사용: python tts_stream.py "읽을 문장" out.wav [--play]
"""
import os, subprocess, sys, time, wave
import httpx

URL = os.environ.get("COPLAND_TTS_URL", "http://localhost:9300")
KEY = os.environ.get("COPLAND_TTS_KEY", "")
RATE = 32000

def speak(text, out_path, play=False):
    body = {"model": "copland-cpu-tts", "input": text, "voice": "F1",
            "response_format": "pcm", "sample_rate": RATE, "stream": True}
    player = None
    if play:  # 받는 대로 재생 (ffplay 가 s16le 모노를 표준 입력으로 받는다)
        player = subprocess.Popen(["ffplay", "-nodisp", "-autoexit", "-loglevel", "error",
                                   "-f", "s16le", "-ar", str(RATE), "-ch_layout", "mono", "-i",
          "-"],
                                  stdin=subprocess.PIPE)
    t0 = time.perf_counter()
    first = None
    with httpx.Client(timeout=httpx.Timeout(30.0, connect=2.0)) as c, \
         c.stream("POST", f"{URL}/v1/audio/speech", json=body,
                  headers={"Authorization": f"Bearer {KEY}"}) as r:
        if r.status_code != 200:
            r.read()
            err = r.json()["error"]
            raise RuntimeError(f"{r.status_code} {err['code']}: {err['message']}
          (rid={r.headers.get('x-request-id')})")
        print("문장 수", r.headers["x-sentences"], "요청 번호", r.headers["x-request-id"])
        with wave.open(out_path, "wb") as w:   # wave 가 닫을 때 헤더 길이를 바르게 쓴다
            w.setnchannels(1); w.setsampwidth(2); w.setframerate(RATE)
            for chunk in r.iter_bytes():
                if first is None:
                    first = time.perf_counter() - t0
                    print(f"첫 소리까지 {first * 1000:.0f}ms")
                w.writeframes(chunk)
                if player:
                    player.stdin.write(chunk)
    if player:
        player.stdin.close()
        player.wait()
    print(f"저장 {out_path} ({time.perf_counter() - t0:.2f}s)")

if __name__ == "__main__":
    speak(sys.argv[1], sys.argv[2], "--play" in sys.argv)
```

실행:

```
COPLAND_TTS_URL=http://<서버>:9300 COPLAND_TTS_KEY=ctts_xxxxxxxx... \
  python tts_stream.py "안녕하세요. 상담원 연결 전에 안내 말씀 드립니다. 통화 내용은 녹음됩니다."
         out.wav --play
```

예상 출력:

```
문장 수 3 요청 번호 c9e292de8bce4ad1
첫 소리까지 45ms
저장 out.wav (8.68s)
```

- `--play`를 주면 재생 시간만큼 걸립니다. 위 예는 7.7초 음성입니다.
- 키가 틀리면 `RuntimeError: 401 invalid_api_key: API 키가 없거나 맞지 않음 (rid=...)` 이 납니다.
- 첫 소리 시간은 캐시에 있는 문장이라 짧게 나왔습니다.
- `ffplay`가 FFmpeg 5.0보다 오래된 판이면 `-ch_layout mono` 대신 `-ac 1`을 씁니다.

11.3 JavaScript (브라우저, Web Audio로 PCM 스트리밍 재생)

`fetch`로 PCM을 받아, 청크가 올 때마다 Web Audio 버퍼로 이어 붙여 재생합니다. 온라인 데모(https://copland.ai/tts/demo/)가 쓰는 방식을 간추린 것입니다.

- 브라우저는 게이트웨이를 직접 부르지 않습니다. 같은 출처의 자사 중계 서버(`/tts`, 11.4)를 부릅니다.
  - 게이트웨이와 시험 엔드포인트는 CORS 헤더를 보내지 않습니다. 다른 출처의 페이지에서 직접 부르면 브라우저가 막습니다.
  - 시험 엔드포인트에 CORS를 열면 다른 사이트가 방문자의 하루 한도를 대신 쓸 수 있습니다. 그래서 CORS는 열지 않습니다.
  - API 키를 브라우저에 두지 않기 위해서도 중계가 필요합니다.
- `AudioContext`는 사용자의 클릭 안에서 처음 만들어야 소리가 납니다.

```
// 브라우저: 받는 대로 재생 (PCM s16le 모노 → Web Audio)
// 같은 출처의 중계 서버(/tts)를 부른다. API 키는 중계 서버에만 있다.
const API = "/tts";
const RATE = 32000;
let ctx = null, ctrl = null, sources = [];

async function speak(text, voice = "F1", api = API) {
  stop();
  ctx = ctx || new AudioContext();              // 사용자 클릭 안에서 처음 만들어야 소리가 난다
  if (ctx.state === "suspended") await ctx.resume();
  ctrl = new AbortController();
  const t0 = performance.now();
  const r = await fetch(api, {
    method: "POST", signal: ctrl.signal,
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ text, voice }),            // 중계 서버가 PCM 32kHz 로 받아 넘긴다
  });
  if (!r.ok) {
    const e = (await r.json().catch(() => null))?.error;   // 게이트웨이 오류 본문을 중계가 그대로 넘
        긴다
    throw new Error(e?.code ? `${e.code}: ${e.message}` : `HTTP ${r.status}`);
  }
  const reader = r.body.getReader();
  let carry = new Uint8Array(0), next = 0, first = true;
  for (;;) {
    const { done, value } = await reader.read();
    if (done) break;
    let buf = value;
    if (carry.length) { const m = new Uint8Array(carry.length + buf.length); m.set(carry);
        m.set(buf, carry.length); buf = m; }
    const even = buf.length - (buf.length %
        2);   // 16비트 표본이 청크 경계에서 잘릴 수 있다 → 남은 1바이트는 다음 청크로
    carry = buf.slice(even);
    if (!even) continue;
    const i16 = new Int16Array(buf.buffer, buf.byteOffset, even / 2);
    const ab = ctx.createBuffer(1, i16.length, RATE);
    const ch = ab.getChannelData(0);
    for (let i = 0; i < i16.length; i++) ch[i] = i16[i] / 32768;
    const src = ctx.createBufferSource();
    src.buffer = ab; src.connect(ctx.destination);
    if (first) { first = false; next = ctx.currentTime + 0.05; console.log("첫 소리",
        Math.round(performance.now() - t0), "ms"); }
    next = Math.max(next, ctx.currentTime + 0.02);  // 늦게 온 청크는 지금 바로
    src.start(next); next += ab.duration;            // 앞 청크 끝에 이어 붙인다
    sources.push(src);
  }
  return next;  // 재생이 끝나는 시각 (ctx.currentTime 기준)
}

function stop() {
```

```
  if (ctrl) { ctrl.abort(); ctrl = null; }          // 끊으면 서버도 남은 문장 합성을 멈춘다
  for (const s of sources) { try { s.stop(); } catch (_) {} }
  sources = [];
}

// 사용: <button onclick="speak('문의는 1588-1234로 주세요.')">듣기</button>
```

- `Int16Array` 는 플랫폼의 바이트 순서를 씁니다. 브라우저가 도는 PC·모바일은 모두 리틀 엔디언이라 그대로 맞습니다.

11.4 Node.js (서버에서 중계)

브라우저는 자사 웹 서버의 `/tts` 를 부르고, 웹 서버가 API 키를 붙여 게이트웨이로 보냅니다. 받은 소리는 그대로 흘려보냅니다. 게이트웨이가 여러 대면 10장 규칙으로 다음 서버로 넘깁니다. Node 18 이상, 외부 패키지 없이 동작합니다.

```
// relay.mjs — COPLAND TTS 중계. 브라우저는 이 서버의 /tts 를 부르고, API 키는 서버에만 둔다.
import http from "node:http";
import { Readable } from "node:stream";

const GATEWAYS = (process.env.COPLAND_TTS_URLS || "http://localhost:9300").split(",");
const KEY = process.env.COPLAND_TTS_KEY;
const RETRY = new Set([429, 502, 503, 504]);   // 다른 게이트웨이로 다시 보내도 되는 상태

async function callGateway(body) {
  let last;
  for (const base of GATEWAYS) {
    try {
      const r = await fetch(base + "/v1/audio/speech", {
        method: "POST",
        headers: { "Content-Type": "application/json", "Authorization": "Bearer " + KEY },
        body: JSON.stringify(body),
        signal: AbortSignal.timeout(30000),
      });
      if (r.ok || !RETRY.has(r.status)) return r;  // 성공, 또는 다시 보내도 소용없는 오류
          (400·401·403·413)
      last = r;
    } catch (e) { last = e; }                      // 연결 실패 → 다음 서버
  }
  if (last instanceof Response) return last;
  throw last;
}

http.createServer(async (req, res) => {
  if (req.method !== "POST" || req.url !== "/tts") { res.writeHead(404).end(); return; }
  let raw = "";
  for await (const c of req) raw += c;
  let text, voice;
  try { ({ text, voice = "F1" } = JSON.parse(raw)); } catch { res.writeHead(400).end(); return; }
  try {
    const r = await callGateway({ input: text, voice, response_format: "pcm", sample_rate:
          32000 });
    const headers = { "Content-Type": r.headers.get("content-type"), "Cache-Control": "no-
          store" };
    for (const h of ["x-request-id", "x-sentences", "x-sample-rate"]) if (r.headers.get(h))
          headers[h] = r.headers.get(h);
    res.writeHead(r.status, headers);
    const body = Readable.fromWeb(r.body);
    req.on("close", () => body.destroy());        // 브라우저가 끊으면 게이트웨이 연결도 끊는다 → 합
          성 중단
    body.pipe(res);                               // 받는 대로 흘려보낸다 (첫 소리가 늦지 않다)
  } catch (e) {
    res.writeHead(503, { "Content-Type": "application/json" }).end(JSON.stringify({ error:
          "tts_unavailable" }));
```

```
  }
}).listen(Number(process.env.PORT || 8080), () => console.log("relay :" + (process.env.PORT ||
          8080)));
```

실행과 확인:

```
COPLAND_TTS_URLS=http://10.0.0.11:9300,http://10.0.0.12:9300 COPLAND_TTS_KEY=ctts_xxxxxxxx... node
          relay.mjs
curl -s -D - -o a.pcm localhost:8080/tts -H 'Content-Type: application/json' -d '{"text":"문의는
          1588-1234로 주세요."}'
```

예상 출력(헤더):

```
HTTP/1.1 200 OK
Content-Type: audio/pcm;rate=32000;channels=1;format=s16le
Cache-Control: no-store
x-request-id: 7f698cf6d2a64e27
x-sentences: 1
x-sample-rate: 32000
Transfer-Encoding: chunked
```

- 첫 주소가 꺼져 있어도 둘째 주소로 넘어가 200을 받는 것을 확인했습니다.
- 11.3의 브라우저 예제가 이 중계를 부릅니다. 요청 본문은 `{"text":"...","voice":"F1"}` 입니다.
- 게이트웨이 오류(예: 빈 문장 400)는 상태와 본문을 그대로 전달합니다.
- 실제 서비스에서는 이 중계에 자사 로그인 확인과 요청 수 제한을 붙입니다.

11.5 Java

Java는 SDK(`copland-tts-client-0.2.1.jar` , Java 8 이상, 외부 의존성 없음)를 씁니다. 장애 전환·재시도·스트리밍을 SDK가 처리합니다. Java SDK 매뉴얼를 봅니다.

12. 변경 내역 (API 관련)

버전	변경
0.3.2	`/v1/usage` · `/v1/license` · `/metrics` 를 관리자 키 전용으로 바꿈 (user 키는 403 `admin_required`)
0.3.2	PCM 응답 `Content-Type` 을 `audio/pcm;rate=...;channels=1;format=s16le` 로 바꿈 (이전 `audio/L16;...`)
0.3.2	JSON이 아닌 본문에 400 `invalid_json`
0.3.2	`/health` 가 라이선스 만료(유예 지남) 때 503 `{"status":"license_expired"}`
0.3.2	읽기 규칙: 천 단위가 아닌 쉼표를 남김, 범위 + 단위를 앞뒤 모두 단위로 읽음(`3~5개` → 세 개에서 다섯 개, `9시~10시` → 아홉 시에서 열 시)
0.3.2	SSML: `say-as` · `sub` 뒤 띄어쓰기를 원문대로 지킴, `input_type: "ssml"` 이면 `<speak>` 가 없어도 감쌈
0.3.2	`daily_per_ip` : 받아들인 요청만 셈(과부하·채널 상한·재기동 거절은 제외), 한국 자정에 초기화. 프록시 뒤 호출 IP는 `X-Real-IP` 만 씀
0.3.2	허용 IP 대역을 지정해도 루프백은 늘 허용. `/metrics` 의 `audio_sec` 에 문장 사이 쉼 포함
SDK 0.2.1	`daily_quota` ·응답 시간 초과(`response_timeout`)는 다시 보내지 않음. `SpeechStream.getHeader()` · `getQuotaRemaining()` 추가
0.3.1	키 옵션 `daily_per_ip` (호출 IP별 하루 합성 한도) 추가. 초과 시 429 `daily_quota` , 헤더 `X-Quota-Limit` · `X-Quota-Remaining` . `/v1/normalize` 는 세지 않음
0.3.1	프록시 뒤 실제 호출 IP 인식(`TTS_TRUST_PROXY` , `X-Real-IP`). 허용 IP 대역은 직접 붙은 주소 기준
0.3.1	공개 시험 엔드포인트 `https://copland.ai/tts/sandbox` (키 불필요, IP당 하루 100회)
0.3.0	장비 고정 라이선스. 동시 채널을 "재생이 끝날 때까지"로 세고 넘으면 429 `channel_limit` . 만료 후 유예 기간에 `X-License-Warning` , 지나면 403 `license_expired`
0.3.0	`/health` 에 `commit` (빌드 커밋)과 라이선스 요약
0.2.0	과부하 429 `overloaded` (예상 대기 1초 초과·재생 중 요청 지연), 워커 재기동 중 503 `restarting`
0.2.0	API 키(user/admin), 허용 IP 대역
0.2.0	SSML 부분집합, 요청 필드 `sentence_pause_ms` · `dictionary` · `volume` , 이름 있는 사전
0.2.0	키별 사용량(`/v1/usage`), Prometheus 지표, OpenAI 형식 오류, `X-Request-Id`

버전	변경
0.1.0	OpenAI 호환 `/v1/audio/speech`, 사용자 사전·정규화, 문장 캐시

서드파티 구성 요소의 라이선스는 패키지의 `NOTICE.md` 를 봅니다.

문의: COPLAND 기술지원 (연락처는 계약서 참조)

COPLAND TTS. 도입 및 기술 문의는 COPLAND 담당자에게 연락해 주십시오.