COPLAND

기술 문서

COPLAND TTS

운영 매뉴얼

설치 이후의 운영 업무를 다룹니다. API 키 발급, 사용자 사전 관리, 상태 점검 방법을 설명합니다.

적용 버전	문서 갱신일	대상 독자
COPLAND TTS 0.3.4(게이트웨이) Java SDK 0.2.1	2026-10-12	고객사 서버 운영자

차례

1. 일상 운영
   - 1.1 상태 확인
   - 1.2 로그
   - 1.3 재기동과 설정 반영
   - 1.4 시간대
2. API 키 관리
   - 2.1 키의 종류
   - 2.2 새 키 만들기
   - 2.3 키 파일 형식
   - 2.4 키 다시 읽기(`/v1/keys/reload` , admin)
   - 2.5 키 폐기
3. 접근 제어
   - 3.1 허용 IP(`TTS_ALLOW`)
   - 3.2 프록시 뒤에 둘 때(`TTS_TRUST_PROXY`)
4. 사용자 사전
   - 4.1 형식
   - 4.2 우선순위
   - 4.3 이름 있는 사전
   - 4.4 반영(`/v1/dictionary/reload` , admin)
   - 4.5 작성 요령 — 정규화 미리보기
5. 예열 목록
   - 5.1 언제 쓰는가
   - 5.2 형식
   - 5.3 캐시에 맞는 조건
   - 5.4 반영과 확인
6. 모니터링
   - 6.1 `/health` — 부하 분산기·상태 검사용
   - 6.2 `/metrics` — JSON
   - 6.3 `/metrics?format=prometheus`
   - 6.4 라이선스 상태 — `/v1/license`
   - 6.5 경보 기준 예
7. 사용량 정산
   - 7.1 조회(`/v1/usage`)
   - 7.2 초기화(`/v1/usage/reset` , admin)

8. 용량·성능

8.1 채널 산정

8.2 과부하 때의 동작

8.3 동시 채널 라이선스의 의미

9. 백업

10. 보안 점검표

- 적용 버전: **COPLAND TTS 0.3.4**(게이트웨이), Java SDK 0.2.1
- 문서 갱신일: 2026-10-12
- 대상 독자: 고객사 서버 운영자
- 관련 문서: 설치 매뉴얼, API 개발자 매뉴얼, Java SDK 매뉴얼, 라이선스 발급 포털 사용 안내

이 문서는 설치가 끝난 COPLAND TTS 서버를 운영하는 방법을 다룹니다. 설치 경로는 기본값 `/opt/copland-tts`, 포트는 기본값 `9300`으로 씁니다. 예시의 API 키 `ctts_xxxxxxxx...`는 실제 키로 바꿔 넣습니다.

명령 예시에서 자주 쓰는 값을 변수로 두면 편합니다.

```
TTS=http://127.0.0.1:9300
USER_KEY=ctts_xxxxxxxx...      # 일반(user) 키
ADMIN_KEY=ctts_yyyyyyyy...     # 관리자(admin) 키
```

1. 일상 운영

1.1 상태 확인

1. 설치 프로그램으로 확인합니다. 설치 경로에 복사된 `install.sh` 를 씁니다.

   ```
   sudo bash /opt/copland-tts/install.sh status
   ```

   컨테이너 목록과 `/health` 응답이 나옵니다(예시, 일부 줄임, 커밋·시각은 예시).

   ```
   COPLAND TTS 0.3.4 (v0.3.4, 2026-10-12T15:00:00Z)
   NAME          IMAGE                  ...   STATUS
   copland-tts   copland-tts-gw:0.3.4   ...   Up 2 hours (healthy)
   {"status":"ok","version":"0.3.4","commit":"1a2b3c4d5e","workers":6,"license":
   {"license_id":"L-000123","state":"valid","expires":"2027-12-31","channels":150}}
   ```

2. Docker로 직접 확인할 수도 있습니다.

   ```
   sudo docker ps --filter name=copland-tts
   curl -s $TTS/health
   ```

- `STATUS` 에 `(healthy)` 가 보이면 정상입니다. 컨테이너가 15초마다 스스로 `/health` 를 확인합니다.
- `/health` 의 `status` 가 `ok` 이고 `license.state` 가 `valid` 이면 정상입니다.
- `install.sh` 는 `.env` 의 `TTS_PORT` 를 읽어 그 포트의 `/health` 를 확인합니다.

1.2 로그

```
sudo docker logs -f copland-tts              # 실시간으로 보기
sudo docker logs --since 1h copland-tts      # 최근 1시간
sudo docker logs copland-tts 2>&1 | grep 'rid=95e55489eeab4ac9'   # 요청 번호로 찾기
```

- 로그는 컨테이너 로그(json-file)에 50MB × 5개로 순환 저장됩니다.
- **입력 문장은 로그에 남지 않습니다**(고객 개인정보 보호). 글자 수, 문장 수, 지연만 남습니다.

접근 로그 형식은 두 가지입니다.

합성 요청(`/v1/audio/speech`)이 끝나면 한 줄을 남깁니다.

```
2026-10-12 23:41:14,143 INFO gateway.access: rid=95e55489eeab4ac9 key=web01 ip=127.0.0.1 speech voice=F1 chars=26 sents=2 audio=4.9s ttfb=232ms total=640ms ok
```

필드	뜻
`rid`	요청 번호. 응답 헤더 `X-Request-Id` 와 같습니다. 요청에 `X-Request-Id` 를 넣으면 그 값을 씁니다(SDK `requestId`)
`key`	API 키 이름
`ip`	호출 주소. 믿는 프록시 뒤면 실제 호출 주소(3.2절)
`voice`	화자
`chars`	입력 글자 수
`sents`	문장 수
`audio`	보낸 음성 길이(초, 문장 사이 쉼 포함)
`ttfb`	첫 소리까지 걸린 시간. 캐시에서 나가면 0ms
`total`	마지막 소리까지 걸린 시간
마지막 값	`ok` 정상, `aborted` 클라이언트가 중간에 끊음, `error` 합성 중 오류

오류 응답(HTTP 400 이상)은 모든 경로에 대해 한 줄을 남깁니다.

```
2026-10-12 23:21:06,801 INFO gateway.access: rid=fe6f934b70ca4c0f key=web01 ip=127.0.0.1 POST /v1/audio/speech → 413
2026-10-12 23:21:12,631 INFO gateway.access: rid=54171f2600734149 key=- ip=127.0.0.1 GET /v1/voices → 401
```

`key=-` 는 키가 없거나 맞지 않은 요청입니다. `ip` 는 위와 같습니다.

주의해서 볼 로그는 다음과 같습니다.

로그 문구	뜻
`ERROR gateway.pool: 합성 워커가 죽었다 — 풀 재기동 (시도 1)`	합성 워커가 비정상 종료해 다시 띄움. 수 초 동안 503
`ERROR gateway: rid=... 합성 중 오류: …`	소리를 보내던 중 오류. 그 요청은 중간에 끊김
`WARNING gateway: 라이선스 만료 … — 유예 N일 뒤 합성 거부`	라이선스 유예 기간 중
`WARNING gateway: 예열: 모르는 화자 … — 건너뜀`	예열 목록의 화자 이름이 틀림
`WARNING gateway: 사전 파일이 없어 건너뜀: …`	고객사 사전 파일이 없음
`CRITICAL gateway: 라이선스: …`	라이선스 문제로 기동 실패(설치 매뉴얼 «문제 해결»)

1.3 재기동과 설정 반영

상황	명령
단순 재기동	`sudo docker restart copland-tts`
`.env`, `docker-compose.yml` 을 고친 뒤	`cd /opt/copland-tts` 후 `sudo docker compose up -d`
멈추기 / 다시 띄우기	`cd /opt/copland-tts` 후 `sudo docker compose stop` / `sudo docker compose up -d`
사전·예열 목록을 고친 뒤	재기동하지 않고 `POST /v1/dictionary/reload` (4.4절)
API 키 파일을 고친 뒤	재기동하지 않고 `POST /v1/keys/reload` (2.4절)
라이선스 키를 바꾼 뒤	`sudo bash /opt/copland-tts/install.sh key`

- 재기동하면 합성 워커가 모델을 다시 올리는 데 수 초가 걸립니다. 그동안 `/health` 는 응답하지 않거나 503을 돌려줍니다.
- 재기동하면 문장 캐시, 사용량(7장), 호출 IP 일일 한도 횟수가 0으로 돌아갑니다. 예열 목록은 기동 때 다시 합성합니다.
- 이중화 구성이면 한 대씩 재기동합니다. SDK가 다른 서버로 넘깁니다.

1.4 시간대

- 컨테이너는 한국 시각(`TZ=Asia/Seoul`)으로 돕니다. 로그 시각과 사용량의 `since` 는 한국 시각입니다.
- 키별 호출 IP 일일 한도(`daily_per_ip`)는 시간대 설정과 관계없이 한국 자정(KST)에 초기화됩니다.

2. API 키 관리

2.1 키의 종류

역할	쓸 수 있는 API	주는 대상
`user`	합성(`/v1/audio/speech`), 정규화 미리보기 (`/v1/normalize`), 목록 조회(`/v1/voices` , `/v1/models` , `/v1/dictionaries`)	웹 서버
`admin`	`user` 의 모든 API + 관리자 전용 API(`/v1/dictionary/reload` , `/v1/keys/reload` , `/v1/usage` , `/v1/usage/reset` , `/v1/license` , `/metrics`)	서버 운영자, 모니터링(Prometheus)

- 요청에는 `Authorization: Bearer <키>` 또는 `X-API-Key: <키>` 헤더로 키를 넣습니다.
- `/health` 와 `/openapi.json` 은 키 없이 열립니다(부하 분산기용).
- 관리자 전용 API를 일반 키로 부르면 403 `admin_required` (`관리자 키가 필요함`)입니다. 다른 키의 사용량, 장비키·고객명, 지표를 보호하기 위해서입니다.
- 설치 프로그램은 첫 설치 때 키 2개를 만듭니다: `web01` (user, 웹 서버용)과 `ops` (admin, 운영자용). 둘 다 설치 화면에서 한 번만 보입니다.

2.2 새 키 만들기

키는 게이트웨이 이미지 안의 도구로 만듭니다.

1. 실행 중인 컨테이너에서 키를 만듭니다. 이름은 부서나 웹 서버를 알아볼 수 있게 짓습니다. 사용량 정산(7장)이 이 이름으로 나뉩니다.

   ```
   sudo docker exec copland-tts python -m gateway.auth new web02            # 일반 키
   sudo docker exec copland-tts python -m gateway.auth new ops2 --admin     # 관리자 키
   ```

   컨테이너가 떠 있지 않으면 이미지로 실행합니다(버전은 `.env` 의 `TTS_VERSION`).

   ```
   sudo docker run --rm --entrypoint python copland-tts-gw:0.3.4 -m gateway.auth new ops2 --admin
   ```

2. 출력은 다음과 같습니다.

```
클라이언트에 줄 키 (다시 볼 수 없음): ctts_xxxxxxxx...
키 파일에 넣을 줄:
ops2 sha256:7c1e90b2...    admin
```

- 첫 줄의 `ctts_...` 가 키입니다. **이때 한 번만 보입니다.** 받을 사람에게 안전하게 전달합니다.
- 셋째 줄을 키 파일에 넣습니다. 칸 사이는 탭입니다.

3. 셋째 줄을 키 파일 끝에 붙입니다. 출력의 줄을 그대로 복사해 편집기로 넣거나 다음처럼 넣습니다. `sha256:` 뒤에는 출력에 나온 64자 값을 빠짐없이 넣습니다.

```
printf 'ops2\tsha256:7c1e90b2...\tadmin\n' | sudo tee -a /opt/copland-tts/conf/keys.tsv
```

4. 반영합니다.
   - 관리자 키로 `POST /v1/keys/reload` (2.4절)를 부릅니다. 재기동은 필요 없습니다.
   - 관리자 키를 모두 잃어버렸다면 새 관리자 키 줄을 넣고 `sudo docker restart copland-tts` 로 반영합니다.
5. 확인합니다. 관리자 키로 키 목록을 다시 읽어 키 개수를 봅니다.

```
curl -s -X POST $TTS/v1/keys/reload -H "Authorization: Bearer $ADMIN_KEY"
```

예상 출력: `{"keys":3}` (키 파일의 키 개수)

2.3 키 파일 형식

`/opt/copland-tts/conf/keys.tsv` 는 한 줄에 키 하나입니다. 칸은 탭으로 나눕니다.

```
# 이름<TAB>키(sha256:...)<TAB>역할(user|admin)[<TAB>옵션]
web01    sha256:2af5e7eb...    user
ops sha256:4df0aa47...    admin
demo    sha256:1df65cd9...    user    daily_per_ip=100
```

칸	내용
1. 이름	키 이름. 로그(`key=`)와 사용량 정산에 쓰입니다
2. 키	`sha256:<값>` 형식(권장). 파일이 새도 키 원문은 알 수 없습니다. 평문 키도 받지만 쓰지 않습니다
3. 역할	`user` 또는 `admin` . 비우면 `user`
4. 옵션(선택)	`daily_per_ip=<정수>` : 호출 IP마다 하루 합성 횟수 상한

- `#` 으로 시작하는 줄과 빈 줄은 무시합니다.

- `daily_per_ip` 는 외부에 공개하는 중계(데모 페이지 등)에 겁니다. 한도를 넘으면 429 `daily_quota` 를 돌려주고, 한국 자정(KST)에 초기화됩니다. 받아들인 합성 요청만 셉니다. 과부하(`overloaded`)·채널 상한(`channel_limit`)·재기동(`restarting`)·입력 오류로 거절된 요청과 정규화 미리보기는 세지 않습니다. 횟수는 메모리에만 있어 재기동하면 0이 됩니다.
  - 한도가 걸린 키로 합성하면 응답에 `X-Quota-Limit` (한도)과 `X-Quota-Remaining` (남은 횟수) 헤더가 붙습니다.
- 역할이나 옵션이 틀리면 키 다시 읽기가 실패하고 이전 키를 그대로 씁니다(2.4절).

2.4 키 다시 읽기(`/v1/keys/reload` , admin)

키 파일을 고친 뒤 재기동 없이 반영합니다.

```
curl -s -X POST $TTS/v1/keys/reload -H "Authorization: Bearer $ADMIN_KEY"
```

결과	응답
성공	`{"keys":3}`
파일 형식 오류	400 `keys_error` : `키 파일을 읽지 못함 (이전 키 유지): …` (몇 번째 줄이 틀렸는지 나옵니다)
일반 키로 부름	403 `admin_required` : `관리자 키가 필요함`

이중화 구성이면 서버마다 키 파일을 고치고 각각 다시 읽습니다.

2.5 키 폐기

1. `conf/keys.tsv` 에서 그 키의 줄을 지웁니다.
2. `POST /v1/keys/reload` 로 반영합니다.
3. 확인합니다. 지운 키로 요청하면 401 `invalid_api_key` 가 나와야 합니다.

   ```
   curl -s $TTS/v1/voices -H "Authorization: Bearer ctts_지운키…"
   ```

   예상 출력: `{"error":{"message":"API 키가 없거나 맞지 않음","type":"authentication_error","code":"invalid_api_key"}}`

키가 샜다고 의심되면 바로 폐기하고 새 키를 발급합니다. 키 원문은 서버에 남지 않으므로 «키 다시 보기»는 없습니다.

3. 접근 제어

3.1 허용 IP(`TTS_ALLOW`)

허용한 주소에서 온 요청만 받습니다. 그 밖의 주소는 403 `ip_not_allowed` (`허용되지 않은 주소: <주소>`)입니다. `/health` 도 이 검사를 받습니다.

1. `/opt/copland-tts/.env` 에 대역(CIDR)을 쉼표로 적습니다.

```
TTS_ALLOW=10.0.0.11/32,10.0.0.12/32
```

   - 서버 자신(루프백 `127.0.0.1` , `::1`)은 따로 넣지 않아도 늘 허용됩니다. 컨테이너 상태 검사와 `install.sh` 의 확인은 막히지 않습니다.
   - 부하 분산기, 모니터링 서버가 부른다면 그 주소도 넣습니다.

2. 반영합니다.

```
cd /opt/copland-tts
sudo docker compose up -d
```

3. 확인합니다. 기동 로그에 `허용 IP ['10.0.0.11/32', ...]` 가 나옵니다. 허용하지 않은 주소에서 부르면 403이 나옵니다.

- 비워 두면 모든 주소를 받습니다. 호스트 방화벽(설치 매뉴얼 2.5절)과 함께 씁니다.
- 판단 기준은 **직접 접속한 주소**입니다. 앞에 프록시가 있으면 프록시 주소를 허용합니다.

3.2 프록시 뒤에 둘 때(`TTS_TRUST_PROXY`)

게이트웨이 앞에 nginx 같은 프록시를 두면 게이트웨이에는 프록시 주소만 보입니다. 키별 일일 한도와 오류 로그의 `ip` 에 실제 호출 주소를 쓰려면 프록시를 믿을 주소로 지정합니다.

1. `.env` 에 프록시의 IP 주소를 적습니다. 대역이 아니라 주소 그대로 적습니다. 여러 개면 쉼표로 나눕니다.

```
TTS_TRUST_PROXY=10.0.0.5
TTS_ALLOW=10.0.0.5/32
```

2. 프록시가 실제 호출 주소를 `X-Real-IP` 헤더로 넘기게 합니다(nginx 예).

```
location /tts/ {
    proxy_pass http://10.0.0.21:9300/;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_buffering off;          # 첫 소리부터 바로 흘려 보내기
}
```

3. `sudo docker compose up -d` 로 반영합니다.

- 게이트웨이는 지정한 프록시에서 온 요청에서만 `X-Real-IP` 를 읽습니다. `X-Forwarded-For` 는 클라이언트가 꾸밀 수 있어 쓰지 않습니다. **프록시에서 `X-Real-IP` 를 반드시 넣습니다.** 넣지 않으면 프록시 주소가 호출 주소로 쓰입니다.
- 다른 주소에서 온 요청의 이 헤더들은 무시합니다.

4. 사용자 사전

엔진은 한글 읽기만 정확히 읽습니다. 숫자, 금액, 날짜, 전화번호, 영문 약어는 게이트웨이가 읽기 규칙으로 바꿉니다. 규칙으로 안 되는 상품명, 전문 용어, 영문 단어는 사용자 사전에 넣습니다. 기본 읽기 규칙은 API 개발자 매뉴얼을 참조합니다.

4.1 형식

한 줄에 `표기<TAB>읽기` 를 씁니다. UTF-8로 저장합니다.

```
# 고객사 사전 — 표기<TAB>읽기
리볼빙 일부결제금액이월약정
Samsung Pay 삼성페이
toss 토스
```

- 칸 사이는 **탭**입니다. 공백으로 나누면 무시됩니다.
- `#` 뒤는 주석입니다. 줄 중간의 `#` 부터도 주석으로 봅니다.
- 대소문자를 구분합니다(`KB` 와 `kb` 는 다른 표기).
- 긴 표기부터 바꿉니다. `KB국민` 과 `KB` 가 함께 있으면 `KB국민` 이 먼저 맞습니다.
- 영문으로 시작하거나 끝나는 표기는 단어 단위로만 바꿉니다. `app` 은 `apple` 안에서 바뀌지 않습니다.
- 사전은 읽기 규칙보다 먼저 적용됩니다.

4.2 우선순위

순위	사전	파일	쓰이는 때
1	이름 있는 사전	`conf/dicts/<이름>.tsv`	요청에 `"dictionary":"<이름>"` 을 준 경우만
2	고객사 사전	`conf/customer.tsv`	늘
3	기본 사전	제품에 포함	늘

같은 표기가 여러 사전에 있으면 순위가 높은 사전의 읽기를 씁니다.

4.3 이름 있는 사전

부서나 상품마다 다른 읽기가 필요할 때 씁니다.

1. `conf/dicts/` 아래에 `<이름>.tsv` 를 만듭니다. 예: `conf/dicts/card.tsv`
2. 사전을 다시 읽습니다(4.4절). 새 파일도 이때 읽힙니다.

3. 요청에 `"dictionary":"card"` 를 줍니다. SDK는 `.dictionary("card")` 입니다.

없는 이름을 주면 400 `invalid_dictionary` (`dictionary 는 ['card'] 중 하나`)입니다.

사전 목록은 다음으로 봅니다.

```
curl -s $TTS/v1/dictionaries -H "Authorization: Bearer $USER_KEY"
```


```
{"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"]}]}
```


`default` 는 이름 없이 요청할 때 쓰는 사전이고, `entries` 는 합친 항목 수입니다. `files` 는 우선순위 순서입니다.

4.4 반영(`/v1/dictionary/reload` , admin)

`conf/customer.tsv` , `conf/dicts/` , `conf/prewarm.txt` 를 고친 뒤 재기동 없이 반영합니다.

```
curl -s -X POST $TTS/v1/dictionary/reload -H "Authorization: Bearer $ADMIN_KEY"
```

- 성공하면 4.3절의 `/v1/dictionaries` 와 같은 형식으로 새 사전 목록을 돌려줍니다.
- **문장 캐시도 비웁니다.** 읽기가 바뀌었을 수 있기 때문입니다. 예열 목록도 다시 읽어 다시 합성합니다.
- 파일을 읽지 못하면 400 `dictionary_error` (`사전을 읽지 못함 (이전 사전 유지): …`)이고 이전 사전을 그대로 씁니다.
- 이중화 구성이면 서버마다 파일을 고치고 각각 다시 읽습니다. Java SDK의 `reloadDictionaries` 는 모든 서버에 보냅니다.

4.5 작성 요령 — 정규화 미리보기

사전에 넣을 말은 `/v1/normalize` 로 찾습니다. 엔진에 실제로 들어갈 문장을 보여 주고, 읽기 규칙으로 바꾸지 못한 영문 단어를 `unknown_latin` 으로 알려 줍니다. **`unknown_latin` 이 사전 후보입니다.** 미리보기는 합성하지 않으므로 서버 부하가 거의 없고, 일일 한도에도 세지 않습니다.

1. 실제 안내문을 넣어 봅니다.

   ```
   curl -s $TTS/v1/normalize -H "Authorization: Bearer $USER_KEY" -H 'Content-Type: application/
            json' \
     -d '{"input":"Samsung Pay 또는 toss 로 결제하시면 KB 포인트가 적립됩니다."}'
   ```

   응답(일부)은 다음과 같습니다.

```
{"normalized": "Samsung Pay 또는 toss 로 결제하시면 케이비 포인트가 적립됩니다.",
 "unknown_latin": ["Pay", "Samsung", "toss"], …}
```

- 대문자 약어 `KB` 는 규칙대로 «케이비»로 읽습니다.
- `Samsung` , `Pay` , `toss` 는 읽는 법을 몰라 그대로 남았습니다. 이 단어들을 사전에 넣습니다.

2. 숫자, 날짜, 전화번호가 원하는 대로 읽히는지도 확인합니다.

```
curl -s $TTS/v1/normalize -H "Authorization: Bearer $USER_KEY" -H 'Content-Type: application/
        json' \
  -d '{"input":"KB 리볼빙 결제일은 10월 25일, 문의는 1588-1234 또는 ARS 상담입니다. NFT 서비스도
        있습니다."}'
```

```
{
    "normalized": "케이비 리볼빙 결제일은 시월 이십오 일, 문의는 일오팔팔 일이삼사 또는 에이알에스
        상담입니다. 엔에프티 서비스도 있습니다.",
    "sentences": [
        "케이비 리볼빙 결제일은 시월 이십오 일, 문의는 일오팔팔 일이삼사 또는 에이알에스 상담입니
        다.",
        "엔에프티 서비스도 있습니다."
    ],
    "unknown_latin": [],
    "units": [
        {"text": "케이비 리볼빙 결제일은 시월 이십오 일, 문의는 일오팔팔 일이삼사 또는 에이알에스 상
        담입니다.", "rate": 1.0, "volume": 1.0, "pause_ms": 300},
        {"text": "엔에프티 서비스도 있습니다.", "rate": 1.0, "volume": 1.0, "pause_ms": 300}
    ]
}
```

필드	뜻
`normalized`	엔진에 들어갈 전체 문장
`sentences`	문장 분할 결과(이 단위로 합성하고 캐시합니다)
`unknown_latin`	남은 영문 단어 = 사전 후보
`units`	합성 단위별 말 속도 배율, 음량 배율, 뒤 쉼(ms)

3. 사전에 넣고 다시 읽은 뒤(4.4절) 미리보기로 결과를 확인합니다. 이름 있는 사전은 `"dictionary"` 를 함께 줍니다.

```
curl -s $TTS/v1/normalize -H "Authorization: Bearer $USER_KEY" -H 'Content-Type: application/
        json' \
  -d '{"input":"리볼빙 약정 금액은 30만 원입니다.","dictionary":"card"}'
```

응답(일부): `"normalized": "일부결제금액이월약정 약정 금액은 삼십만 원입니다."`

4. 필요하면 실제로 합성해 들어 봅니다(API 개발자 매뉴얼).

작성 요령은 다음과 같습니다.

- 읽기는 소리 나는 대로 한글로 씁니다. 띄어 읽을 곳은 띄어 씁니다.
- 대문자 약어는 사전이 없어도 알파벳 이름으로 읽습니다(`ATM` → 에이티엠). 다르게 읽어야 하는 것(`LTE` → 엘티이처럼 이미 기본 사전에 든 것 외)만 넣습니다.
- 소문자·대소문자가 섞인 단어(`toss` , `Samsung`)는 규칙으로 읽지 못하므로 사전에 넣습니다.

5. 예열 목록

5.1 언제 쓰는가

- 법정 고지문처럼 **자주, 똑같이 읽는 문장**을 기동할 때 미리 합성해 캐시에 고정합니다.
- 예열한 문장은 재기동 직후에도 첫 요청부터 바로(수 ms) 나가고, 캐시 상한에 밀려나지 않습니다.
- 예열은 가장 낮은 우선순위로 돌아 서비스 요청을 방해하지 않습니다.

5.2 형식

`/opt/copland-tts/conf/prewarm.txt` 에 한 줄에 하나씩 씁니다.

```
# 예열 목록 — 한 줄에 `문장` 또는 `화자<TAB>문장`
이 상품은 원금 손실이 발생할 수 있으며, 예금자보호법에 따라 보호되지 않습니다.
M1  상담 내용은 서비스 품질 향상을 위해 녹취됩니다.
```

- `#` 으로 시작하는 줄과 빈 줄은 무시합니다.
- 화자를 적지 않은 줄은 첫 화자(`F1`)로 합성합니다. 바꾸려면 `TTS_PREWARM_VOICES` 를 씁니다(설치 매뉴얼 7.3절).
- 문장은 기본 사전과 읽기 규칙을 거쳐 문장 단위로 나뉘어 합성됩니다.

5.3 캐시에 맞는 조건

다음이 모두 같은 요청만 예열한 소리를 씁니다.

- 문장(읽기 규칙을 거친 뒤의 문장)
- 화자
- 샘플레이트: 기본 32000. 다른 값으로 요청한다면 `TTS_PREWARM_RATES` 에 그 값을 넣습니다
- 말 속도 `speed` 1.0, `seed` 를 주지 않음

음량(`volume`)과 문장 사이 쉼은 달라도 됩니다.

5.4 반영과 확인

1. 파일을 고친 뒤 `POST /v1/dictionary/reload` (4.4절)를 부릅니다. 예열을 다시 합니다.
2. `/metrics` 의 `prewarm` 으로 확인합니다(6.2절).

```
"prewarm": {"total": 1, "done": 1, "failed": 0, "running": false}
```

`total` 은 예열할 합성 단위 수(문장 × 화자 × 샘플레이트), `done` 은 끝난 수, `failed` 는 실패 수입니다. 기동 로그에도 `예열 1문장 끝 0.7s (실패 0)` 가 남습니다.

6. 모니터링

6.1 `/health` — 부하 분산기·상태 검사용

- 키가 필요 없습니다(허용 IP 검사는 받습니다).
- 정상이면 200입니다. 합성 워커 재기동 중이거나 라이선스가 만료(유예 기간도 지남)되면 503입니다. 부하 분산기와 SDK가 다른 서버로 넘깁니다.

```
curl -s -w '\n%{http_code}\n' $TTS/health
```

```
{"status":"ok","version":"0.3.4","commit":"1a2b3c4d5e","workers":6,"license":
{"license_id":"L-000123","state":"valid","expires":"2027-12-31","channels":150}}
200
```

필드	뜻
`status`	`ok` 정상(HTTP 200), `restarting` 워커 재기동 중(HTTP 503), `license_expired` 라이선스 만료(HTTP 503)
`version` , `commit`	실행 중인 버전과 빌드 커밋
`workers`	합성 워커 수
`license.license_id`	라이선스 번호
`license.state`	`valid` 유효, `grace` 만료 후 유예 중, `expired` 유예도 지남(합성 거부)
`license.expires`	만료일
`license.channels`	허가 동시 채널 수

`license.state` 가 `grace` 여도 `/health` 는 200입니다. 유예 중에 갱신하도록 `license.state` 를 따로 감시합니다.

6.2 `/metrics` — JSON

관리자(admin) 키가 필요합니다. 일반 키로 부르면 403 `admin_required` 입니다.

```
curl -s $TTS/metrics -H "Authorization: Bearer $ADMIN_KEY"
```

시험 서버(워커 2개)의 실제 출력은 다음과 같습니다.

```
{"version": "0.3.4", "build": {…}, "uptime_sec": 52.7, "workers": 2, "worker_cpus": [0, 1],
 "front_cpus": [2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15], "worker_restarts": 0, "requests": 1,
          "inflight": 0,
 "errors": 0, "rejected": 0, "timeouts": 0, "aborted": 0, "auth_failed": 0, "pool_queued": 0,
          "est_wait_ms": 0.0,
 "sentences": 2, "audio_sec": 4.9, "synth_cpu_sec": 1.3,
 "cache": {"hits": 0, "joined": 0, "misses": 3, "entries": 2, "pinned": 1, "mb": 0.7},
 "prewarm": {"total": 1, "done": 1, "failed": 0, "running": false}, "license_channels": 20,
 "license_active_channels": 1, "ttfb_ms": {"p50": 232.7, "p95": 232.7, "p99": 232.7},
 "sentence_ms": {"p50": 406.9, "p95": 406.9}}
```


카운터(«누적»)는 기동 뒤 누적 값이고 재기동하면 0이 됩니다.

필드	종류	뜻
`version`	정보	게이트웨이 버전
`build`	정보	빌드 정보(버전, 커밋, 빌드 시각 등)
`uptime_sec`	값	기동 후 경과 초
`workers`	값	합성 워커 수
`worker_cpus`	정보	합성 워커가 고정된 CPU 번호
`front_cpus`	정보	앞단이 쓰는 CPU 번호
`worker_restarts`	누적	합성 워커 재기동 횟수
`requests`	누적	받아들인 합성 요청 수(거절된 요청 제외)
`inflight`	값	지금 소리를 보내는 중인 요청 수
`errors`	누적	합성 실패 요청 수(첫 문장 실패, 보내는 중 오류)
`rejected`	누적	429로 거절한 요청 수(`overloaded` , `channel_limit` , `daily_quota` 합계)
`timeouts`	누적	문장 합성 시간 초과(504) 수
`aborted`	누적	클라이언트가 중간에 끊은 요청 수
`auth_failed`	누적	API 키 인증 실패(401) 수
`pool_queued`	값	합성 대기·실행 중인 문장 수
`est_wait_ms`	값	지금 새 요청이 들어오면 첫 문장이 워커를 잡기까지 예상 대기(ms)
`sentences`	누적	보낸 문장 수(캐시에서 나간 문장 포함, 예열 제외)
`audio_sec`	누적	보낸 음성 길이(초, 문장 사이 쉼 포함). `/v1/usage` 의 `audio_sec` 합계와 같은 기준입니다
`synth_cpu_sec`	누적	합성에 쓴 워커 시간(초, 예열 포함)
`cache.hits`	누적	캐시에서 바로 나간 문장 수
`cache.joined`	누적	같은 문장을 다른 요청이 합성 중이라 그 결과를 나눠 받은 수
`cache.misses`	누적	새로 합성한 문장 수
`cache.entries`	값	캐시에 든 문장 수(예열 제외)
`cache.pinned`	값	예열로 고정된 문장 수
`cache.mb`	값	캐시 크기(MB, 예열 포함)

필드	종류	뜻
`prewarm.total` / `done` / `failed` / `running`	값	예열 합성 단위 수 / 끝난 수 / 실패 수 / 진행 중 여부
`license_channels`	값	라이선스 허가 동시 채널 수
`license_active_channels`	값	지금 점유 중인 채널 수(8.3절)
`ttfb_ms.p50` / `p95` / `p99`	값	최근 요청 2,000건의 첫 소리 지연 백분위(ms). 요청이 없으면 `null`
`sentence_ms.p50` / `p95`	값	최근 새로 합성한 문장 5,000개의 합성 지연 백분위(ms)

워커 사용률은 `synth_cpu_sec` 의 증가량으로 계산합니다: 사용률 = Δ`synth_cpu_sec` ÷ (경과 초 × `workers`).

6.3 `/metrics?format=prometheus`

Prometheus 형식으로 같은 값을 냅니다. 이름 앞에 `copland_tts_` 가 붙습니다.

```
curl -s "$TTS/metrics?format=prometheus" -H "Authorization: Bearer $ADMIN_KEY"
```

```
# HELP copland_tts_requests_total 요청 수
# TYPE copland_tts_requests_total counter
copland_tts_requests_total 1
# HELP copland_tts_rejected_total 과부하 거절(429)
# TYPE copland_tts_rejected_total counter
copland_tts_rejected_total 0
…
# TYPE copland_tts_ttfb_ms_p95 gauge
copland_tts_ttfb_ms_p95 234.1
```

지표 이름	종류	JSON 필드
`copland_tts_requests_total`	counter	`requests`
`copland_tts_errors_total`	counter	`errors`
`copland_tts_rejected_total`	counter	`rejected`
`copland_tts_timeouts_total`	counter	`timeouts`
`copland_tts_aborted_total`	counter	`aborted`
`copland_tts_auth_failed_total`	counter	`auth_failed`
`copland_tts_sentences_total`	counter	`sentences`
`copland_tts_audio_sec_total`	counter	`audio_sec`
`copland_tts_synth_cpu_sec_total`	counter	`synth_cpu_sec`
`copland_tts_worker_restarts_total`	counter	`worker_restarts`
`copland_tts_inflight`	gauge	`inflight`
`copland_tts_pool_queued`	gauge	`pool_queued`
`copland_tts_workers`	gauge	`workers`
`copland_tts_uptime_sec`	gauge	`uptime_sec`
`copland_tts_est_wait_ms`	gauge	`est_wait_ms`
`copland_tts_license_channels`	gauge	`license_channels`
`copland_tts_license_active_channels`	gauge	`license_active_channels`
`copland_tts_cache_hits_total`	counter	`cache.hits`
`copland_tts_cache_joined_total`	counter	`cache.joined`
`copland_tts_cache_misses_total`	counter	`cache.misses`
`copland_tts_cache_entries`	gauge	`cache.entries + cache.pinned`
`copland_tts_cache_mb`	gauge	`cache.mb`
`copland_tts_ttfb_ms_p50`	gauge	`ttfb_ms.p50`
`copland_tts_ttfb_ms_p95`	gauge	`ttfb_ms.p95`
`copland_tts_ttfb_ms_p99`	gauge	`ttfb_ms.p99`

- `ttfb_ms_*` 는 요청이 한 번도 없으면 나오지 않습니다.
- `sentence_ms` , `prewarm` , 라이선스 만료일은 Prometheus 형식에 없습니다. JSON(`/metrics`), `/health` , `/v1/license` 로 봅니다.

Prometheus 수집 설정 예는 다음과 같습니다. **관리자(admin) 키**를 씁니다. 수집 설정 파일의 권한을 좁혀 키를 보호합니다.

```
scrape_configs:
  - job_name: copland-tts
    metrics_path: /metrics
    params:
      format: [prometheus]
    authorization:
      credentials: ctts_yyyyyyyy...
    static_configs:
      - targets: ["10.0.0.21:9300", "10.0.0.22:9300"]
```

6.4 라이선스 상태 — `/v1/license`

```
curl -s $TTS/v1/license -H "Authorization: Bearer $ADMIN_KEY"
```


```
{"license_id":"L-000123","product":"tts","product_name":"COPLAND TTS","entitlements":{"channels":
          150},"expires":"2027-12-31","grace_days":14,"state":"valid","days_left":445,"machine_key":"KYS1-
          W8VT","customer":"OO은행","active_channels":12,"inflight":3}
```


`days_left` 는 만료일까지 남은 날 수입니다. 만료 뒤에는 음수가 됩니다.

6.5 경보 기준 예

감시 항목	경보 조건 예	의미와 조치
`/health` HTTP 상태	200이 아님이 1분 이상	워커 재기동이 끝나지 않거나 서버 정지. 로그 확인
`rejected` 증가	5분 동안 계속 증가	처리 능력 또는 라이선스 채널 부족. 아래 두 행으로 원인 구분
`license_active_channels` ÷ `license_channels`	0.9 이상	라이선스 채널이 거의 다 참 (`channel_limit`). 채널 증설 검토
워커 사용률	80% 이상이 지속	처리 능력이 거의 다 참(`overloaded`). 코어·서버 증설 검토
`ttfb_ms.p95`	1,000ms 초과가 지속	첫 소리가 늦음. 과부하나 CPU 고정 확인
`worker_restarts` 증가	1 이상 증가	합성 워커 비정상 종료. 메모리 부족(OOM) 확인
`timeouts` 증가	1 이상 증가	문장 합성이 30초를 넘음. 서버 상태 확인
`errors` 증가	지속 증가	로그의 `합성 중 오류` 확인
`auth_failed` 급증	평소보다 크게 증가	키 설정 오류나 무단 접근 시도. 오류 로그의 `ip` 확인
라이선스 `days_left`	30일 이하	갱신 키 발급 준비
`license.state`	`valid` 가 아님	`grace` 면 갱신 키를 바로 넣음. `expired` 면 합성이 거부되는 중

7. 사용량 정산

7.1 조회(`/v1/usage`)

API 키 이름별 누적 사용량입니다. 부서별로 키를 나눠 발급하면 부서별로 정산할 수 있습니다.

```
curl -s $TTS/v1/usage -H "Authorization: Bearer $ADMIN_KEY"
```

```
{"since":"2026-10-12T23:40:21","keys":{"web01":{"requests":1,"chars":26,"audio_sec":4.9}}}
```

필드	뜻
`since`	집계 시작 시각(기동 또는 마지막 초기화, 한국 시각)
`keys.<이름>.requests`	받아들인 합성 요청 수
`keys.<이름>.chars`	입력 글자 수(SSML 태그 포함)
`keys.<이름>.audio_sec`	보낸 음성 길이(초, 문장 사이 쉼 포함)

7.2 초기화(`/v1/usage/reset` , admin)

```
curl -s -X POST $TTS/v1/usage/reset -H "Authorization: Bearer $ADMIN_KEY"
```

- 초기화 **직전의 값**을 7.1절과 같은 형식으로 돌려주고 0으로 되돌립니다.
- 사용량은 메모리에만 있습니다. **재기동하면 0이 됩니다.** 정산에 쓰려면 정기적으로 초기화하며 응답을 저장합니다.

매일 자정에 저장하는 cron 예는 다음과 같습니다(`/etc/cron.d/copland-tts-usage`).

```
0 0 * * * root curl -s -X POST http://127.0.0.1:9300/v1/usage/reset -H 'Authorization: Bearer
ctts_yyyyyyyy...' >> /var/log/copland-tts-usage.jsonl && echo >> /var/log/copland-tts-usage.jsonl
```

이 파일에는 관리자 키가 들어가므로 권한을 `600` 으로 둡니다. 이중화 구성이면 서버마다 따로 모아 합칩니다.

8. 용량·성능

8.1 채널 산정

필요한 물리 코어는 설치 매뉴얼 2.2절의 산정표를 따릅니다. 요약은 다음과 같습니다.

- 시험 서버(Intel Xeon w5-2445, 물리 10코어, AMX)에서 150채널 동시 발화:
  - 물리 4코어: 끊김 0회, 첫 소리 p95 368ms
  - 물리 6코어: 끊김 0회, 첫 소리 p95 247ms
- **150채널 권장: AMX 물리 6코어 + 앞단 1코어.**
- AMX 없는 서버는 물리 코어가 약 3배 필요합니다.
- 운영 중에는 6.5절의 워커 사용률과 `ttfb_ms.p95` 로 여유를 봅니다.

8.2 과부하 때의 동작

게이트웨이는 받을 수 없는 요청을 바로 429로 돌려보냅니다. 이미 재생 중인 요청이 끊기지 않게 하기 위해서입니다.

상황	응답	SDK 동작
새 요청의 첫 문장 예상 대기가 1초 (`TTS_MAX_WAIT`)를 넘음	429 `overloaded` , `Retry-After: 1`	다음 서버로 넘김
재생 중인 요청의 다음 문장이 늦어지기 시작함	429 `overloaded` , `Retry-After: 1`	다음 서버로 넘김
라이선스 동시 채널을 다 씀	429 `channel_limit` , `Retry-After: 1`	다음 서버로 넘김
합성 워커 재기동 중	503 `restarting` , `Retry-After: 5`	다음 서버로 넘김

- 첫 문장이 캐시에 있으면 과부하여도 내보냅니다.
- 문장은 «마감 시각이 이른 것부터» 합성합니다. 새 요청의 첫 문장은 지금이 마감이고, 재생 중인 요청의 다음 문장은 클라이언트가 받은 소리를 다 재생하기 1초 전이 마감입니다. 그래서 첫 소리는 빠르고 재생 중에는 끊기지 않습니다.
- 서버가 1대뿐이면 SDK가 넘길 곳이 없어 웹 서버에 오류가 돌아갑니다. 이중화를 권장합니다.

8.3 동시 채널 라이선스의 의미

- 라이선스의 «동시 채널»은 **동시에 재생되고 있는 합성 요청 수**입니다.
- 요청 하나는 **클라이언트가 받은 소리를 다 재생할 때까지** 채널 하나를 차지합니다. 합성은 실시간보다 훨씬 빨리 끝나지만, 채널은 재생 시간만큼 점유됩니다.
  - 예: 30초짜리 안내문은 합성이 1초 만에 끝나도 약 30초 동안 채널 1개를 씁니다.
  - `stream: false` 로 한 번에 받으면 다 받은 시각부터 재생 시간만큼 점유합니다.

- 클라이언트가 중간에 끊거나 오류가 나면 채널을 바로 돌려줍니다.
- 지금 점유 중인 채널은 `/metrics` 의 `license_active_channels` , `/v1/license` 의 `active_channels` 로 봅니다.
- 채널이 다 차면 429 `channel_limit` 입니다. 처리 능력이 남아 있어도 거절됩니다.

9. 백업

대상	경로	이유
라이선스 키	`/opt/copland-tts/KEY`	재발급하려면 포털 절차가 필요합니다
설정	`/opt/copland-tts/.env`	CPU 고정, 워커 수, 허용 IP 등
API 키	`/opt/copland-tts/conf/keys.tsv`	잃으면 모든 키를 다시 발급해야 합니다
사전·예열	`/opt/copland-tts/conf/customer.tsv`, `conf/dicts/`, `conf/prewarm.txt`	고객사가 만든 자산입니다
실행 구성	`/opt/copland-tts/docker-compose.yml`	고친 경우만

모델과 이미지는 패키지에서 다시 설치할 수 있으므로 백업하지 않아도 됩니다.

```
sudo tar czf /backup/copland-tts-$(date +%F).tar.gz -C /opt/copland-tts KEY .env docker-
        compose.yml conf
```

설정을 바꿀 때마다, 그리고 정기적으로 백업합니다. 복원은 같은 장비에 설치한 뒤 이 파일들을 제자리에 풀고 `sudo docker compose up -d` 로 합니다. KEY는 그 장비에서만 쓸 수 있습니다.

10. 보안 점검표

점검 항목	확인 방법
9300 포트는 웹 서버·부하 분산기·모니터링 주소에만 열려 있다	호스트 방화벽 규칙 확인
`TTS_ALLOW` 에 필요한 주소만 있다	`.env` 확인, 기동 로그의 `허용 IP`
게이트웨이는 내부망에만 두었다. 외부로 낼 때는 TLS 프록시를 거친다	네트워크 구성 확인
프록시 뒤라면 `TTS_TRUST_PROXY` 는 프록시 주소만이고, 프록시가 `X-Real-IP` 를 넣는다	`.env` , 프록시 설정 확인
키 파일에는 `sha256:` 값만 있다(평문 키 없음)	`grep -v '^#' /opt/copland-tts/conf/keys.tsv` 의 둘째 칸
관리자 키는 운영자만 가지고 있고, 웹 서버에는 일반 키만 주었다	키 파일의 역할 칸
웹 서버·부서마다 키를 따로 발급했다(폐기·정산 단위)	키 파일의 이름 칸
쓰지 않는 키, 퇴사자·종료된 시스템의 키를 지웠다	키 파일 정기 점검
키 원문(`ctts_...`)을 문서·소스 저장소·로그에 남기지 않았다	웹 서버 설정은 환경 변수나 보안 저장소 사용
관리자 키가 든 cron 파일 등은 권한이 `600` 이다	`ls -l /etc/cron.d/copland-tts-usage`
설치 폴더는 root 소유이고 일반 사용자가 쓸 수 없다	`ls -la /opt/copland-tts`
`KEY` , `conf/` , `.env` 를 백업했다	9장
라이선스 만료 경보를 걸었다	6.5절
관리자 키를 Prometheus 등 수집기에 넣었다면 그 설정 파일의 권한을 좁혔다	수집 설정 파일 권한 확인
`/health` , `/openapi.json` 은 키 없이 열린다(허용 IP 검사는 받음)	운영 정책 확인

참고로 게이트웨이는 다음을 기본으로 지킵니다.

- 입력 문장을 로그에 남기지 않습니다.
- 컨테이너는 일반 사용자(UID 1000)로 실행되고 설치 폴더를 읽기 전용으로 마운트합니다.
- 외부(인터넷)로 연결하지 않습니다.

문의: COPLAND 기술지원 (연락처는 계약서 참조)