COPLAND

기술 문서

COPLAND TTS

설치 매뉴얼

서버 사양 산정부터 장비키 확인, install.sh 설치, 동작 확인까지의 절차를 설명합니다.

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

차례

1. 제품 개요
   1.1 구성
   1.2 패키지 구성
2. 서버 요구 사항
   2.1 운영체제와 소프트웨어
   2.2 CPU
   2.3 메모리
   2.4 디스크
   2.5 포트와 방화벽
   2.6 `network_mode: host` 가 필요한 이유
3. 설치 전 준비 — 장비키 확인
   3.1 장비키 확인 방법
   3.2 QR 표시(`--qr`)
   3.3 장비키가 바뀌는 경우
4. 라이선스 받기
   4.1 발급 흐름
   4.2 KEY 파일 형식
5. 설치 절차
   5.1 반입과 무결성 확인
   5.2 설치 프로그램 실행
   5.3 첫 API 키 발급(4/5 단계)
   5.4 라이선스 키 입력(5/5 단계)
   5.5 기동과 확인
6. 설치 후 폴더 구조
7. 설정
   7.1 `.env`
   7.2 `docker-compose.yml` 의 `env_file` 과 `environment`
   7.3 전체 옵션
   7.4 CPU 고정(cpuset) 정하는 법
   7.5 설정 반영
8. 이중화
9. 업그레이드
   9.1 새 버전 설치
   9.2 되돌리기

10. 라이선스 갱신·재발급
    10.1 새 라이선스 키 넣기
    10.2 만료와 유예 기간
    10.3 장비 교체
11. 버전 확인
12. 문제 해결
    12.1 설치·기동
    12.2 라이선스
    12.3 운영 중 API 오류
13. 제거

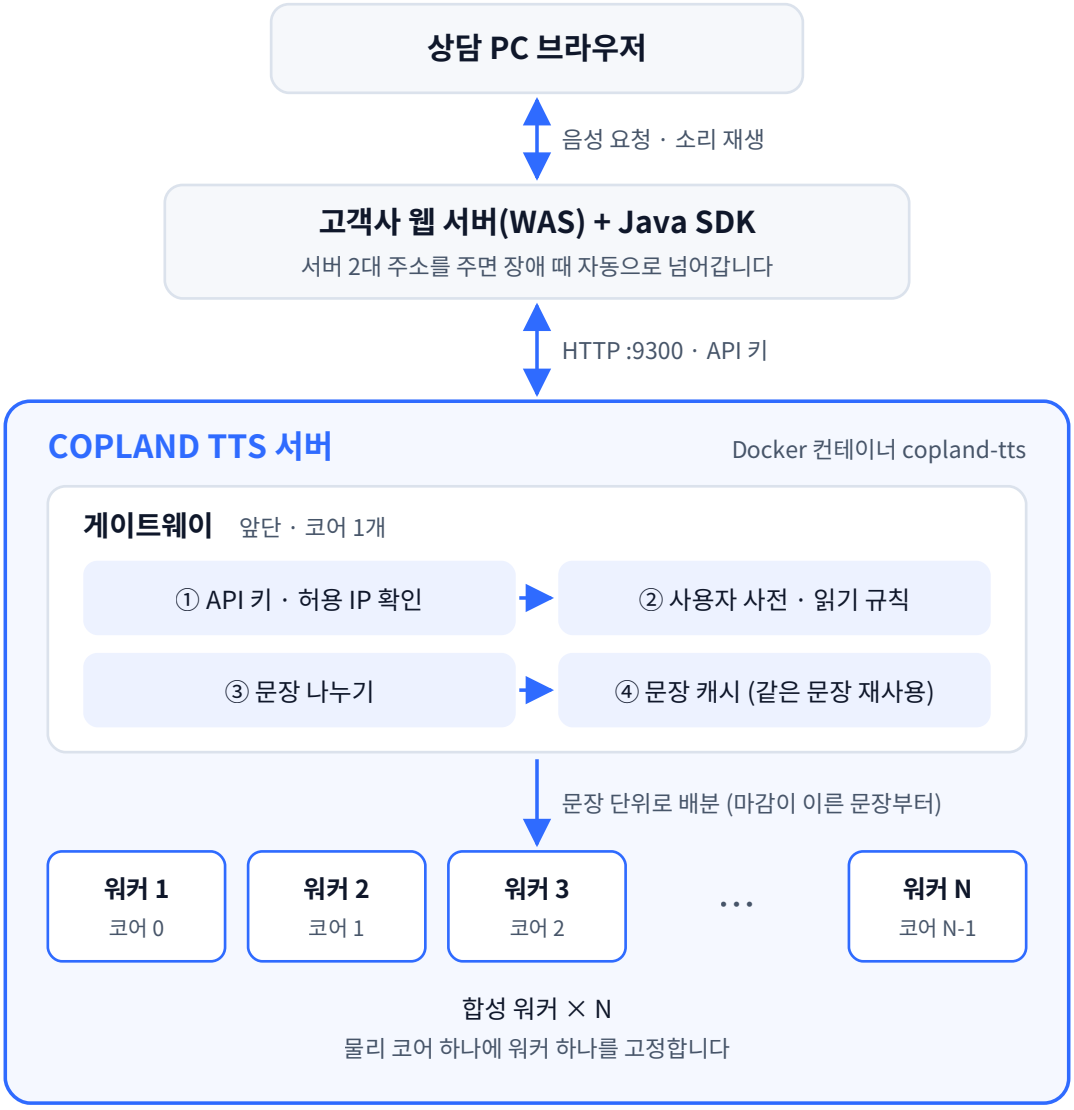
- 적용 버전: **COPLAND TTS 0.3.4**(게이트웨이), Java SDK 0.2.1
- 문서 갱신일: 2026-10-12
- 대상 독자: 고객사 서버 설치 담당자
- 관련 문서: 운영 매뉴얼, API 개발자 매뉴얼, Java SDK 매뉴얼, 라이선스 발급 포털 사용 안내

이 문서만 보고 서버 준비부터 설치, 설정, 업그레이드, 제거까지 끝낼 수 있게 썼습니다. 명령은 그대로 복사해 실행할 수 있습니다. 각 절차 뒤에 확인 방법과 예상 출력을 붙였습니다.

1. 제품 개요

COPLAND TTS는 고객사 서버에 설치하여 운영하는 한국어 음성 합성 서버 제품입니다. 상담 화면이 읽어 줄 문장을 웹 서버로 보내면, 웹 서버가 게이트웨이를 호출하여 음성을 받아 브라우저로 전달합니다.

1.1 구성



COPLAND TTS 구성

- 게이트웨이는 첫 문장이 합성되는 즉시 소리를 보내기 시작합니다. 긴 안내문도 첫 소리까지 문장 1개 합성 시간만 걸립니다.
- 같은 문장은 늘 같은 소리로 합성되므로 캐시에서 바로 나갑니다.
- 입력 문장은 로그에 남지 않습니다. 접근 로그에는 글자 수, 문장 수, 지연만 남습니다.

- 라이선스는 장비에 고정됩니다. 인터넷 연결이 없어도 동작합니다(폐쇄망 설치 가능).

1.2 패키지 구성

납품물은 `copland-tts-0.3.4.tar.gz` (약 430MB) 하나와 그 체크섬 파일 `copland-tts-0.3.4.tar.gz.sha256` 입니다. 압축을 풀면 다음 파일이 나옵니다.

경로	내용
`install.sh`	설치 프로그램. 하위 명령 `machine-key` · `key` · `status` · `version`
`copland-machine-key.sh`	장비키 확인 프로그램. 설치 전 단독 실행 가능
`qr.awk`	장비키 QR 표시용 보조 파일(`--qr`)
`docker-compose.yml`	실행 구성(`network_mode: host` , CPU 고정, 설치 경로 읽기 전용 마운트)
`images/copland-tts-gw-0.3.4.tar.gz`	게이트웨이 Docker 이미지
`models/`	음성 모델(라이선스 사본 포함)
`conf/keys.tsv.example`	API 키 파일 견본
`conf/customer.tsv.example`	고객사 사전 견본
`conf/prewarm.txt.example`	예열 목록 견본
`sdk/copland-tts-client-0.2.1.jar`	Java SDK(소스 jar `-sources.jar` 포함)
`sdk/README.md`	Java SDK 사용 안내
`NOTICE.md`	서드파티 라이선스 고지
`CHANGELOG.md`	변경 내역
`VERSION`	패키지 버전(`0.3.4`)
`BUILD_INFO.json`	빌드 정보(버전, 커밋, 빌드 시각)
`docs/`	고객용 매뉴얼(Markdown, PDF). 이 문서도 들어 있습니다
`SHA256SUMS`	패키지 안 모든 파일의 체크섬

서드파티 소프트웨어와 모델의 라이선스 조건은 패키지의 `NOTICE.md` 를 참조합니다.

2. 서버 요구 사항

2.1 운영체제와 소프트웨어

항목	요구 사항
운영체제	Linux x86_64
컨테이너	Docker Engine과 compose 플러그인(`docker compose`). 플러그인이 없으면 `docker-compose` 를 씁니다
기타 명령	`bash` , `curl` , `gunzip` , `sha256sum` , `awk`
권한	설치는 root(`sudo`)로 실행합니다. 이 문서의 `docker` 명령은 root 또는 `docker` 그룹 사용자로 실행합니다
인터넷	필요 없습니다. 라이선스는 오프라인으로 확인합니다

확인 방법은 다음과 같습니다.

```
uname -m                # x86_64
docker --version
docker compose version
```

2.2 CPU

- **AMX가 있는 Intel Xeon(4세대 이상)을 권장합니다.** AMX 가속(BF16)으로 합성합니다.
- AMX가 없는 서버(AMD EPYC, 이전 세대 Xeon 등)도 동작합니다. 이때는 기본 실행(FP32)으로 바꾸고, **물리 코어가 약 3배 필요합니다.**
- 하이퍼스레딩은 처리량을 늘리지 못합니다. **코어 수는 물리 코어로 셉니다.**

AMX 확인 방법은 다음과 같습니다.

```
grep -o -m1 amx_bf16 /proc/cpuinfo
```

`amx_bf16` 이 출력되면 AMX가 있습니다. 아무것도 출력되지 않으면 AMX가 없습니다.

물리 코어 산정표

필요한 코어는 «합성 워커 수(물리 코어) + 앞단 1코어»입니다.

동시 채널	AMX 있음: 합성 워커	AMX 있음: 필요 물리 코어	AMX 없음: 합성 워커	AMX 없음: 필요 물리 코어
50	2	3	6	7
100	4	5	12	13
150	**6** (최소 4)	**7** (최소 5)	18	19
200	8	9	24	25
300	12	13	36	37

- 근거는 시험 서버(Intel Xeon w5-2445, 물리 10코어, AMX)의 실측입니다. 150채널이 동시에 쉬지 않고 말하는 조건(32kHz)에서 측정했습니다.
  - 물리 4코어: 끊김 0회, 첫 소리 p95 368ms, 워커 사용률 87%
  - 물리 6코어: 끊김 0회, 첫 소리 p95 247ms, 워커 사용률 61%
- **150채널 권장 사양은 AMX 물리 6코어 + 1코어입니다.** 4코어로도 끊김 없이 동작하지만 여유가 적습니다.
- 150채널 외의 행은 권장값(워커 1개당 25채널)으로 비례 산정한 값입니다. 설치 뒤 실제 부하로 확인하기를 권합니다.
- 2소켓 서버는 워커를 한 소켓 안의 코어로 잡는 것이 좋습니다(7.4절).

2.3 메모리

- 합성 워커 1개가 약 1GB를 씁니다(시험 실측 약 0.9GB).
- 앞단과 문장 캐시(기본 상한 512MB)에 약 1GB가 더 듭니다.
- 워커 6개 구성은 약 7GB를 씁니다. **16GB 이상을 권장합니다.**

2.4 디스크

항목	크기
패키지 압축 파일	약 430MB
패키지를 푼 폴더	약 430MB
Docker 이미지(적재 후)	약 770MB
설치 경로의 모델(`/opt/copland-tts/models`)	약 260MB
컨테이너 로그	최대 250MB(50MB × 5개 순환)

여유 공간 3GB 이상을 권장합니다. 업그레이드하면 이전 버전 이미지가 남으므로 버전마다 약 770MB가 늘어납니다.

2.5 포트와 방화벽

방향	포트	상대	용도
들어옴	TCP 9300(`TTS_PORT`)	웹 서버, 부하 분산기, 모니터링 서버	합성 API, `/health` , `/metrics`
나감	없음	-	외부 연결이 필요 없습니다

- 게이트웨이는 HTTP(평문)로 응답합니다. 내부망에서만 열거나, 앞에 TLS를 처리하는 프록시를 둡니다.
- 컨테이너가 호스트 네트워크를 쓰므로 **호스트 방화벽이 그대로 적용됩니다.** 웹 서버 주소에만 9300을 엽니다.

firewalld 예시(웹 서버 10.0.0.11, 10.0.0.12)는 다음과 같습니다.

```
sudo firewall-cmd --permanent --add-rich-rule='rule family="ipv4" source address="10.0.0.11/32"
          port port="9300" protocol="tcp" accept'
sudo firewall-cmd --permanent --add-rich-rule='rule family="ipv4" source address="10.0.0.12/32"
          port port="9300" protocol="tcp" accept'
sudo firewall-cmd --reload
```

2.6 `network_mode: host` 가 필요한 이유

- 라이선스는 장비의 CPU와 **물리 네트워크 카드(NIC)** 를 확인합니다.
- Docker 기본 네트워크(bridge)로 띄우면 컨테이너 안에는 가상 NIC만 보입니다. 그러면 라이선스를 확인하지 못해 기동하지 않습니다.
- 그래서 `docker-compose.yml` 은 `network_mode: host` 로 되어 있습니다. 이 설정을 바꾸지 마십시오.
- 호스트 네트워크에서는 포트 매핑(`ports:`)을 쓰지 않습니다. 게이트웨이가 호스트의 `TTS_PORT` (기본 9300)에서 직접 받습니다.

3. 설치 전 준비 — 장비키 확인

라이선스는 장비마다 발급합니다. 설치 전에 이 장비의 **장비키(8자)** 를 확인해 발급 포털에 입력합니다.

3.1 장비키 확인 방법

패키지에 든 `copland-machine-key.sh` 는 단독으로 실행할 수 있습니다. bash와 기본 명령(`sha256sum` , `awk`)만 쓰고 root 권한이 필요 없습니다. 패키지를 반입하기 전에 이 파일만 먼저 보내 장비키를 받아도 됩니다.

1. 실제 장비(호스트)에서 실행합니다. 컨테이너 안에서는 물리 NIC가 보이지 않습니다.

```
bash copland-machine-key.sh
```

2. 화면의 «장비키»를 옮겨 적습니다.

예상 출력(장비 이름과 CPU는 예시입니다)은 다음과 같습니다.

```
COPLAND 장비키 확인
──────────────────────────────────────────────────────────────────────────────
장비 이름   : tts-svr01
CPU         : Intel(R) Xeon(R) Gold 6430 (소켓 1, 물리 코어 32)
물리 NIC    : ens1f0 ens1f1

장비키      :  KYS1-W8VT

1) COPLAND 발급 포털 https://copland.ai/license/ 에 거래처 계정으로 로그인
2) 제품을 고르고 위 장비키 8자를 입력 (가운데 '-' 는 없어도 됩니다)
3) 받은 라이선스 키를 설치 경로의 KEY 파일로 넣기 (예: /opt/copland-tts/KEY)
※ CPU 를 바꾸거나 이 네트워크 카드를 빼면 장비키가 달라져 다시 발급받아야 합니다.
```

- 장비키는 `0` ~ `9` , `A` ~ `Z` (I·L·O·U 제외)로 된 8자입니다. 가운데 `-` 는 읽기 편하게 넣은 것입니다.
- 장비키에는 확인 문자가 들어 있습니다. 옮겨 적다 틀린 글자는 포털이 대부분(255/256) 잡아냅니다.
- MAC 주소 원문은 장비키에 들어가지 않습니다.
- 장비키만 8자로 출력하려면 `-q` 를 붙입니다(스크립트용). 출력은 `KYS1W8VT` 처럼 `-` 없이 나옵니다.
- 설치한 뒤에는 `sudo bash /opt/copland-tts/install.sh machine-key` 로도 같은 값을 볼 수 있습니다.

3.2 QR 표시(`--qr`)

```
bash copland-machine-key.sh --qr
```

- 장비키 아래에 QR을 함께 표시합니다. 폰으로 촬영하면 발급 포털의 발급 화면이 열립니다.
- 같은 폴더에 `qr.awk` 가 있어야 합니다.
- **화면 촬영이 허용된 곳에서만 씁니다.** 촬영이 금지된 전산실에서는 장비키 8자를 손으로 옮겨 적는 것이 기본입니다.

3.3 장비키가 바뀌는 경우

장비키는 다음 정보로 만듭니다.

- CPU: 제조사, 모델, 소켓 수, 물리 코어 수
- 물리 NIC: 장비의 물리 네트워크 카드 MAC

다음 경우에 장비키(또는 라이선스 판정)가 바뀌고, 라이선스를 다시 받아야 합니다.

- **CPU를 교체**하거나 소켓·코어 구성을 바꾼 경우
- 라이선스를 받을 때 **등록된 NIC를 제거**한 경우

NIC를 **추가**하는 것은 상관없습니다. 등록된 NIC가 장비에 남아 있으면 같은 장비로 봅니다. BIOS·마이크로코드 업데이트로는 바뀌지 않습니다.

4. 라이선스 받기

4.1 발급 흐름

발급 포털 주소는 https://copland.ai/license/ 입니다. 거래처 계정은 COPLAND가 미리 만들어 드립니다.

1. 발급 포털에 거래처 계정으로 로그인합니다.
2. 제품 **COPLAND TTS**를 고릅니다.
3. 3장에서 확인한 장비키 8자를 입력합니다.
4. 장비 이름과 동시 채널 수를 정하고 계약 잔여량을 확인합니다.
5. 발급합니다. 화면에 라이선스 키가 표시되고 `KEY` 파일로 내려받을 수 있습니다.

계약 방식에 따라 즉시 발급되거나 COPLAND 승인 뒤 발급됩니다. 화면별 상세 절차는 라이선스 발급 포털 사용 안내를 참조합니다.

4.2 KEY 파일 형식

```
# COPLAND 라이선스 키 — 설치 경로의 KEY 파일로 넣는다 (# 줄은 설명, 나머지 줄이 키)
# 제품: COPLAND TTS
# 라이선스: L-000123
# 동시 채널: 150 채널
# 만료: 2027-12-31 (+유예 14일)
# 장비키: KYS1-W8VT
XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX Q
XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX Y
XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX K
XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX P
```

- `#` 로 시작하는 줄은 사람이 읽는 설명입니다. 서명 대상이 아니며 설명이 더 붙을 수 있습니다.
- 나머지 줄이 키입니다. 한 줄은 **4자 × 8묶음 + 확인 문자 1자**입니다. TTS 라이선스는 4줄입니다.
- 줄 끝의 확인 문자로 어느 줄을 잘못 옮겼는지 바로 알 수 있습니다.
- 입력할 때 대소문자, `-`, 공백은 구분하지 않습니다. `O`는 `0`으로, `I`·`L`은 `1`로 읽습니다.
- 라이선스 키에는 제품, 라이선스 번호, 장비, 동시 채널 수, 만료일, 유예 일수가 서명되어 들어 있습니다. 한 글자라도 바꾸면 쓸 수 없습니다.

5. 설치 절차

5.1 반입과 무결성 확인

1. 패키지 파일 2개를 서버의 작업 폴더(예: `/root`)로 옮깁니다.
2. 압축 파일의 체크섬을 확인합니다.

```
sha256sum -c copland-tts-0.3.4.tar.gz.sha256
```

   예상 출력: `copland-tts-0.3.4.tar.gz: OK` (한국어 환경에서는 `성공`)

3. 압축을 풉니다.

```
tar xzf copland-tts-0.3.4.tar.gz
cd copland-tts-0.3.4
```

4. 패키지 안 파일의 체크섬을 확인합니다.

```
sha256sum -c --quiet SHA256SUMS && echo 이상 없음
```

   예상 출력: `이상 없음`. 틀린 파일이 있으면 그 파일 이름과 `FAILED`가 나옵니다. 이때는 패키지를 다시 받습니다.

> 설치 프로그램은 `install.sh`, `VERSION`, `BUILD_INFO.json`, `docs/`를 설치 경로(`/opt/copland-tts`)에도 복사합니다. 설치가 끝나면 패키지 폴더를 지워도 됩니다. 이후 `status` · `version` · `key` 명령은 `sudo bash /opt/copland-tts/install.sh <명령>`으로 실행합니다.

5.2 설치 프로그램 실행

```
sudo bash install.sh
```

- 설치 경로는 기본 `/opt/copland-tts` 입니다. 다른 경로에 설치하려면 `--prefix`를 줍니다.

```
sudo bash install.sh --prefix /data/copland-tts
```

`--prefix` 를 쓰면 이후 `key` · `status` 명령은 그 경로의 `install.sh` 로 실행하고 같은 `--prefix` 를 줍니다. 이 문서의 `/opt/copland-tts` 도 그 경로로 바꿔 읽습니다.

- 도움말은 `bash install.sh --help` 로 봅니다.

`install.sh` 는 다음 5단계를 진행합니다. 화면에 `▶ 1/5` 처럼 단계가 표시됩니다.

단계	화면 표시	하는 일
사전 검사	(문제가 있을 때만)	root·docker 확인, AMX 확인
1/5	`▶ 1/5 장비키`	장비키 표시(3장과 같은 화면)
2/5	`▶ 2/5 이미지 적재 (copland-tts-gw:0.3.4)`	Docker 이미지 적재(이미 있으면 건너뜀)
3/5	`▶ 3/5 파일 복사 → /opt/copland-tts`	구성 파일·모델·견본 복사, `.env` 작성
4/5	`▶ 4/5 API 키`	첫 API 키 2개 발급(키 파일이 비어 있을 때만)
5/5	`▶ 5/5 라이선스`	라이선스 키 입력과 확인
기동	`▶ 기동`	컨테이너 기동, `/health` 확인

첫 줄에는 버전이 표시됩니다.

```
COPLAND TTS 0.3.4 (v0.3.4, 2026-10-12T15:00:00Z)
```

AMX가 없는 CPU면 첫 줄 아래에 다음 경고가 나옵니다. 설치는 계속됩니다. 설치가 끝난 뒤 7.1절대로 `TTS_BACKEND=ov-f32` 를 넣습니다.

```
  ⚠ 이 CPU 는 AMX 가 없습니다 → 설치 뒤 /opt/copland-tts/.env 에 TTS_BACKEND=ov-f32 (코어가 약 3배 필요)
```

5.3 첫 API 키 발급(4/5 단계)

키 파일(`conf/keys.tsv`)에 키가 하나도 없으면 키 2개를 만듭니다.

- `web01` : 웹 서버용 일반(user) 키
- `ops` : 운영자용 관리자(admin) 키. 사전·키 다시 읽기, 지표(`/metrics`), 사용량, 라이선스 조회에 씁니다

```
▶ 4/5 API 키
  웹 서버용(user)   — 클라이언트에 줄 키 (다시 볼 수 없음): ctts_xxxxxxxx...
  운영자용(admin)   — 클라이언트에 줄 키 (다시 볼 수 없음): ctts_yyyyyyyy...
  ↑ 두 키는 다시 볼 수 없습니다. user 키는 웹 서버 담당자에게, admin 키(사전·키 다시 읽기·지표·사용량)
는 운영자가 보관하세요.
    추가 발급은 운영 매뉴얼 «API 키 관리» 참고
```

- **`ctts_` 로 시작하는 두 키는 이때 한 번만 보입니다.** 바로 안전한 곳에 기록합니다.
  - user 키는 웹 서버 담당자에게 전달합니다.
  - admin 키는 운영자만 보관합니다. 웹 서버에 주지 않습니다.
- 서버의 키 파일에는 키의 sha256 값만 저장됩니다. 키 원문은 서버에 남지 않습니다.
- 키를 잃어버리면 새 키를 발급합니다(운영 매뉴얼 «API 키 관리»).
- 다시 설치할 때 키 파일에 키가 있으면 `기존 키 파일 유지: /opt/copland-tts/conf/keys.tsv` 가 나오고 새로 만들지 않습니다.

5.4 라이선스 키 입력(5/5 단계)

설치 경로에 이 장비에 맞는 `KEY` 가 이미 있으면 `기존 KEY 가 이 장비에 맞습니다.` 가 나오고 넘어갑니다. 없으면 다음 화면이 나옵니다.

```
▶ 5/5 라이선스

  라이선스 키 넣기 — 발급 포털에서 받은 키
   1) KEY 파일 경로 입력 (USB 등으로 옮겨 온 경우)
   2) 키를 한 줄씩 직접 입력 (줄 끝 한 글자가 확인 문자 — 틀리면 그 줄만 다시)
  선택 [1/2]:
```

방법 1: KEY 파일로 넣기

발급 포털에서 내려받은 `KEY` 파일을 USB 등으로 서버에 옮겨 둔 경우입니다.

1. `1` 을 입력합니다.
2. 파일 경로를 입력합니다.

```
  선택 [1/2]: 1
  KEY 파일 경로: /root/KEY
```

파일이 없으면 `오류: 파일이 없습니다: /root/KEY` 가 나오고 설치가 멈춥니다. 경로를 확인한 뒤 `sudo bash install.sh` 를 다시 실행합니다. 이미 끝난 단계(이미지 적재, API 키 발급)는 건너뜁니다.

방법 2: 한 줄씩 직접 입력

폰 촬영이나 파일 반입이 막힌 곳에서 화면의 키를 손으로 옮겨 입력하는 경우입니다.

1. `2` 를 입력합니다.
2. 키 줄을 1번째 줄부터 하나씩 입력합니다. `#` 설명 줄은 입력하지 않습니다.
3. 줄마다 바로 확인합니다. 맞으면 `✓` 가 나오고 다음 줄로 넘어갑니다.
4. 틀린 글자가 있으면 **그 줄만** 다시 입력합니다.
5. 4줄을 다 넣었으면 빈 줄에서 Enter를 누릅니다.

```
선택 [1/2]: 2
1번째 줄 (끝났으면 그냥 Enter): XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX Q
  ✓
2번째 줄 (끝났으면 그냥 Enter): XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXY Y
  ✗ 2번째 줄에 틀린 글자가 있습니다. 다시 입력하세요.
2번째 줄 (끝났으면 그냥 Enter): XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX Y
  ✓
3번째 줄 (끝났으면 그냥 Enter): XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX K
  ✓
4번째 줄 (끝났으면 그냥 Enter): XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX P
  ✓
5번째 줄 (끝났으면 그냥 Enter):
```

- 글자 수가 맞지 않으면 `✗ 한 줄은 4자 × 8묶음 + 확인 1자 (33자) 입니다. 다시 입력하세요.` 가 나옵니다.
- 소문자로 입력해도 되고, `-` 와 공백은 빼도 됩니다.

라이선스 확인 결과

입력이 끝나면 라이선스를 확인하고 내용을 보여 줍니다.

```
▶ 라이선스 확인
{
 "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"
}
```

- `entitlements.channels` 가 계약한 동시 채널 수인지, `machine_key` 가 이 장비의 장비키인지 확인합니다.
- 장비나 제품이 맞지 않으면 오류 문구 뒤에 다음 안내가 나오고 다시 입력을 받습니다.

```
오류: 이 장비용 라이선스가 아님 (CPU 가 다름)
  → 이 장비의 장비키와 발급 포털에 넣은 장비키가 같은지, 제품이 TTS 인지 확인하세요.
  다시 넣으세요 (Ctrl+C 로 중단 — 나중에 'bash install.sh key')
```

- 키가 아직 없으면 Ctrl+C로 중단합니다. 키를 받은 뒤 `sudo bash /opt/copland-tts/install.sh key` 로 넣습니다(10.1 절).

5.5 기동과 확인

라이선스가 확인되면 컨테이너를 띄우고 최대 3분 동안 `/health` 를 기다립니다. 합성 워커가 모델을 올리는 데 수 초가 걸립니다.

```
▶ 기동
(docker compose 기동 메시지)
{"status":"ok","version":"0.3.4","commit":"1a2b3c4d5e","workers":6,"license":
{"license_id":"L-000123","state":"valid","expires":"2027-12-31","channels":150}}

  설치 끝. 설치 경로 /opt/copland-tts · 로그: docker logs -f copland-tts · 상태: bash /root/copland-
tts-0.3.4/install.sh status
```

마지막 줄의 상태 명령은 패키지 폴더 경로로 나옵니다. 설치 경로의 `sudo bash /opt/copland-tts/install.sh status` 도 같습니다.

설치 뒤 확인 방법은 다음과 같습니다.

1. 상태를 확인합니다.

```
curl -s http://127.0.0.1:9300/health
```

`"status":"ok"` 와 `"state":"valid"` 가 나오면 정상입니다.

2. 기동 로그를 확인합니다.

```
docker logs copland-tts 2>&1 | head
```

예상 출력(예시)은 다음과 같습니다.

```
2026-10-12 23:18:18,596 INFO gateway: 라이선스 L-000123 — 동시 150채널, 만료 2027-12-31
(valid), 고객 OO은행
2026-10-12 23:18:21,912 INFO gateway.pool: 워커 6개 준비 3.3s — 코어 [0, 1, 2, 3, 4, 5], 앞단
cpu [6]
2026-10-12 23:18:21,912 INFO gateway: COPLAND TTS 게이트웨이 0.3.4 (commit 1a2b3c4d5e, 빌드
2026-10-12T15:00:00Z) — :9300, 인증 키 2개, 허용 IP 전체, 라이선스 L-000123 동시 150채널 (valid)
2026-10-12 23:18:22,607 INFO gateway: 예열 1문장 끝 0.7s (실패 0)
```

«코어 [...]»가 합성 워커가 고정된 CPU 번호이고 «앞단 cpu»가 게이트웨이 앞단이 쓰는 CPU 번호입니다.

3. 실제로 합성해 봅니다. `ctts_xxxxxxxx...` 자리에 5.3절에서 받은 user 키를 넣습니다.

```
curl -s -o test.wav -w '%{http_code}\n' http://127.0.0.1:9300/v1/audio/speech \
  -H 'Authorization: Bearer ctts_xxxxxxxx...' -H 'Content-Type: application/json' \
  -d '{"input":"안녕하세요. 문의는 1588-1234로 주세요."}'
```

예상 출력: `200` . `test.wav` (32kHz, 16비트 모노)가 만들어집니다.

6. 설치 후 폴더 구조

```
/opt/copland-tts/
  KEY                     라이선스 키 (발급 포털에서 받은 파일)
  install.sh              설치 프로그램 사본 (key·status·version 명령용)
  VERSION, BUILD_INFO.json  설치한 버전과 빌드 정보
  docs/                   고객용 매뉴얼
  .env                    버전·CPU 고정·워커 수·실행 방식·포트·허용 IP (7.1절)
  docker-compose.yml      실행 구성 (업그레이드 때 덮어쓰지 않음)
  docker-compose.yml.new  (업그레이드 때 새 구성이 다를 경우에만 생김)
  models/                 음성 모델 (라이선스 사본 포함)
  conf/
    keys.tsv              API 키 (이름<TAB>sha256 값<TAB>역할)
    customer.tsv          고객사 사전 (표기<TAB>읽기)
    dicts/                이름 있는 사전 (<이름>.tsv → 요청 dictionary=<이름>)
    prewarm.txt           예열 목록 (기동 때 미리 합성할 문장)
  NOTICE.md               서드파티 라이선스 고지
  copland-machine-key.sh  장비키 확인 프로그램
  qr.awk                  장비키 QR 표시용
```

- 컨테이너는 이 폴더를 같은 경로(`/opt/copland-tts`)에 **읽기 전용**으로 마운트합니다. 설정은 호스트에서 고칩니다.
- 컨테이너 안의 게이트웨이는 UID 1000으로 실행됩니다. 이 폴더의 파일은 UID 1000이 읽을 수 있어야 합니다(설치 프로그램이 만든 권한 그대로 두면 됩니다).
- `conf/` 파일의 사용법은 운영 매뉴얼을 참조합니다.

7. 설정

7.1 `.env`

설치 프로그램이 처음 만드는 `.env` 는 다음과 같습니다.

```
TTS_VERSION=0.3.4
# TTS_CPUSET=0-6
# TTS_WORKERS=6
# TTS_BACKEND=ov-bf16
# TTS_PORT=9300
```

`#` 을 지우고 값을 바꾸면 적용됩니다. 자주 쓰는 값은 다음 7개입니다. 이 밖에 7.3절의 `TTS_*` 옵션도 `.env` 에 적으면 컨테이너에 전달됩니다(7.2절).

변수	비우면 쓰는 값	설명
`TTS_VERSION`	설치한 버전	실행할 이미지 버전. 설치·업그레이드 때 `install.sh` 가 고칩니다
`TTS_CPUSET`	`0-6`	컨테이너가 쓸 CPU 번호. 물리 코어 «워커 수 + 1»개 분량(7.4절)
`TTS_WORKERS`	`6`	합성 워커 수 = 쓸 물리 코어 수
`TTS_BACKEND`	`ov-bf16`	`ov-bf16` = AMX 가속(BF16), `ov-f32` = 기본 실행(FP32, AMX 없는 서버)
`TTS_PORT`	`9300`	받는 포트
`TTS_ALLOW`	(비움 = 전체 허용)	허용할 호출 주소 대역(CIDR), 쉼표로 여러 개
`TTS_TRUST_PROXY`	(비움)	실제 호출 IP 헤더를 믿을 프록시 주소

AMX 없는 서버의 예(물리 13코어: 워커 12 + 앞단 1)는 다음과 같습니다.

```
TTS_VERSION=0.3.4
TTS_CPUSET=0-12
TTS_WORKERS=12
TTS_BACKEND=ov-f32
TTS_ALLOW=10.0.0.11/32,10.0.0.12/32
TTS_CACHE_MB=1024
```

- `TTS_ALLOW` 를 지정해도 서버 자신(루프백 `127.0.0.1` , `::1`)은 늘 허용됩니다. 컨테이너 상태 검사와 `install.sh` 의 확인은 막히지 않습니다.
- 부하 분산기나 모니터링 서버가 `/health` 를 부르면 그 주소도 `TTS_ALLOW` 에 넣습니다.

7.2 `docker-compose.yml` 의 `env_file` 과 `environment`

`docker-compose.yml` 은 `.env` 를 두 가지로 씁니다.

- `env_file: .env` : `.env` 의 모든 값을 컨테이너 환경 변수로 넘깁니다. 7.3절의 `TTS_*` 옵션은 `.env` 에 적기만 하면 됩니다.
- `environment` : 아래 값을 컨테이너에 넣습니다. **`.env` 와 같은 이름이면 `environment` 가 이깁니다.**

```
    env_file: .env
    environment:
      TZ: Asia/Seoul
      COPLAND_HOME: /opt/copland-tts
      TTS_PORT: "${TTS_PORT:-9300}"
      TTS_WORKERS: "${TTS_WORKERS:-6}"
      TTS_BACKEND: "${TTS_BACKEND:-ov-bf16}"
      TTS_KEYS: /opt/copland-tts/conf/keys.tsv
      TTS_DICT: /opt/copland-tts/conf/customer.tsv
      TTS_DICT_DIR: /opt/copland-tts/conf/dicts
      TTS_PREWARM: /opt/copland-tts/conf/prewarm.txt
      TTS_ALLOW: "${TTS_ALLOW:-}"
      TTS_TRUST_PROXY: "${TTS_TRUST_PROXY:-}"
```

- `TTS_PORT` · `TTS_WORKERS` · `TTS_BACKEND` · `TTS_ALLOW` · `TTS_TRUST_PROXY` 는 `.env` 의 값을 받아 넣으므로 `.env` 에서 바꿉니다.
- `TTS_KEYS` · `TTS_DICT` · `TTS_DICT_DIR` · `TTS_PREWARM` 은 경로가 고정되어 있습니다. `.env` 에 적어도 이 값이 이깁니다. 바꾸려면 `docker-compose.yml` 을 고칩니다.
- `TZ: Asia/Seoul` 로 로그 시각이 한국 시각입니다. 키별 일일 한도의 «하루»는 시간대 설정과 관계없이 한국 자정(KST)에 바뀝니다.
- 명령줄 옵션으로 줄 수도 있습니다. 이미지의 기본 명령 뒤에 덧붙습니다.

```
  command: ["--cache-mb", "1024"]
```

- 고친 `docker-compose.yml` 은 업그레이드 때 덮어쓰지 않습니다(9.1절).

7.3 전체 옵션

게이트웨이의 모든 옵션입니다. 환경 변수와 명령줄을 함께 주면 명령줄이 이깁니다. 여러 값을 받는 옵션(`--dict` , `--allow` , `--trust-proxy`)도 같습니다. 명령줄로 주면 명령줄 값만 쓰고, 주지 않으면 환경 변수(쉼표로 구분)를 씁니다. 둘을 합치지 않습니다.

환경 변수	명령줄	기본값	설명
`TTS_HOST`	`--host`	`0.0.0.0`	받을 주소. 컨테이너에서는 이미지가 `--host 0.0.0.0` 을 주므로 바뀌지 않습니다
`TTS_PORT`	`--port`	`9300`	받을 포트
`TTS_WORKERS`	`--workers`	논리 CPU 수/2 − 1 (compose는 `6`)	합성 워커 수 = 물리 코어 수
`TTS_BACKEND`	`--backend`	`ov-bf16`	`ov-bf16` (AMX 가속, BF16), `ov-f32` (기본 실행, FP32), `ort` (대체 실행 방식, 평소 쓰지 않음)
`TTS_MODEL_ROOT`	`--model-root`	`$COPLAND_HOME/models`	모델 폴더
`TTS_DICT`	`--dict`	없음 (compose는 `conf/customer.tsv`)	고객사 사전. 여러 개면 앞이 우선(환경 변수는 쉼표로). 기본 사전은 늘 맨 뒤. 파일이 없으면 경고를 남기고 건너뜁니다
`TTS_DICT_DIR`	`--dict-dir`	없음 (compose는 `conf/dicts`)	이름 있는 사전 폴더. `<이름>.tsv` → 요청 `dictionary=<이름>`
`TTS_KEYS`	`--keys`	없음 (compose는 `conf/keys.tsv`)	API 키 파일. 주지 않으면 인증을 끕니다(개발용). 주었는데 파일이 없으면 기동하지 않습니다
`TTS_LICENSE`	`--license`	`$COPLAND_HOME/KEY`	라이선스 KEY 파일. 찾는 순서: `TTS_LICENSE` → `COPLAND_LICENSE` → `$COPLAND_HOME/KEY` → 앱 폴더의 `KEY`
`TTS_SENTENCE_PAUSE_MS`	`--sentence-pause-ms`	`300`	문장 사이 쉼(ms) 기본값. 요청의 `sentence_pause_ms` 가 이깁니다
`TTS_TRUST_PROXY`	`--trust-proxy`	없음	실제 호출 IP(`X-Real-IP` 헤더만)를 믿을 프록시 주소. `X-Forwarded-For` 는 쓰지 않습니다. IP 주소 그대로 적습니다(대역 불가). 키별 일일 한도와 로그에 씁니다
`TTS_ALLOW`	`--allow`	없음 (전체 허용)	허용 IP 대역(CIDR). 여러 개 가능. 직접 접속한 주소(프록시 뒤면 프록시 주소)로 판단합니다. 루프백은 늘 허용합니다
`TTS_CACHE_MB`	`--cache-mb`	`512`	

환경 변수	명령줄	기본값	설명
			문장 캐시 상한(MB). `0` 이면 캐시를 끕니다. 예열한 문장은 별도로 고정됩니다
`TTS_PREWARM`	`--prewarm`	없음 (compose는 `conf/prewarm.txt`)	예열 목록 파일
`TTS_PREWARM_VOICES`	`--prewarm-voices`	첫 화자(`F1`)	화자를 적지 않은 예열 줄에 쓸 화자. 여러 개 가능
`TTS_PREWARM_RATES`	`--prewarm-rates`	`32000`	예열할 샘플레이트. 여러 개 가능
`TTS_MAX_WAIT`	`--max-wait`	`1.0`	새 요청의 첫 문장 예상 대기가 이 초를 넘으면 429 `overloaded` 로 거절
`TTS_SYNTH_TIMEOUT`	`--synth-timeout`	`30.0`	문장 1개 합성 시간 상한(초). 넘으면 504 `synthesis_timeout`
`TTS_LOG_LEVEL`	`--log-level`	`info`	로그 수준(`debug` , `info` , `warning` , `error`)

`COPLAND_HOME` (compose는 `/opt/copland-tts`)은 KEY와 모델을 찾는 기준 폴더입니다. 바꾸지 않습니다.

7.4 CPU 고정(cpuset) 정하는 법

합성 워커는 물리 코어 하나에 하나씩 고정됩니다. 남는 CPU는 앞단(게이트웨이)이 씁니다. 따라서 `TTS_CPUSET` 에는 **물리 코어 «워커 수 + 1»개**가 들어가야 합니다.

1. 논리 CPU와 물리 코어의 짝을 확인합니다.

```
lscpu -e=CPU,CORE,SOCKET
```

하이퍼스레딩이 켜진 8코어 서버의 출력 예는 다음과 같습니다. `CORE` 값이 같은 CPU끼리 한 물리 코어의 짝입니다.

```
CPU CORE SOCKET
  0    0      0
  1    1      0
  2    2      0
  3    3      0
  4    4      0
  5    5      0
  6    6      0
  7    7      0
  8    0      0
  9    1      0
...
 15    7      0
```

2. 서로 다른 물리 코어에 속한 CPU를 «워커 수 + 1»개 고릅니다.
   - 위 예처럼 짝이 뒤쪽에 있으면 `0-6` 이 물리 7코어입니다(워커 6 + 앞단 1).
   - 짝이 붙어 있는 서버(CPU 0·1이 같은 코어)라면 `0,2,4,6,8,10,12` 처럼 하나씩 건너 고릅니다. 이때 `0-6` 은 물리 4코어밖에 안 되므로 워커 6개가 뜨지 않습니다.
   - 2소켓 서버는 한 소켓(`SOCKET` 값이 같은 CPU) 안에서 고르는 것이 좋습니다.
3. `.env` 에 적습니다.

```
TTS_CPUSET=0-6
TTS_WORKERS=6
```

4. 반영하고 확인합니다(7.5절). 기동 로그의 `워커 6개 준비 … 코어 [0, 1, 2, 3, 4, 5], 앞단 cpu [6]` 에서 워커와 앞단이 다른 CPU에 잡혔는지 봅니다.

워커 수가 cpuset 안의 물리 코어 수보다 많으면 `워커 6개 > 쓸 수 있는 물리 코어 4개` 오류로 기동하지 않습니다.

7.5 설정 반영

바꾼 것	반영 방법
`.env` , `docker-compose.yml`	`cd /opt/copland-tts` 후 `sudo docker compose up -d` (컨테이너를 다시 만듭니다)
`KEY`	`sudo bash /opt/copland-tts/install.sh key` 또는 위와 같이 컨테이너를 다시 만듭니다
`conf/customer.tsv` , `conf/dicts/` , `conf/prewarm.txt`	관리 API `POST /v1/dictionary/reload` (재시작 불필요, 운영 매뉴얼)
`conf/keys.tsv`	관리 API `POST /v1/keys/reload` (재시작 불필요, 운영 매뉴얼)

`docker restart copland-tts` 는 컨테이너를 다시 만들지 않으므로 `.env` 변경이 반영되지 않습니다.

8. 이중화

1. 서버 2대에 같은 패키지와 같은 설정으로 설치합니다.
2. **라이선스는 장비마다 따로 받습니다.** 장비키가 장비마다 다르기 때문입니다. 계약 수량 안에서 장비별로 동시 채널을 나눠 발급합니다.
3. 두 서버의 `conf/` (사전, 예열 목록)를 같게 맞춥니다. API 키는 같은 키 줄을 두 서버의 `keys.tsv` 에 넣으면 웹 서버가 키 하나로 두 서버를 씁니다.
4. 웹 서버의 Java SDK에 주소 2개를 줍니다.

```
CoplandTtsClient tts = CoplandTtsClient.builder()
    .endpoints("http://10.0.0.21:9300", "http://10.0.0.22:9300")
    .apiKey(System.getenv("COPLAND_TTS_KEY"))
    .build();
```

- SDK는 두 서버를 번갈아 씁니다. 연결 실패, 429(`overloaded` · `channel_limit`), 502, 503, 504를 받으면 다음 서버로 넘깁니다.
- 연결에 실패한 서버는 5초 동안 건너뜁니다. 소리를 받기 시작한 뒤에는 다시 보내지 않습니다.
- 별도 L4 장비가 없어도 됩니다. L4를 쓴다면 상태 검사 주소로 `/health` (키 불필요, 정상 200, 워커 재기동 중이나 라이선스 만료면 503)를 씁니다.
- 한 대가 멈춰도 나머지 한 대가 전체 채널을 받을 수 있게 산정하려면, 각 서버를 전체 채널 기준으로 준비합니다.

9. 업그레이드

9.1 새 버전 설치

1. 새 패키지를 반입하고 5.1절대로 체크섬을 확인한 뒤 압축을 풉니다.
2. 새 패키지 폴더에서 설치 프로그램을 실행합니다.

```
cd copland-tts-<새 버전>
sudo bash install.sh
```

3. 설치 프로그램이 하는 일은 다음과 같습니다.
   - 새 이미지를 적재합니다. 이전 버전 이미지는 지우지 않습니다.
   - 설치 경로의 `install.sh`, `VERSION`, `BUILD_INFO.json`, `docs/` 를 새 버전으로 바꿉니다.
   - `.env` 의 `TTS_VERSION` 을 새 버전으로 바꿉니다. 나머지 값은 그대로 둡니다.
   - `KEY`, `conf/` 파일, `.env` 의 다른 설정은 유지합니다. API 키도 새로 만들지 않습니다.
   - `docker-compose.yml` 은 덮어쓰지 않습니다. 새 구성이 다르면 `docker-compose.yml.new` 를 옆에 둡니다.
   - `models/` 가 이미 있으면 덮어쓰지 않습니다.
   - 컨테이너를 새 이미지로 다시 띄우고 `/health` 를 확인합니다.
4. `docker-compose.yml.new` 가 생겼으면 기존 파일과 비교해 필요한 내용을 옮깁니다.

```
diff /opt/copland-tts/docker-compose.yml /opt/copland-tts/docker-compose.yml.new
```

5. `/health` 의 `version` 이 새 버전인지 확인합니다.

```
curl -s http://127.0.0.1:9300/health
```

새 버전의 변경 사항은 패키지의 `CHANGELOG.md` 를 확인합니다. 모델이 바뀌는 버전이면 COPLAND가 별도로 안내합니다.

9.2 되돌리기

이전 버전 이미지가 남아 있으므로 `TTS_VERSION` 만 되돌리면 됩니다.

1. 남아 있는 이미지를 확인합니다.

```
sudo docker images copland-tts-gw
```

2. `/opt/copland-tts/.env`의 `TTS_VERSION`을 이전 버전으로 고칩니다.

```
TTS_VERSION=0.3.4
```

3. 컨테이너를 다시 만듭니다.

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

4. `/health`의 `version`이 이전 버전인지 확인합니다.

이전 버전 이미지를 지웠다면 이전 버전 패키지를 다시 풀어 그 폴더에서 `sudo bash install.sh`를 실행합니다. 이전 패키지 파일을 보관해 두기를 권합니다.

10. 라이선스 갱신·재발급

10.1 새 라이선스 키 넣기

계약 갱신, 채널 수 변경, 재발급 때 씁니다.

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

1. 장비키가 표시됩니다.
2. 5.4절과 같은 화면에서 키를 넣습니다(파일 또는 한 줄씩).
3. 확인되면 컨테이너를 다시 만들어 새 라이선스를 읽습니다.
4. `curl -s http://127.0.0.1:9300/health` 의 `license` 에서 새 만료일과 채널 수를 확인합니다.

게이트웨이는 기동할 때 KEY를 읽습니다. 파일만 바꾸고 컨테이너를 다시 만들지 않으면 반영되지 않습니다.

10.2 만료와 유예 기간

기간	합성 요청	표시
만료일까지	정상	`/health` 의 `license.state` = `valid`
만료 뒤 유예 기간(기본 14일)	정상. 응답에 `X-License-Warning: expired 2027-12-31; grace until +14d` 헤더가 붙습니다	`state` = `grace` . 기동 로그에 `라이선스 만료 2027-12-31 — 유예 14일 뒤 합성 거부`
유예 기간 뒤	403 `license_expired` (`라이선스 만료 (2027-12-31, 유예 14일 지남)`)	`/health` 가 503, `status` = `license_expired` , `license.state` = `expired`

- 유예 일수는 라이선스마다 다를 수 있습니다. KEY 파일의 `# 만료:` 줄과 `/health` 로 확인합니다.
- 유예 기간이 지나면 `/health` 가 503을 돌려주므로 부하 분산기와 SDK가 다른 서버로 넘깁니다. 이중화한 두 서버가 함께 만료되면 넘길 곳이 없습니다. 만료를 놓치지 않도록 `/health` 의 `license.state` 나 `/v1/license` (관리자 키)의 `days_left` 를 감시합니다(운영 매뉴얼 «모니터링»).
- 만료 전에 발급 포털에서 갱신 키를 받아 10.1절대로 넣습니다.

10.3 장비 교체

1. 새 장비에서 장비키를 확인합니다(3장).
2. 발급 포털에서 새 장비키로 라이선스를 받습니다.
3. 새 장비에 설치합니다(5장). 이전 장비의 `conf/` 와 `.env` 를 옮기면 같은 설정이 됩니다(KEY는 옮겨도 쓸 수 없습니다).

4. 이전 장비에서 제거하고(13장), 발급 포털에서 이전 장비의 폐기를 처리합니다. 라이선스 키는 오프라인이라 원격으로 끌 수 없으므로 «장비에서 지웠다»는 확인을 거쳐야 계약 수량이 돌아옵니다.

CPU나 등록된 NIC를 바꾼 경우도 장비 교체와 같이 처리합니다. 포털 절차는 «라이선스 발급 포털 사용 안내»를 참조합니다.

11. 버전 확인

문의할 때는 아래 값을 함께 알려 주십시오. 정확한 빌드를 찾는 데 씁니다.

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

예상 출력은 다음과 같습니다(일부 줄임, 커밋·시각은 예시).

```
COPLAND TTS 0.3.4 (v0.3.4, 2026-10-12T15:00:00Z)
{"product": "COPLAND TTS", "version": "0.3.4", "commit": "1a2b3c4d5e", "commit_full": "...",
 "describe": "v0.3.4", "branch": "main", "license_lib": "v2.0", "built_at":
"2026-10-12T15:00:00Z",
 "built_by": "COPLAND"}
{"status":"ok","version":"0.3.4","commit":"1a2b3c4d5e","workers":6,"license":
{"license_id":"L-000123","state":"valid","expires":"2027-12-31","channels":150}}
```

- 첫 줄: 패키지 버전, 빌드 이름, 빌드 시각(UTC)
- 둘째 묶음: 패키지의 `BUILD_INFO.json`
- 마지막 줄: 실행 중인 게이트웨이의 `/health` . `version` 과 `commit` 이 실행 중인 빌드입니다

게이트웨이만 확인하려면 다음을 실행합니다.

```
curl -s http://127.0.0.1:9300/health
```

문의할 때 알려 줄 값: `version` , `commit` , `license_id` , 문제가 난 요청의 `X-Request-Id` (응답 헤더, 로그의 `rid=`).

12. 문제 해결

오류 문구는 화면(`install.sh`), 컨테이너 로그(`docker logs copland-tts`), API 응답에 나옵니다. 아래 «증상»의 문구로 찾습니다.

12.1 설치·기동

증상	원인	조치
`오류: root 로 실행하세요 (sudo bash install.sh)`	일반 사용자로 실행함	`sudo bash install.sh` 로 실행합니다
`오류: docker 가 없습니다. Docker Engine(+compose)을 먼저 설치하세요.`	Docker 미설치	Docker Engine과 compose 플러그인을 설치합니다
`오류: docker 데몬에 연결할 수 없습니다`	Docker 서비스가 멈춤	`sudo systemctl start docker` 후 다시 실행합니다
`⚠ 이 CPU 는 AMX 가 없습니다 → …`	AMX 없는 CPU	설치 뒤 `.env` 에 `TTS_BACKEND=ov-f32` 를 넣고, 코어 산정표(2.2절)의 «AMX 없음» 열로 `TTS_CPUSET` · `TTS_WORKERS` 를 정합니다
`오류: 파일이 없습니다: <경로>`	KEY 파일 경로가 틀림	경로를 확인하고 다시 실행합니다
`✗ 한 줄은 4자 × 8묶음 + 확인 1자 (33자) 입니다.`	줄 글자 수가 맞지 않음	그 줄을 다시 입력합니다. 줄 끝의 확인 문자까지 넣습니다
`✗ N번째 줄에 틀린 글자가 있습니다.`	옮겨 적다 틀림	그 줄만 다시 입력합니다. 0과 O, 1과 I·L은 같게 처리되므로 다른 글자를 봅니다
`기동 확인 실패 — docker logs copland-tts`	3분 안에 `/health` 가 200이 아님	`docker logs copland-tts` 로 원인을 봅니다 (아래 행)
로그 `워커 6개 > 쓸 수 있는 물리 코어 4개`	`TTS_CPUSET` 안의 물리 코어가 워커 수보다 적음	7.4절대로 cpuset을 다시 정하거나 `TTS_WORKERS` 를 줄입니다
컨테이너 생성 오류(지정한 CPU가 없음)	`TTS_CPUSET` 에 없는 CPU 번호가 있음. 기본값 `0-6` 은 논리 CPU 7개 이상이 필요	`lscpu -e=CPU,CORE` 로 확인하고 `.env` 의 `TTS_CPUSET` 을 고칩니다
로그 `[Errno 98] error while attempting to bind on address ('0.0.0.0', 9300): address already in use`	다른 프로그램이 9300을 씀	`sudo ss -ltnp 'sport = :9300'` 로 확인하고, 그 프로그램을 멈추거나 `.env` 에 `TTS_PORT` 를 다른 값으로 줍니다
로그 `API 키 파일이 없다: /opt/copland-tts/conf/keys.tsv (python -m gateway.auth new <이름> 으로 만든 줄을 넣는다)`	키 파일이 지워짐	운영 매뉴얼의 절차로 키를 만들어 파일에 넣고 다시 띄웁니다
로그 `사전 파일이 없어 건너뜀: <경로>` (경고)	고객사 사전 파일이 없음	사전이 필요 없으면 그대로 둡니다. 필요하면 `conf/customer.tsv` 를 만듭니다

12.2 라이선스

라이선스 문제로 기동하지 못하면 컨테이너 로그에 다음 3줄이 나오고 종료(코드 3)합니다. 컨테이너는 자동으로 다시 시도합니다.

```
... CRITICAL gateway: 라이선스: <원인>
... CRITICAL gateway: 이 장비의 장비키: KYS1-W8VT
... CRITICAL gateway: COPLAND 발급 포털(https://copland.ai/license/)에 장비키를 넣어 라이선스 키를 받아 /opt/copland-tts/KEY 에 넣은 뒤 다시 기동하세요.
```

증상(«원인» 자리 문구)	원인	조치
`KEY 파일이 없음: /opt/copland-tts/KEY`	KEY가 없음	`sudo bash /opt/copland-tts/install.sh key` 로 넣습니다
`/opt/copland-tts/KEY 가 폴더다 — 지우고 라이선스 키 파일로 넣어야 한다`	KEY 자리에 폴더가 생김(파일 없이 마운트한 경우 등)	폴더를 지우고 `sudo bash /opt/copland-tts/install.sh key` 로 넣습니다
`이 장비용 라이선스가 아님 (CPU 가 다름)`	다른 장비의 KEY이거나 CPU를 바꿈	로그의 장비키로 다시 발급받습니다(10.3절)
`이 장비용 라이선스가 아님 (등록된 NIC 가 없음)`	다른 장비의 KEY이거나 등록된 NIC를 제거함	NIC를 되돌리거나 새 장비키로 다시 발급받습니다
`다른 제품의 라이선스다 (…) — COPLAND TTS 라이선스가 필요`	다른 COPLAND 제품의 키	포털에서 제품을 COPLAND TTS로 골라 받습니다
`라이선스 서명이 맞지 않음 (변조되었거나 다른 발급자)`	키 내용이 바뀜(확인 문자까지 맞춰 고친 경우 등)	포털에서 받은 원본 KEY를 다시 넣습니다
`N번째 줄에 틀린 글자가 있다`	KEY 파일의 N번째 키 줄이 잘못됨	원본 KEY로 바꾸거나 그 줄을 고칩니다
`라이선스 키가 짧음 (줄이 빠졌다)` 또는 `라이선스 키 길이가 맞지 않음 (줄이 빠지거나 더 있다)`	키 줄이 빠지거나 더 있음	키 줄 4개가 모두 있는지 확인합니다
`이 장비에서 COPLAND TTS 이(가) 이미 실행 중`	같은 장비에 게이트웨이가 이미 떠 있음(한 장비에 하나만 실행)	`sudo docker ps` 로 다른 컨테이너·프로세스를 찾아 멈춥니다
`장비 정보를 읽지 못해 라이선스를 확인할 수 없다: 물리 NIC 를 찾지 못함 — 컨테이너라면 network_mode: host 로 띄워야 한다`	bridge 네트워크로 띄움	`docker-compose.yml` 의 `network_mode: host` 를 되살립니다

12.3 운영 중 API 오류

API 오류는 `{"error":{"message":...,"type":...,"code":...}}` 형식입니다. `code` 로 찾습니다. 전체 코드표는 API 개발자 매뉴얼을 참조합니다.

증상	원인	조치
401 `invalid_api_key` (`API 키가 없거나 맞지 않음`)	키가 없거나 틀림, 키 파일에서 지워짐	웹 서버의 키 설정과 `conf/keys.tsv` 를 확인합니다
403 `ip_not_allowed` (`허용되지 않은 주소: <주소>`)	호출 주소가 `TTS_ALLOW` 밖	그 주소를 `TTS_ALLOW` 에 넣고 컨테이너를 다시 만듭니다
403 `admin_required` (`관리자 키가 필요함`)	관리 API를 일반 키로 부름	관리자 키를 씁니다
403 `license_expired` , `/health` 503 `"status":"license_expired"`	유예 기간까지 지남	갱신 키를 받아 넣습니다(10장)
503 `restarting` (`합성 워커 재기동 중`), `/health` 503	합성 워커가 비정상 종료해 다시 뜨는 중(수 초). 로그 `합성 워커가 죽었다 — 풀 재기동 (시도 1)`	대개 저절로 회복됩니다. SDK는 다른 서버로 넘깁니다. 자주 반복되면 메모리 부족 (`dmesg` 의 OOM)을 확인합니다
429 `overloaded` (`합성 대기열이 가득 참`)	동시 요청이 처리 능력을 넘음	일시적이면 SDK가 다른 서버로 넘깁니다. 자주 나면 워커(물리 코어)를 늘리거나 서버를 추가합니다
429 `channel_limit` (`라이선스 동시 채널 N개를 모두 쓰는 중`)	재생 중인 채널이 라이선스 채널 수에 닿음	동시 채널 수를 늘린 라이선스를 받거나 서버를 추가합니다
429 `daily_quota`	키에 건 호출 IP 일일 한도를 다 씀	한국 자정(KST)에 초기화됩니다. 한도는 운영 매뉴얼 «API 키 관리»를 참조합니다
400 `invalid_json` (`요청 본문이 올바른 JSON 이 아님 …`)	요청 본문이 JSON이 아님	웹 서버가 `Content-Type: application/json` 과 올바른 JSON을 보내는지 확인합니다
504 `synthesis_timeout`	문장 1개 합성이 30초를 넘음	서버 과부하나 CPU 고정을 확인합니다

13. 제거

1. 필요하면 백업합니다(`KEY` , `conf/` , `.env` , 고친 `docker-compose.yml`).

```
sudo tar czf /root/copland-tts-backup.tar.gz -C /opt/copland-tts KEY .env docker-compose.yml
          conf
```

2. 컨테이너를 멈추고 지웁니다.

```
cd /opt/copland-tts
sudo docker compose down
```

3. 이미지를 지웁니다.

```
sudo docker images copland-tts-gw
sudo docker image rm copland-tts-gw:0.3.4
```

4. 설치 폴더를 지웁니다. 패키지 폴더(예: `/root/copland-tts-0.3.4`)가 남아 있으면 함께 지웁니다.

```
sudo rm -rf /opt/copland-tts
sudo rm -rf /root/copland-tts-0.3.4
```

5. 장비를 더 쓰지 않으면 발급 포털에서 이 장비의 폐기를 처리합니다(10.3절).

확인 방법: `sudo docker ps -a | grep copland-tts` 가 아무것도 출력하지 않으면 제거된 것입니다.

문의: COPLAND 기술지원 (연락처는 계약서 참조)