COPLAND

기술 문서

COPLAND TTS

Java SDK 매뉴얼

Java 8 이상 WAS에 SDK를 적용하는 방법입니다. 서블릿과 Spring 중계 예제, 이중화 구성 시 장애 전환을 설명합니다.

적용 버전

COPLAND TTS 0.3.4(게이트웨이)
Java SDK 0.2.1

문서 갱신일

2026-10-12

copland.ai/tts

차례

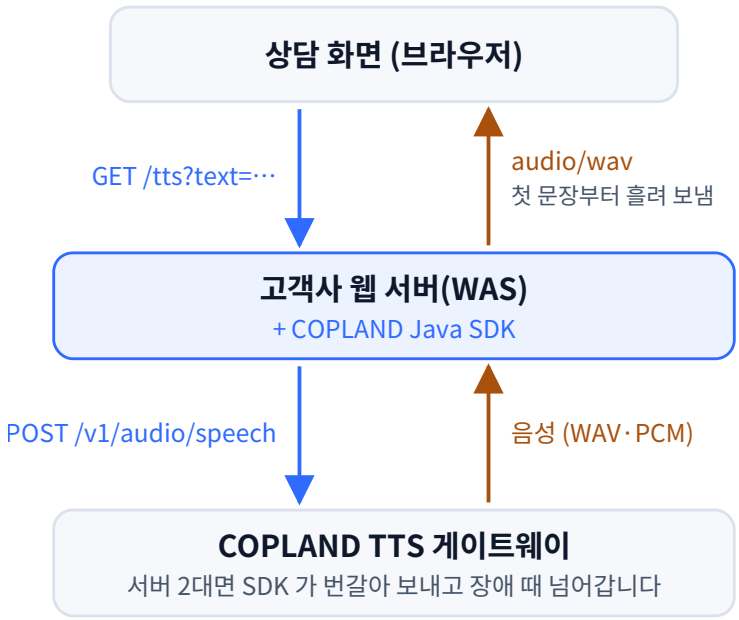
1. 개요
    1.1 요구 사항
    1.2 배포물
2. 설치
    2.1 jar를 클래스패스에 넣기
    2.2 Maven 로컬 저장소에 설치
    2.3 Gradle
    2.4 사내 Maven 저장소(Nexus·Artifactory)에 등록
3. 빠른 시작
4. API 레퍼런스
    4.1 CoplandTtsClient
    4.2 CoplandTtsClient.Builder
    4.3 SpeechRequest
    4.4 SpeechStream
    4.5 AudioFormat
    4.6 Voices
    4.7 NormalizeResult
    4.8 Health
    4.9 TtsException
    4.10 Wav
5. 실행 예제로 확인하기
    5.1 흘려 받기와 파일 저장
    5.2 정규화 미리보기
    5.3 오류 처리
6. 사용 패턴
    6.1 전체 받기와 흘려 받기 고르기
    6.2 서블릿 중계 (Java 8, javax.servlet)
    6.3 Spring MVC 중계 (StreamingResponseBody)
    6.4 파일 저장
    6.5 비동기 (CompletableFuture + 전용 Executor)
    6.6 브라우저 `<audio>` 연동
    6.7 녹취 시스템에 PCM 넘기기
    6.8 SSML
    6.9 사전 선택

6.10 요청 번호 연계
7. 운영 설정
7.1 여러 서버와 장애 전환
7.2 시간 제한 정하기
7.3 프록시
7.4 HTTPS (사설 인증서)
7.5 로그
7.6 스레드 안전과 싱글턴
8. 문제 해결
8.1 Java 8에서 copland.ai 인증서 오류
8.2 흘려 받는 중 끊김
9. 명령줄 도구 (Cli)
9.1 상태 확인
9.2 정규화 미리보기
9.3 합성
10. 관련 문서

- 적용 버전: **COPLAND TTS 0.3.4**(게이트웨이), Java SDK 0.2.1
- 문서 갱신일: 2026-10-12
- 독자: 상담 화면·업무 시스템을 만드는 **Java 웹 서버(WAS) 개발자**

1. 개요

Java SDK는 웹 서버에서 COPLAND TTS 게이트웨이를 부르는 클라이언트 라이브러리입니다. 문장을 넘기면 WAV 또는 PCM 음성을 돌려받습니다.



SDK가 놓이는 자리

SDK가 대신 처리하는 일은 다음과 같습니다.

- 게이트웨이 여러 대에 번갈아 보내고, 죽은 서버는 건너뜁니다(장애 전환).
- 소리를 첫 문장부터 흘려 받습니다. 긴 안내문도 첫 소리가 빨리 나옵니다.
- 오류를 `TtsException` 하나로 모아, 다시 보내도 되는 오류인지 알려 줍니다.
- 요청마다 요청 번호를 붙여 게이트웨이 로그와 맞춰 볼 수 있게 합니다.

1.1 요구 사항

항목	내용
Java	**Java 8 이상** (Java 8 바이트코드로 빌드)
외부 의존성	**없음**. JDK의 `HttpURLConnection` 만 씁니다. 고객 라이브러리와 충돌하지 않습니다
WAS	JEUS, WebLogic, Tomcat, Spring Boot 등 서블릿 컨테이너 모두 가능
배포 형태	jar 하나(약 35KB)
패키지 이름	`ai.copland.tts`
Maven 좌표	`ai.copland:copland-tts-client:0.2.1`

1.2 배포물

납품 패키지의 `sdk/` 폴더에 다음 파일이 있습니다.

파일	내용
`sdk/copland-tts-client-0.2.1.jar`	라이브러리. 명령줄 확인 도구(`java -jar`)도 들어 있습니다
`sdk/copland-tts-client-0.2.1-sources.jar`	소스. IDE에서 설명(Javadoc)과 소스를 볼 때 씁니다
`sdk/README.md`	요약 안내

2. 설치

2.1 jar를 클래스패스에 넣기

빌드 도구 없이 쓰는 경우입니다.

1. `copland-tts-client-0.2.1.jar` 를 웹 애플리케이션의 `WEB-INF/lib/` 에 복사합니다.
2. WAS를 다시 기동하거나 애플리케이션을 다시 배포합니다.

명령줄에서 시험할 때는 `-cp` 로 넘깁니다.

```
javac -cp copland-tts-client-0.2.1.jar Hello.java
java  -cp copland-tts-client-0.2.1.jar:. Hello        # Windows 는 ; 로 구분
```

확인 방법: 아래 명령으로 jar가 정상인지 봅니다.

```
java -jar copland-tts-client-0.2.1.jar --help
```

예상 출력:

```
사용: java -jar copland-tts-client.jar --url http://host:9300[,http://host2:9300] [--key KEY]
       [--voice F1] [--rate 32000] [--out out.wav] [--stream] "읽을 문장"
       --normalize "문장"   정규화 미리보기
       --health             게이트웨이 상태
환경 변수 COPLAND_TTS_URL, COPLAND_TTS_KEY 도 읽는다.
```

2.2 Maven 로컬 저장소에 설치

개발 PC의 로컬 저장소(`~/.m2`)에 넣고 의존성으로 씁니다.

1. `sdk/` 폴더에서 다음을 실행합니다.

   ```
   mvn install:install-file \
     -Dfile=copland-tts-client-0.2.1.jar \
     -Dsources=copland-tts-client-0.2.1-sources.jar \
     -DgroupId=ai.copland -DartifactId=copland-tts-client -Dversion=0.2.1 -Dpackaging=jar
   ```

2. `pom.xml` 에 의존성을 넣습니다.

```xml
<dependency>
  <groupId>ai.copland</groupId>
  <artifactId>copland-tts-client</artifactId>
  <version>0.2.1</version>
</dependency>
```

확인 방법: 1번 명령의 마지막에 `BUILD SUCCESS` 가 나옵니다.

예상 출력(일부):

```
[INFO] Installing /.../copland-tts-client-0.2.1-sources.jar to /.../.m2/repository/ai/copland/
copland-tts-client/0.2.1/copland-tts-client-0.2.1-sources.jar
[INFO] BUILD SUCCESS
```

2.3 Gradle

로컬 저장소에 설치했다면 `mavenLocal()` 을 씁니다.

```
repositories {
    mavenLocal()
    mavenCentral()
}
dependencies {
    implementation 'ai.copland:copland-tts-client:0.2.1'
}
```

저장소 없이 jar를 프로젝트에 넣어 쓸 수도 있습니다.

```
dependencies {
    implementation files('libs/copland-tts-client-0.2.1.jar')
}
```

2.4 사내 Maven 저장소(Nexus·Artifactory)에 등록

여러 개발자가 함께 쓰면 사내 저장소에 한 번 올립니다.

```
mvn deploy:deploy-file \
  -Dfile=copland-tts-client-0.2.1.jar \
  -Dsources=copland-tts-client-0.2.1-sources.jar \
  -DgroupId=ai.copland -DartifactId=copland-tts-client -Dversion=0.2.1 -Dpackaging=jar \
  -DrepositoryId=internal-releases \
  -Durl=https://nexus.example.local/repository/maven-releases/
```

- `-DrepositoryId` 는 `~/.m2/settings.xml` 의 `<server><id>` 와 같아야 합니다(저장소 계정).
- `-Durl` 은 고객사 저장소 주소로 바꿉니다.
- 올린 뒤에는 2.2의 `<dependency>` 를 그대로 씁니다.

3. 빠른 시작

자사 서버에 게이트웨이를 설치하기 전에, COPLAND가 운영하는 **시험 엔드포인트**로 연동을 먼저 확인할 수 있습니다. API 키는 필요 없습니다.

```
import ai.copland.tts.*;
import java.nio.file.*;

public class Hello {
    public static void main(String[] args) throws Exception {
        CoplandTtsClient tts = CoplandTtsClient.builder()
            .endpoints("https://copland.ai/tts/sandbox")
            .build();
        byte[] wav = tts.synthesize("문의는 1588-1234로 주세요.");
        Files.write(Paths.get("hello.wav"), wav);
        System.out.println("hello.wav " + wav.length + "바이트");
    }
}
```

1. 위 코드를 `Hello.java` 로 저장합니다.
2. 컴파일하고 실행합니다.

```
javac -cp copland-tts-client-0.2.1.jar Hello.java
java  -cp copland-tts-client-0.2.1.jar:. Hello
```

예상 출력:

```
hello.wav 198222바이트
```

확인 방법: `hello.wav` 를 재생하면 "문의는 일오팔팔 일이삼사로 주세요."로 들립니다. 전화번호는 한 자리씩 읽습니다.

시험 엔드포인트의 제한은 다음과 같습니다.

- 쓸 수 있는 API는 합성·정규화·화자 목록·모델 목록·상태(`/health`)입니다. 관리 API는 막혀 있습니다.
- **접속 주소(IP)마다 하루 100회**까지 합성합니다. 넘으면 429 `daily_quota` 를 돌려주고, 한국 자정에 초기화됩니다. 남은 횟수는 `SpeechStream.getQuotaRemaining()` 으로 봅니다(4.4).
- 성능은 시연용 소형 장비 기준입니다(AMX 없음). 첫 소리 지연과 동시 처리 성능은 실제 설치 장비에서 측정합니다.
- 입력 문장은 저장하지 않습니다. 그래도 실제 고객 정보는 넣지 않습니다.

자사 게이트웨이를 설치한 뒤에는 주소와 API 키만 바꿉니다.

```
CoplandTtsClient tts = CoplandTtsClient.builder()
    .endpoints("http://10.0.0.11:9300", "http://10.0.0.12:9300")
    .apiKey(System.getenv("COPLAND_TTS_KEY"))      // 예: ctts_xxxxxxxxxxxxxxxxxxxxxxxxxx
    .build();
```

API 키는 게이트웨이 설치 때 설치 프로그램이 한 번 보여 줍니다. 추가 키는 게이트웨이 관리자가 만듭니다(운영 매뉴얼 참조). 키는 소스에 적지 말고 환경 변수나 WAS 설정 파일에서 읽습니다.

4. API 레퍼런스

모든 클래스는 `ai.copland.tts` 패키지에 있습니다.

클래스	역할
`CoplandTtsClient`	게이트웨이 클라이언트. 애플리케이션에 하나 만들어 공유합니다
`CoplandTtsClient.Builder`	클라이언트 설정(주소·키·시간 제한·장애 전환)
`SpeechRequest`	합성 요청(불변). `SpeechRequest.Builder` 로 만듭니다
`SpeechStream`	흘려 받는 음성(`InputStream`). 반드시 닫습니다
`AudioFormat`	응답 형식 `WAV` · `PCM`
`Voices`	화자 이름 상수 `F1` ~ `F5` , `M1` ~ `M5`
`NormalizeResult`	정규화 미리보기 결과
`Health`	게이트웨이 한 대의 상태
`TtsException`	호출 실패 (`RuntimeException`)
`Wav`	WAV 헤더 도우미
`Cli`	명령줄 확인 도구(`java -jar`). 9장 참조

4.1 CoplandTtsClient

스레드 안전합니다. 애플리케이션에 **하나만** 만들어 모든 요청 스레드가 공유합니다(Spring이면 빈 하나).

상수	값
`CoplandTtsClient.VERSION`	`"0.2.1"`

builder()

```
public static CoplandTtsClient.Builder builder()
```

설정용 빌더를 만듭니다. 4.2를 봅니다.

synthesize(String text)

```
public byte[] synthesize(String text)
```

- 기본 설정(화자 `F1`, 32kHz, WAV)으로 합성해 **WAV 전체**를 돌려줍니다.
- `SpeechRequest.builder(text).build()` 와 같습니다.
- 예외: `IllegalArgumentException` (빈 글·5000자 초과), `TtsException` (호출 실패)

synthesize(SpeechRequest req)

```
public byte[] synthesize(SpeechRequest req)
```

- 게이트웨이가 끝까지 합성한 뒤 한 번에 받습니다.
- WAV면 헤더의 길이 칸이 정확합니다. 파일 저장·첨부에 씁니다.
- 바로 재생할 거면 `stream()` 이 첫 소리가 빠릅니다.
- 전체 합성이 끝날 때까지 응답이 오지 않습니다. 그 시간이 `readTimeoutMillis` 를 넘으면 `response_timeout` 으로 끝납니다. 긴 글이면 `stream()` 을 쓰거나 `readTimeoutMillis` 를 늘립니다(7.2 참조).
- 예외: `TtsException`. 받는 중 끊기면 코드 `stream_broken` 입니다.

```
byte[] wav = tts.synthesize(SpeechRequest.builder("고객님의 이번 달 청구 금액은 32,500원입니다.")
    .voice(Voices.M1).build());
```

stream(SpeechRequest req)

```
public SpeechStream stream(SpeechRequest req)
```

- 흘려 받습니다. 첫 문장이 합성되는 즉시 읽을 수 있습니다.
- 돌려받은 `SpeechStream` 은 **반드시 닫습니다**(try-with-resources).
- WAV면 앞 44바이트가 헤더이고, 길이를 모르므로 크기 칸이 `0xFFFFFFFF` 입니다.
- 예외: `TtsException`. 이 예외는 소리를 받기 전에만 납니다. 그래서 웹 응답을 보내기 전에 HTTP 오류로 바꿀 수 있습니다.

```
try (SpeechStream s = tts.stream(req)) {
    s.transferTo(response.getOutputStream());
}
```

synthesizeTo(SpeechRequest req, OutputStream out)

```
public long synthesizeTo(SpeechRequest req, OutputStream out) throws IOException
```

- 흘려 받으면서 `out` 으로 그대로 보냅니다. 보낸 바이트 수를 돌려줍니다.
- `stream()` + `transferTo()` + `close()` 를 한 번에 합니다. `out` 은 닫지 않습니다.
- WAV면 스트리밍 헤더(크기 칸 `0xFFFFFFFF`)가 그대로 갑니다.
- 예외: `TtsException` (시작 전 실패), `IOException` (보내는 중 끊김)

synthesizeToFile(SpeechRequest req, File file)

```
public void synthesizeToFile(SpeechRequest req, File file) throws IOException
```

- `synthesize(req)` 로 전체를 받아 파일로 씁니다. WAV 헤더 길이가 정확합니다.
- 예외: `TtsException` , `IOException` (파일 쓰기)

synthesizeAsync(SpeechRequest req, Executor executor)

```
public CompletableFuture<byte[]> synthesizeAsync(SpeechRequest req, Executor executor)
```

- `executor` 에서 `synthesize(req)` 를 돌립니다.
- 실패하면 `CompletionException` 으로 감싼 `TtsException` 으로 완료됩니다.
- 네트워크를 기다리는 작업이므로 **전용 Executor**를 넘깁니다(6.5 참조).

normalize(String text) / normalize(String text, String dictionary)

```
public NormalizeResult normalize(String text)
public NormalizeResult normalize(String text, String dictionary)
```

- 정규화 미리보기입니다. 엔진에 실제로 들어갈 한글 읽기를 돌려줍니다. 음성은 만들지 않습니다.
- `dictionary` 는 게이트웨이에 둔 이름 있는 사전입니다. `null` 이면 기본 사전입니다.
- 사전 검수, 화면의 "읽기 미리보기"에 씁니다.
- 예외: `TtsException` (예: `invalid_dictionary` , `invalid_ssml`)

voices()

```
public List<String> voices()
```

쓸 수 있는 화자 목록을 돌려줍니다. 예: `[F1, F2, F3, F4, F5, M1, M2, M3, M4, M5]`

usage()

```
public Map<String, Object> usage()
```

- 게이트웨이가 기동한 뒤 API 키 이름별 누적 사용량입니다(원본 JSON을 `Map` 으로).
- **관리자 키**가 필요합니다. 일반 키로 부르면 403 `admin_required` 입니다.
- 응답이 온 게이트웨이 한 대의 값입니다. 여러 대면 대마다 다릅니다.

```
{since=2026-10-12T23:18:18, keys={web01={requests=16, chars=499, audio_sec=92.0}}}
```

license()

```
public Map<String, Object> license()
```

- 응답한 게이트웨이의 라이선스 상태입니다(원본 JSON을 `Map` 으로).
- **관리자 키**가 필요합니다. 일반 키로 부르면 403 `admin_required` 입니다. 장비키·고객명이 들어 있기 때문입니다.
- 주요 키: `license_id` , `customer` , `entitlements.channels` (허가 동시 채널), `expires` , `grace_days` , `state` (`valid` · `grace` · `expired`), `days_left` , `active_channels` , `inflight`

```
{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=..., customer=...,
active_channels=6, inflight=0}
```

health()

```
public List<Health> health()
```

- 등록한 **모든** 게이트웨이에 하나씩 `/health` 를 묻습니다. 장애 전환을 하지 않습니다.
- 예외를 던지지 않습니다. 연결이 안 되면 그 서버를 `unreachable` 로 돌려줍니다.
- API 키가 필요 없습니다.
- WAS 기동 점검·모니터링 화면에 씁니다.

```
for (Health h : tts.health()) {
    System.out.println(h);
}
```

```
http://localhost:9399 unreachable (Connection refused (Connection refused))
http://localhost:9300 ok v0.3.4 workers=2
```

reloadDictionaries()

```
public void reloadDictionaries()
```

- **모든** 게이트웨이에 사전 다시 읽기를 보냅니다. **관리자 키**가 필요합니다.
- 사전 파일을 바꾼 뒤 부릅니다. 게이트웨이는 문장 캐시도 비웁니다.
- 한 대라도 실패하면 `TtsException`을 던집니다. 메시지에 실패한 서버 목록이 들어 있습니다.
- 일반 키로 부르면 403 `admin_required` 입니다.

reloadKeys()

```
public void reloadKeys()
```

모든 게이트웨이에 API 키 파일 다시 읽기를 보냅니다. **관리자 키**가 필요합니다. 실패 처리는 `reloadDictionaries()`와 같습니다.

getEndpoints()

```
public List<String> getEndpoints()
```

등록한 게이트웨이 주소 목록입니다(끝의 `/`는 뗀 값).

4.2 CoplandTtsClient.Builder

메서드	기본값	범위·규칙	설명
`endpoints(String... urls)`	(필수)	하나 이상	게이트웨이 주소. 여러 대면 번갈아 보냅니다
`endpoint(String url)`	–	`http://` 또는 `https://` 로 시작	주소 하나 추가. 끝의 `/` 는 뗍니다
`apiKey(String)`	없음	빈 값· `null` 이면 키 없음	`Authorization: Bearer <키>` 로 보냅니다
`model(String)`	`copland-cpu-tts`	`copland-cpu-tts` · `tts-1` · `tts-1-hd`	바꿀 일이 거의 없습니다
`connectTimeoutMillis(int)`	2000	밀리초	연결 제한 시간. 짧아야 죽은 서버를 빨리 건너뜁니다
`readTimeoutMillis(int)`	30000	밀리초	받는 바이트 사이의 최대 간격. 7.2 참조
`maxAttempts(int)`	max(2, 서버 수)	0 이하면 기본값	한 요청의 최대 시도 횟수
`failoverCooldownMillis(long)`	5000	밀리초	연결에 실패한 서버를 건너뛰는 시간
`proxy(Proxy)`	`Proxy.NO_PROXY`	`null` 이면 JVM 설정	기본은 시스템 프록시를 쓰지 않습니다
`sslSocketFactory(SSLSocketFactory)`	JVM 기본	–	https 게이트웨이(사설 인증서)용
`hostnameVerifier(HostnameVerifier)`	JVM 기본	–	https 호스트 이름 확인
`userAgent(String)`	없음	–	`User-Agent: copland-tts-java/0.2.1 <이름>` 으로 붙습니다
`build()`	–	–	클라이언트를 만듭니다

예외:

- `endpoint("")` → `IllegalArgumentException: 빈 주소`
- `endpoints("localhost:9300")` → `IllegalArgumentException: http:// 또는 https:// 로 시작해야 함: localhost:9300`
- 주소 없이 `build()` → `IllegalStateException: endpoints 를 하나 이상 지정`

```
CoplandTtsClient tts = CoplandTtsClient.builder()
    .endpoints("http://10.0.0.11:9300", "http://10.0.0.12:9300")
    .apiKey(System.getenv("COPLAND_TTS_KEY"))
    .connectTimeoutMillis(2000)
    .readTimeoutMillis(30000)
    .failoverCooldownMillis(5000)
    .userAgent("counter-web")
    .build();
```

4.3 SpeechRequest

합성 요청입니다. 만든 뒤에는 바뀌지 않으므로 여러 스레드가 같이 써도 됩니다.

상수	값
`SpeechRequest.SAMPLE_RATES`	`{16000, 22050, 24000, 32000, 44100}`
`SpeechRequest.MAX_INPUT_CHARS`	`5000` (SSML 태그 포함)

builder(String input)

```
public static SpeechRequest.Builder builder(String input)
```

- 읽을 글을 받습니다. `<speak>` 로 시작하면 SSML로 읽습니다.
- 예외: `IllegalArgumentException`
  - 빈 글·공백만 → `input 이 비어 있음`
  - 5000자 초과 → `input 은 5000자 이하 (5001자)`

Builder 메서드

값의 범위는 SDK가 먼저 검사합니다. 범위를 벗어나면 서버에 보내기 전에 `IllegalArgumentException` 이 납니다.

메서드	기본값	범위	설명
`voice(String)`	`F1`	`Voices` 상수, 빈 값 불가	화자. 없는 이름은 서버가 400 `invalid_voice`
`format(AudioFormat)`	`WAV`	`WAV` · `PCM`, `null` 불가	응답 형식
`speed(double)`	1.0	0.67~1.9	말 빠르기
`sampleRate(int)`	32000	16000·22050·24000·32000·44100	출력 샘플레이트(Hz)
`volume(double)`	1.0	0.1~2.0	출력 배율. 넘치는 값은 잘립니다
`normalize(boolean)`	true	–	사용자 사전·숫자 읽기 적용. 이미 한글 읽기면 false
`seed(long)`	지정 안 함	0~4294967295	지정하지 않으면 같은 문장·화자는 늘 같은 소리(권장)
`requestId(String)`	SDK가 만듦	–	거래·요청 번호. 게이트웨이 로그에 그대로 남습니다
`sentencePauseMs(int)`	서버 기본(300)	0~5000	문장 사이 쉼(밀리초)
`dictionary(String)`	기본 사전	게이트웨이에 있는 이름	이름 있는 사전. 기본 사전 위에 얹힙니다
`ssml(boolean)`	자동 판별	–	true = SSML, false = 일반 글. 6.8 참조
`build()`	–	–	요청을 만듭니다

범위 오류 메시지 예:

```
speed 는 0.67~1.9
sampleRate 는 16000·22050·24000·32000·44100 중 하나
volume 은 0.1~2.0
sentencePauseMs 는 0~5000
seed 는 0~4294967295
```

읽기 메서드

`getInput()`, `getVoice()`, `getFormat()`, `getSpeed()`, `getSampleRate()`, `getVolume()`, `isNormalize()`, `getSeed()` (지정 안 하면 `null`), `getRequestId()` (지정 안 하면 `null`), `getSentencePauseMs()` (지정 안 하면 `null`), `getDictionary()`

```
SpeechRequest req = SpeechRequest.builder("리볼빙 결제일은 매월 25일입니다. 감사합니다.")
    .voice(Voices.F2)
    .sampleRate(24000)
    .speed(1.1)
    .volume(0.8)
    .sentencePauseMs(500)
    .dictionary("card")
    .requestId("CALL-7781-02")
    .build();
```

4.4 SpeechStream

`stream()` 이 돌려주는 `InputStream` 입니다. 첫 문장이 합성되는 즉시 읽을 수 있습니다.

- 끝까지 읽고 닫으면 연결을 다시 씁니다(keep-alive).
- 중간에 닫으면 연결을 끊습니다. 게이트웨이는 남은 문장 합성을 버립니다.
- 한 스레드에서만 읽습니다(스레드 안전하지 않음).

메서드	설명
`int read()` · `int read(byte[], int, int)`	일반 `InputStream` 읽기. 받는 중 끊기면 `IOException("음성을 받는 중 끊김 (requestId=..., endpoint=...)")`
`long transferTo(OutputStream out)`	끝까지 `out` 으로 보냅니다. **청크마다 `flush()`** 해 첫 소리가 늦지 않습니다. 보낸 바이트 수를 돌려줍니다
`void close()`	닫습니다. 여러 번 불러도 됩니다
`String getContentType()`	`audio/wav` 또는 `audio/pcm;rate=<샘플레이트>;channels=1;format=s16le`
`String getHeader(String name)`	게이트웨이 응답 헤더 값. 없으면 `null` . 예: `X-Quota-Limit` , `X-License-Warning`
`int getQuotaRemaining()`	하루 한도가 걸린 키(시험 엔드포인트 등)의 오늘 남은 횟수 (`X-Quota-Remaining`). 한도 없는 키면 `-1`
`int getSampleRate()`	응답 헤더 `X-Sample-Rate` 값
`int getSentences()`	게이트웨이가 나눈 문장 수(`X-Sentences`)
`AudioFormat getFormat()`	요청한 형식
`String getEndpoint()`	응답한 게이트웨이 주소
`String getRequestId()`	요청 번호
`long getBytesRead()`	지금까지 읽은 바이트 수

하루 한도와 라이선스 경고를 읽는 예입니다.

```
try (SpeechStream s = tts.stream(SpeechRequest.builder("남은 횟수 확인입니다.").build())) {
    s.transferTo(out);
    System.out.println("남은 횟수=" + s.getQuotaRemaining() + " 한도=" + s.getHeader("X-Quota-
          Limit")
        + " 경고=" + s.getHeader("X-License-Warning"));
}
```

예상 출력: (하루 5회 한도가 걸린 키로 첫 요청)

```
남은 횟수=4 한도=5 경고=null
```

- 한도 없는 키면 `getQuotaRemaining()` 은 `-1` 이고, `X-Quota-Limit` 은 `null` 입니다.
- `X-License-Warning` 은 라이선스 만료 뒤 유예 기간에만 붙습니다(예: `expired 2027-12-31; grace until +14d`).
- 전체 받기(`synthesize()`)는 헤더를 돌려주지 않습니다. 헤더가 필요하면 `stream()` 을 씁니다.

> Java 9 이상의 `InputStream.transferTo()` 와 이름이 같습니다. `SpeechStream` 의 것은 청크마다 `flush()` 하는 점이 다릅니다. Java 8에서도 씁니다.

4.5 AudioFormat

값	설명
`AudioFormat.WAV`	WAV(44바이트 헤더 + PCM). 브라우저 `<audio>` 에서 바로 재생됩니다
`AudioFormat.PCM`	헤더 없는 PCM. **16비트 부호 있는 정수, 리틀 엔디언(s16le), 모노**

둘 다 16비트 모노입니다.

4.6 Voices

상수	값
`Voices.F1` ~ `Voices.F5`	여성 화자 5명 (`"F1"` ~ `"F5"`)
`Voices.M1` ~ `Voices.M5`	남성 화자 5명 (`"M1"` ~ `"M5"`)

실제로 쓸 수 있는 목록은 `tts.voices()` 로 확인합니다. 화자 소리는 온라인 데모 https://copland.ai/tts/demo/ 에서 들어 봅니다.

4.7 NormalizeResult

메서드	설명
`String getNormalized()`	엔진에 들어갈 한글 읽기 전체
`List<String> getSentences()`	게이트웨이가 나눈 문장
`List<String> getUnknownLatin()`	한글로 못 바꾼 영문 단어. 사용자 사전에 넣을 후보
`String toString()`	`getNormalized()` 와 같습니다

4.8 Health

메서드	설명
`String getEndpoint()`	게이트웨이 주소
`boolean isOk()`	요청을 받을 수 있으면 true (`/health` 가 200)
`String getStatus()`	`ok` · `restarting` (합성 워커 재기동 중) · `license_expired` (라이선스 유예 기간도 지남) · `unreachable` (연결 실패)
`String getVersion()`	게이트웨이 버전 (예: `0.3.4`)
`int getWorkers()`	합성 워커 수
`String getError()`	연결 실패 사유. 정상이면 `null`

4.9 TtsException

게이트웨이 호출 실패입니다. `RuntimeException` 이므로 `throws` 에 적지 않아도 됩니다.

메서드	설명
`int getStatus()`	HTTP 상태. 연결 실패·받는 중 끊김은 `0`
`String getCode()`	오류 코드(아래 표). 게이트웨이 응답에 코드가 없으면 `http_<상태>`
`boolean isRetryable()`	다른 서버로 다시 보내도 되는 오류인가. `connection_failed` , 429(`daily_quota` 제외), 502·503·504이면 true
`String getRequestId()`	게이트웨이 접근 로그와 맞춰 볼 요청 번호
`String getEndpoint()`	마지막으로 시도한 게이트웨이 주소
`String getMessage()`	사람이 읽는 사유 + `[status=..., code=..., requestId=..., endpoint=...]`

오류 코드표

code	HTTP	뜻	다시 보냄
`invalid_parameter`	400	요청 값 형식 오류	아니요
`invalid_json`	400	요청 본문이 JSON이 아님 (SDK는 늘 JSON을 보내므로 직접 호출 때만)	아니요
`invalid_voice`	400	없는 화자	아니요
`invalid_sample_rate`	400	지원하지 않는 샘플레이트	아니요
`invalid_model`	400	없는 모델 이름	아니요
`invalid_dictionary`	400	없는 사전 이름	아니요
`invalid_ssml`	400	SSML 문법 오류	아니요
`empty_input`	400	읽을 내용이 없음 (예: `break` 만 있는 SSML)	아니요
`dictionary_error`	400	사전 다시 읽기 실패 (이전 사전 유지)	아니요
`keys_error`	400	키 파일 다시 읽기 실패 (이전 키 유지)	아니요
`invalid_api_key`	401	키 없음·틀림	아니요
`admin_required`	403	관리자 키가 필요한 API (`usage()` · `license()` · `reloadDictionaries()` · `reloadKeys()`)	아니요
`ip_not_allowed`	403	허용하지 않은 접속 주소	아니요
`license_expired`	403	라이선스 만료(유예 기간도 지남)	아니요
`not_found`	404	없는 경로 (주소 뒤 경로 오류)	아니요
`method_not_allowed`	405	허용하지 않은 메서드	아니요
`input_too_long`	413	5000자 초과(SSML 태그 포함)	아니요
`overloaded`	429	합성 대기열이 가득 참 (`Retry-After: 1`)	예
`channel_limit`	429	라이선스 동시 채널을 모두 쓰는 중 (`Retry-After: 1`)	예
`daily_quota`	429	하루 사용 한도 초과 (시험 엔드포인트 등). 한국 자정에 초기화	아니요
`http_502`	502	게이트웨이 앞 프록시가 응답을 못 받음	예
`restarting`	503	합성 워커 재기동 중 (`Retry-After: 5`)	예
`synthesis_timeout`	504	문장 하나 합성 시간이 상한(기본 30초)을 넘음	예
`connection_failed`	0	SDK: 연결 실패(거부·연결 시간 초과·이름 해석 실패), 또는 응답 전에 연결이 끊김	예
`response_timeout`	0	SDK: 연결은 됐는데 응답이 `readTimeoutMillis` 안에 오지 않음	아니요
`stream_broken`	0	SDK: 전체 받기·JSON 응답을 받는 중 끊김	아니요

- "다시 보냄"은 SDK가 다음 서버로 자동으로 다시 보내는지를 뜻합니다(7.1 참조).
- `daily_quota`는 429지만 다시 보내지 않습니다. 다음 날까지 풀리지 않기 때문입니다.
- `response_timeout`은 서버가 아직 합성 중일 수 있어 다시 보내지 않습니다. 다른 서버로 보내면 같은 글을 두 번 합성합니다.
- `SpeechStream`을 읽는 중에 끊기면 `TtsException`이 아니라 `IOException`이 납니다.

```
try {
    tts.synthesize(SpeechRequest.builder("안녕하세요.").voice("X9").build());
} catch (TtsException e) {
    log.warn("TTS 실패 code={} status={} retryable={} rid={}",
        e.getCode(), e.getStatus(), e.isRetryable(), e.getRequestId());
}
```

4.10 Wav

16비트 모노 PCM용 WAV 헤더 도우미입니다.

멤버	설명
`Wav.HEADER_BYTES`	`44`
`static byte[] header(int sampleRate, long pcmBytes)`	PCM 앞에 붙일 44바이트 헤더
`static byte[] fromPcm(byte[] pcm, int sampleRate)`	PCM → WAV 바이트
`static byte[] fixHeader(byte[] wav)`	흘려 받은 WAV(크기 칸 `0xFFFFFFFF`)의 크기 칸을 실제 길이로 고칩니다. 배열을 제자리에서 고치고 그대로 돌려줍니다

```
byte[] pcm = tts.synthesize(SpeechRequest.builder("녹취 시스템으로 넘길 음성입니다.")
    .format(AudioFormat.PCM).sampleRate(16000).build());
byte[] wav = Wav.fromPcm(pcm, 16000);
```

5. 실행 예제로 확인하기

이 장의 예제는 Java 8(Temurin 1.8.0)로 컴파일·실행해 확인했습니다. 출력은 시험 환경의 실제 출력입니다. 게이트웨이 주소(`http://localhost:9300`)와 키(환경 변수 `COPLAND_TTS_KEY`)는 고객사 값으로 바꿉니다.

5.1 흘려 받기와 파일 저장

```java
import ai.copland.tts.*;
import java.io.*;

public class StreamSave {
    public static void main(String[] args) throws Exception {
        CoplandTtsClient tts = CoplandTtsClient.builder()
            .endpoints("http://localhost:9300")
            .apiKey(System.getenv("COPLAND_TTS_KEY"))
            .build();

        SpeechRequest req = SpeechRequest.builder(
                "고객님의 이번 달 청구 금액은 32,500원입니다. 납부 기한은 10월 25일입니다.")
            .voice(Voices.M1)
            .sampleRate(32000)
            .requestId("TX-20261012-0001")
            .build();

        // 1) 흘려 받기: 첫 바이트까지 걸린 시간을 잰다
        long t0 = System.nanoTime();
        ByteArrayOutputStream buf = new ByteArrayOutputStream();
        try (SpeechStream s = tts.stream(req)) {
            int first = s.read();
            long ttfb = (System.nanoTime() - t0) / 1_000_000;
            buf.write(first);
            s.transferTo(buf);
            System.out.println("endpoint=" + s.getEndpoint() + " requestId=" + s.getRequestId()
                + " rate=" + s.getSampleRate() + " sentences=" + s.getSentences()
                + " type=" + s.getContentType() + " bytes=" + s.getBytesRead() + " firstByteMs="
          + ttfb);
        }
        // 2) 흘려 받은 WAV를 파일로: 크기 칸을 고친다
        byte[] wav = Wav.fixHeader(buf.toByteArray());
        try (FileOutputStream f = new FileOutputStream("stream.wav")) {
            f.write(wav);
        }
        // 3) 전체 받기를 바로 파일로
        tts.synthesizeToFile(req, new File("whole.wav"));
        System.out.println("stream.wav " + wav.length + " / whole.wav " + new
          File("whole.wav").length());

        // 4) OutputStream 으로 중계
        try (OutputStream out = new FileOutputStream("relay.wav")) {
            long n = tts.synthesizeTo(req, out);
            System.out.println("relay.wav " + n + "바이트 (스트리밍 헤더)");
        }
    }
}
```

예상 출력:

```
endpoint=http://localhost:9300 requestId=TX-20261012-0001 rate=32000 sentences=2 type=audio/wav
bytes=509296 firstByteMs=550
stream.wav 509296 / whole.wav 509296
relay.wav 509296바이트 (스트리밍 헤더)
```

확인 방법:

- `stream.wav` 와 `whole.wav` 의 크기가 같습니다. `fixHeader` 로 고친 파일은 전체 받기 결과와 같은 WAV입니다.
- `file stream.wav` 가 `RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 32000 Hz` 로 나옵니다.
- 첫 바이트 시간(`firstByteMs`)은 장비마다 다릅니다. 위 값은 AMX가 없는 시험 장비의 값입니다.

5.2 정규화 미리보기

```
NormalizeResult r = tts.normalize("KB국민카드 32,500원. 문의는 1588-1234로 주세요. ZQX 서비스");
System.out.println(r.getNormalized());
System.out.println(r.getSentences());
System.out.println(r.getUnknownLatin());

System.out.println(tts.normalize("리볼빙 신청이 완료되었습니다.", "card"));
System.out.println(tts.normalize("KB국민카드 32,500원, cloud 서비스").getUnknownLatin());
```

예상 출력: (`KB→케이비` 가 고객사 사전에, `리볼빙→일부결제금액이월약정` 이 `card` 사전에 있는 시험 환경)

```
케이비국민카드 삼만 이천오백 원. 문의는 일오팔팔 일이삼사로 주세요. 지큐엑스 서비스
[케이비국민카드 삼만 이천오백 원., 문의는 일오팔팔 일이삼사로 주세요., 지큐엑스 서비스]
[]
일부결제금액이월약정 신청이 완료되었습니다.
[cloud]
```

- 대문자 약어(`ZQX`)는 알파벳 이름으로 읽습니다.
- 소문자 영단어(`cloud`)는 읽기를 정하지 못해 `getUnknownLatin()` 에 나옵니다. 사용자 사전에 넣을 후보입니다.

5.3 오류 처리

```
static void tryIt(String what, Runnable r) {
    try {
        r.run();
        System.out.println(what + " → 성공");
    } catch (TtsException e) {
        System.out.println(what + " → status=" + e.getStatus() + " code=" + e.getCode()
            + " retryable=" + e.isRetryable());
    } catch (IllegalArgumentException | IllegalStateException e) {
        System.out.println(what + " → " + e.getClass().getSimpleName() + ": " + e.getMessage());
    }
}

// noKey = 키 없이 만든 클라이언트, tts = 일반 키, admin = 관리자 키
tryIt("키 없음", () -> noKey.synthesize("안녕하세요."));
tryIt("화자 틀림", () -> tts.synthesize(SpeechRequest.builder("안녕하세요.").voice("X9").build()));
tryIt("사전 이름 틀림", () -> tts.synthesize(SpeechRequest.builder("안녕하세
          요.").dictionary("nope").build()));
tryIt("SSML 틀림", () -> tts.synthesize(SpeechRequest.builder("<speak>안녕<break time=\"x\"/></
          speak>").build()));
tryIt("읽을 내용 없음", () -> tts.synthesize(SpeechRequest.builder("<speak><break time=\"1s\"/></
          speak>").build()));
tryIt("일반 키로 사전 다시 읽기", () -> tts.reloadDictionaries());
tryIt("관리자 키로 사전 다시 읽기", () -> admin.reloadDictionaries());
tryIt("speed 범위", () -> SpeechRequest.builder("a").speed(2.5));
tryIt("주소 형식", () -> CoplandTtsClient.builder().endpoints("localhost:9300"));
```

예상 출력:

```
키 없음 → status=401 code=invalid_api_key retryable=false
화자 틀림 → status=400 code=invalid_voice retryable=false
사전 이름 틀림 → status=400 code=invalid_dictionary retryable=false
SSML 틀림 → status=400 code=invalid_ssml retryable=false
읽을 내용 없음 → status=400 code=empty_input retryable=false
일반 키로 사전 다시 읽기 → status=403 code=admin_required retryable=false
관리자 키로 사전 다시 읽기 → 성공
speed 범위 → IllegalArgumentException: speed 는 0.67~1.9
주소 형식 → IllegalArgumentException: http:// 또는 https:// 로 시작해야 함: localhost:9300
```

`e.getMessage()` 에는 서버가 준 사유가 들어 있습니다. 예:

```
voice 는 ['F1', 'F2', 'F3', 'F4', 'F5', 'M1', 'M2', 'M3', 'M4', 'M5'] 중 하나 [status=400,
code=invalid_voice, requestId=98cd4dfb824d4371, endpoint=http://localhost:9300]
break time 형식: x (예: 500ms, 1s) [status=400, code=invalid_ssml, requestId=367f0dfda5664b40,
endpoint=http://localhost:9300]
```

6. 사용 패턴

6.1 전체 받기와 흘려 받기 고르기

용도	쓸 메서드	이유
상담 화면에서 바로 재생	`stream()` · `synthesizeTo()`	첫 문장만 합성되면 소리가 나갑니다
파일 저장·메일 첨부·DB 보관	`synthesize()` · `synthesizeToFile()`	WAV 헤더 길이가 정확합니다
녹취 장비로 PCM 넘기기	`stream()` + `AudioFormat.PCM`	헤더 없이 바로 넣습니다
여러 문장을 미리 만들어 두기	`synthesizeAsync()`	요청 스레드를 막지 않습니다

긴 안내문(수십 문장)을 `synthesize()` 로 받으면 마지막 문장까지 합성한 뒤에야 응답이 옵니다. 재생용이면 `stream()` 을 씁니다.

6.2 서블릿 중계 (Java 8, javax.servlet)

상담 화면이 `<audio src="/tts?text=...">` 로 재생하는 구조입니다.

```java
public class TtsServlet extends HttpServlet {
    static final CoplandTtsClient TTS = CoplandTtsClient.builder()
        .endpoints(System.getProperty("copland.tts.urls").split(","))
        .apiKey(System.getProperty("copland.tts.key"))
        .userAgent("counter-web")
        .build();

    @Override
    protected void doGet(HttpServletRequest rq, HttpServletResponse rs) throws IOException {
        rq.setCharacterEncoding("UTF-8");
        String text = rq.getParameter("text");
        if (text == null || text.trim().isEmpty()) {
            rs.sendError(400, "text 가 비어 있음");
            return;
        }
        SpeechRequest req = SpeechRequest.builder(text).voice(Voices.F1).build();
        try (SpeechStream s = TTS.stream(req)) {          // 실패(TtsException)는 여기서 난다
            rs.setContentType(s.getContentType());
            rs.setHeader("Cache-Control", "no-store");
            s.transferTo(rs.getOutputStream());          // 청크마다 flush → 첫 소리가 늦지 않다
        } catch (TtsException e) {
            if (!rs.isCommitted()) {
                rs.sendError(e.isRetryable() ? 503 : 502, e.getCode() + " " + e.getRequestId());
            }
        }
    }
}
```

- `TTS.stream()` 이 돌아오기 전의 실패는 아직 응답을 보내지 않았으므로 HTTP 오류로 바꿀 수 있습니다.
- 브라우저가 재생 중에 창을 닫으면 `transferTo` 가 `IOException` 으로 끝납니다. try-with-resources가 스트림을 닫아 게이트웨이 연결을 끊고, 게이트웨이는 남은 문장 합성을 버립니다.
- GET 쿼리 문자열의 `text` 는 WAS의 URI 인코딩 설정(예: Tomcat `URIEncoding="UTF-8"`)을 따릅니다. 한글이 깨지면 그 설정을 확인합니다.

6.3 Spring MVC 중계 (StreamingResponseBody)

```
@Configuration
class TtsConfig {
    @Bean
    CoplandTtsClient coplandTts(@Value("${copland.tts.urls}") String[] urls,   // 쉼표로 여러 대
                                @Value("${copland.tts.key}") String key) {
        return CoplandTtsClient.builder().endpoints(urls).apiKey(key).userAgent("counter-
          web").build();
    }
}

@RestController
public class TtsController {
    private final CoplandTtsClient tts;

    public TtsController(CoplandTtsClient tts) {
        this.tts = tts;
    }

    @GetMapping(value = "/tts", produces = "audio/wav")
    public ResponseEntity<StreamingResponseBody> speak(@RequestParam String text,
                                                       @RequestParam(defaultValue = "F1") String
          voice,
                                                       @RequestHeader(value = "X-Tx-Id", required
          = false) String txId) {
        SpeechRequest req = SpeechRequest.builder(text).voice(voice).requestId(txId).build();
        SpeechStream s = tts.stream(req);   // 실패(TtsException)는 여기서 나므로 아직 HTTP 오류로 돌
          려줄 수 있다
        StreamingResponseBody body = out -> {
            try (SpeechStream in = s) {
                in.transferTo(out);
            }
        };
        return ResponseEntity.ok()
            .contentType(MediaType.parseMediaType(s.getContentType()))
            .header("Cache-Control", "no-store")
            .body(body);
    }

    @ExceptionHandler(TtsException.class)
    public ResponseEntity<String> ttsFailed(TtsException e) {
        int status = e.isRetryable() ? 503 : 502;
        return ResponseEntity.status(status)
            .contentType(MediaType.TEXT_PLAIN)
            .body("음성 합성 실패 (" + e.getCode() + ", " + e.getRequestId() + ")");
    }

    @ExceptionHandler(IllegalArgumentException.class)
    public ResponseEntity<String> badInput(IllegalArgumentException e) {
        return ResponseEntity.badRequest().contentType(MediaType.TEXT_PLAIN).body(e.getMessage());
```

```
    }
}
```

`application.yml` 예:

```yaml
copland:
  tts:
    urls: http://10.0.0.11:9300,http://10.0.0.12:9300
    key: ${COPLAND_TTS_KEY}
spring:
  mvc:
    async:
      request-timeout: 600000   # 긴 안내문을 흘려 보내는 동안 비동기 요청이 끊기지 않게 (밀리초)
```

- `StreamingResponseBody` 는 Spring의 비동기 요청으로 돌아갑니다. 비동기 요청 시간 제한이 짧으면 긴 안내문 재생 도중 끊깁니다. `spring.mvc.async.request-timeout` 을 안내문 길이보다 넉넉히 줍니다.
- `requestId(null)` 은 "지정 안 함"과 같습니다. 이때 SDK가 요청 번호를 만듭니다.

6.4 파일 저장

```java
SpeechRequest req = SpeechRequest.builder("고객님의 이번 달 청구 금액은 32,500원입니다.").build();
tts.synthesizeToFile(req, new File("/data/notice/2026-10/bill-0001.wav"));
```

흘려 받은 결과를 파일로 남기려면 `Wav.fixHeader()` 로 크기 칸을 고칩니다(5.1 참조). 고치지 않은 파일도 대부분의 플레이어가 재생하지만, 일부 편집 도구는 길이를 잘못 읽습니다.

6.5 비동기 (CompletableFuture + 전용 Executor)

```
// 애플리케이션에 하나. 공용 ForkJoinPool 대신 전용 스레드 풀을 쓴다 (네트워크 대기 작업이므로)
ExecutorService ttsPool = Executors.newFixedThreadPool(8);

List<CompletableFuture<byte[]>> jobs = new ArrayList<>();
for (String text : Arrays.asList("첫 번째 안내입니다.", "두 번째 안내입니다.", "세 번째 안내입니
        다.")) {
    jobs.add(tts.synthesizeAsync(SpeechRequest.builder(text).build(), ttsPool));
}
for (CompletableFuture<byte[]> f : jobs) {
    System.out.println(f.get(30, TimeUnit.SECONDS).length + "바이트");
}

CompletableFuture<byte[]> bad = tts.synthesizeAsync(SpeechRequest.builder("안
        녕").voice("X9").build(), ttsPool);
try {
    bad.join();
} catch (CompletionException e) {
    TtsException te = (TtsException) e.getCause();
    System.out.println("실패: " + te.getCode());
}
```

예상 출력:

```
137606바이트
132008바이트
129868바이트
실패: invalid_voice
```

- 풀 크기는 이 WAS가 동시에 보낼 합성 요청 수에 맞춥니다. 게이트웨이 라이선스 동시 채널을 넘게 보내면 429 `channel_limit` 가 납니다.
- WAS를 내릴 때 `ttsPool.shutdown()` 을 부릅니다.

6.6 브라우저 `<audio>` 연동

6.2·6.3의 `/tts` 주소를 `<audio>` 의 `src` 로 씁니다.

```
<audio id="tts" controls preload="none"></audio>
<script>
  function speak(text, voice) {
    var a = document.getElementById('tts');
    a.src = '/tts?voice=' + (voice || 'F1') + '&text=' + encodeURIComponent(text);
    a.play();
  }
</script>
<button onclick="speak('고객님, 안녕하세요. 상담원 연결 전 안내 말씀 드립니다.')">안내 듣기</button>
```

- 흘려 받는 WAV는 크기 칸이 `0xFFFFFFFF` 입니다. 주요 브라우저는 그대로 재생합니다. 상담 PC에서 쓰는 브라우저로 한 번 확인합니다.
- 긴 안내문은 URL 길이 제한에 걸릴 수 있습니다. 이때는 안내문을 WAS에 두고 번호(`/tts?id=...`)로 부르게 만듭니다.
- 응답에 `Cache-Control: no-store` 를 붙여, 고객 정보가 든 음성이 브라우저 캐시에 남지 않게 합니다.
- 브라우저 스크립트에서 게이트웨이나 시험 엔드포인트를 직접 부를 수 없습니다(CORS를 열지 않습니다). 반드시 자사 WAS를 거칩니다. API 키도 브라우저에 두지 않습니다.

6.7 녹취 시스템에 PCM 넘기기

녹취 장비·믹서가 헤더 없는 PCM을 받는 경우입니다.

```
SpeechRequest req = SpeechRequest.builder("녹취 시스템으로 넘길 음성입니다.")
    .format(AudioFormat.PCM)
    .sampleRate(16000)
    .build();
try (SpeechStream s = tts.stream(req)) {
    System.out.println("type=" + s.getContentType());   // audio/
          pcm;rate=16000;channels=1;format=s16le
    recorder.write(s);                                    // 고객사 녹취 연동부 (예시)
}
```

- PCM 형식은 **16비트 부호 있는 정수, 리틀 엔디언(s16le), 모노**입니다. 바이트 수 ÷ 2 ÷ 샘플레이트 = 초입니다.
- 응답 Content-Type은 `audio/pcm;rate=16000;channels=1;format=s16le` 입니다. 받는 쪽 설정을 s16le로 맞춥니다.
- 녹취 장비가 WAV를 원하면 `Wav.fromPcm(pcm, 16000)` 으로 감쌉니다.

실행 결과 예:

```
pcm 95362바이트 = 2.9800625초, wav 95406
pcm stream type=audio/pcm;rate=16000;channels=1;format=s16le
```

6.8 SSML

글이 `<speak>` 로 시작하면 SSML로 읽습니다.

```
tts.synthesize("<speak>고객님, 안녕하세요.<break time=\"800ms\"/>"
    + "계좌번호는 <say-as interpret-as=\"digits\">110123456789</say-as>입니다. "
    + "<prosody rate=\"slow\">중요한 내용이니 천천히 안내해 드리겠습니다.</prosody></speak>");
```

태그	지원
`break`	`time="500ms"` · `time="1s"` (최대 5초), `strength="none"` ~ `"x-strong"`
`say-as interpret-as=`	`digits` · `telephone` (한 자리씩, 0 = 공), `characters` · `spell-out`, `cardinal` · `number`, `date`
`sub alias="읽기"`	표기를 읽기로 바꿉니다
`prosody`	`rate` (slow·fast·80%·+10%), `volume` (soft·loud·+6dB·50%)
`p`, `s`	문장 경계
그 밖의 태그	태그는 무시하고 내용만 읽습니다. `audio` 는 내용째 뺍니다

`.ssml(boolean)` 의 쓰임은 다음과 같습니다.

- 지정하지 않으면 글이 `<speak` 로 시작하는지로 판별합니다.
- `.ssml(true)` 면 `<speak>` 로 시작하지 않는 글도 SSML로 읽습니다. 게이트웨이가 `<speak>` 로 감쌉니다.
- `.ssml(false)` 는 `<speak>` 로 시작하는 글을 SSML로 해석하지 않고 일반 글로 읽게 합니다.

미리보기로 확인할 수 있습니다.

```
System.out.println(tts.normalize(
    "<speak>계좌번호는 <say-as interpret-as=\"digits\">110123456789</say-as>입니다.</speak>"));
```

```
계좌번호는 일일공일이삼사오육칠팔구입니다.
```

`say-as` · `sub` 바로 뒤의 조사는 원문 띄어쓰기대로 붙여 읽습니다.

```
byte[] wav = tts.synthesize(SpeechRequest.builder("고객님, <break time=\"800ms\"/> 안녕하세
        요.").ssml(true).build());
```

6.9 사전 선택

- 게이트웨이의 기본 사전(고객사 사전 + 기본 규칙)은 모든 요청에 적용됩니다.
- 업무별로 다른 읽기가 필요하면 게이트웨이에 이름 있는 사전을 두고 요청마다 고릅니다.

```
SpeechRequest req = SpeechRequest.builder("리볼빙 결제일은 매월 25일입니다.")
    .dictionary("card")        // 게이트웨이 사전 폴더의 card.tsv
    .build();
```

- 없는 이름이면 400 `invalid_dictionary` 이고, 메시지에 쓸 수 있는 이름이 나옵니다.
- 사전 파일을 바꾼 뒤에는 관리자 키로 `reloadDictionaries()` 를 부릅니다. 모든 게이트웨이에 보냅니다.
- 이미 한글 읽기로 만든 글이면 `.normalize(false)` 로 사전·숫자 규칙을 끕니다.

6.10 요청 번호 연계

```
SpeechRequest req = SpeechRequest.builder(text)
    .requestId("CALL-7781-02")      // 우리 거래·상담 번호
    .build();
```

- SDK는 요청 번호를 `X-Request-Id` 헤더로 보냅니다. 게이트웨이 접근 로그에 그대로 남습니다.

```
... gateway.access: rid=TX-20261012-0001 key=web01 ip=127.0.0.1 speech voice=M1 chars=47
sents=2 audio=8.0s ttfb=0ms total=0ms ok
```

- 지정하지 않으면 SDK가 16자리 번호를 만듭니다. `SpeechStream.getRequestId()` · `TtsException.getRequestId()` 로 꺼내 WAS 로그에 함께 남깁니다.
- 장애 전환으로 다른 서버에 다시 보내도 요청 번호는 같습니다. 로그를 서버 여러 대에서 찾을 때 씁니다.

7. 운영 설정

7.1 여러 서버와 장애 전환

```
CoplandTtsClient tts = CoplandTtsClient.builder()
    .endpoints("http://10.0.0.11:9300", "http://10.0.0.12:9300")
    .apiKey(System.getenv("COPLAND_TTS_KEY"))
    .build();
```

동작 규칙은 다음과 같습니다.

1. **번갈아 보내기:** 요청마다 시작 서버를 하나씩 바꿉니다(라운드 로빈).
2. **다시 보내는 오류:** 연결 실패, 응답 전에 연결이 끊김, HTTP 429(`daily_quota` 제외)·502·503·504를 받으면 다음 서버로 다시 보냅니다.
3. **다시 보내지 않는 오류:** 400·401·403·404·413과 `daily_quota` 는 바로 `TtsException` 을 던집니다. 다른 서버로 보내도 결과가 같기 때문입니다.
   - **응답 시간 초과(`response_timeout`)도 다시 보내지 않습니다.** 연결은 됐으므로 서버가 아직 합성 중일 수 있습니다. 다시 보내면 같은 글을 두 번 합성합니다. 그 서버를 쉬게 하지도 않습니다.
4. **쉬는 시간:** 연결 단계에서 실패한 서버는 `failoverCooldownMillis` (기본 5초) 동안 건너뜁니다. 쉬는 시간이 지나면 다시 차례에 넣습니다. 모든 서버가 쉬는 중이면 차례대로 시도합니다.
5. **기다렸다 보내기:** 429·503을 받았고 아직 안 가 본 서버가 없으면, 0.2초·0.4초… (최대 1초) 기다린 뒤 다시 보냅니다.
6. **최대 시도:** `maxAttempts` (기본 max(2, 서버 수))번 시도한 뒤 마지막 오류를 던집니다. 서버가 한 대면 한 번 더 시도합니다.
7. **소리를 받기 시작한 뒤에는 다시 보내지 않습니다.** 게이트웨이가 응답 헤더를 보낸 뒤의 끊김은 다시 보내지 않습니다. 같은 안내가 두 번 재생되는 것을 막기 위해서입니다.

실행 예: 꺼진 서버(`9399`)와 살아 있는 서버(`9300`)를 함께 등록했습니다.

```
CoplandTtsClient tts = CoplandTtsClient.builder()
    .endpoints("http://localhost:9399", "http://localhost:9300")  // 9399 = 꺼진 서버
    .apiKey(System.getenv("COPLAND_TTS_KEY"))
    .build();
for (Health h : tts.health()) {
    System.out.println(h);
}
for (int i = 0; i < 4; i++) {
    long t0 = System.nanoTime();
    try (SpeechStream s = tts.stream(SpeechRequest.builder("장애 전환 시험 " + (i + 1) + "번입니
          다.").build())) {
        s.transferTo(new java.io.ByteArrayOutputStream());
        System.out.println((i + 1) + "번 ← " + s.getEndpoint() + " " + (System.nanoTime() -
          t0) / 1_000_000 + "ms");
    }
}
```

예상 출력:

```
http://localhost:9399 unreachable (Connection refused (Connection refused))
http://localhost:9300 ok v0.3.4 workers=2
Oct 12, 2026 2:22:01 PM ai.copland.tts.CoplandTtsClient call
WARNING: TTS 게이트웨이 연결 실패 http://localhost:9399: java.net.ConnectException: Connection
refused (Connection refused)
1번 ← http://localhost:9300 331ms
2번 ← http://localhost:9300 314ms
3번 ← http://localhost:9300 372ms
4번 ← http://localhost:9300 326ms
```

- 1번 요청은 꺼진 서버로 먼저 갔다가 다음 서버로 넘어갔습니다. WARNING 로그가 한 번 남습니다.
- 2~4번은 꺼진 서버가 쉬는 중이라 바로 살아 있는 서버로 갑니다.

모든 서버가 꺼져 있으면 다음과 같이 끝납니다.

```
status=0 code=connection_failed retryable=true
연결 실패: Connection refused (Connection refused) [status=0, code=connection_failed,
requestId=b340c4f2ae084fcd, endpoint=http://localhost:9399]
```

게이트웨이 앞에 L4 스위치·로드밸런서가 있으면 그 주소 하나만 넣어도 됩니다. 이때 로드밸런서의 상태 확인은 게이트웨이 `/health`를 씁니다. 준비되면 200이고, 워커 재기동 중이거나 라이선스가 유예 기간까지 지나면 503입니다.

7.2 시간 제한 정하기

설정	기본	정하는 법
`connectTimeoutMillis`	2000	같은 데이터센터면 1000~2000. 짧을수록 죽은 서버를 빨리 건너뜁니다
`readTimeoutMillis`	30000	아래 설명대로 정합니다
`failoverCooldownMillis`	5000	서버 재기동에 걸리는 시간 정도. 보통 그대로 둡니다

`readTimeoutMillis`는 **받는 바이트 사이의 최대 간격**입니다. 호출 방식에 따라 뜻이 다릅니다.

- `stream()` : 첫 문장 합성 시간, 그리고 문장과 문장 사이 간격의 상한입니다. 문장 하나는 보통 1초 안에 합성되므로 기본 30초로 충분합니다.
- `synthesize()` : 게이트웨이가 **모든 문장을 합성할 때까지** 응답이 오지 않습니다. 긴 글이면 30초를 넘을 수 있습니다.
  - 넘으면 `response_timeout`으로 끝납니다. 다른 서버로 다시 보내지 않습니다.
  - 그래서 긴 글은 `stream()`으로 받습니다. `synthesize()`를 꼭 써야 하면, 실제 장비에서 가장 긴 안내문의 전체 받기 시간을 재고 그보다 넉넉하게 `readTimeoutMillis`를 늘립니다.

실행 예: `readTimeoutMillis(800)`으로 8문장을 전체 받기 했습니다. 서버 두 대를 등록했지만 다시 보내지 않았고, 바로 다음 요청은 두 서버에 번갈아 갔습니다.

```
status=0 code=response_timeout retryable=false 850ms
응답 시간 초과 (readTimeoutMillis=800) — 긴 글을 한 번에 받는다면 stream() 을 쓰거나
readTimeoutMillis 를 늘린다 [status=0, code=response_timeout, requestId=7888b0f1faaa4364,
endpoint=http://localhost:9300]
다음 요청 ← http://127.0.0.1:9300
다음 요청 ← http://localhost:9300
```

게이트웨이는 문장 하나가 `TTS_SYNTH_TIMEOUT`(기본 30초)을 넘으면 504 `synthesis_timeout`을 돌려줍니다. `readTimeoutMillis`를 이 값보다 짧게 잡으면 504 대신 `response_timeout`으로 보입니다.

7.3 프록시

- 기본은 **프록시를 쓰지 않습니다**(`Proxy.NO_PROXY`). 게이트웨이는 보통 내부망에 있기 때문입니다. WAS에 `-Dhttp.proxyHost`가 걸려 있어도 무시합니다.
- 프록시를 거쳐야 하면 직접 지정합니다.

```
CoplandTtsClient tts = CoplandTtsClient.builder()
    .endpoints("https://copland.ai/tts/sandbox")
    .proxy(new Proxy(Proxy.Type.HTTP, new InetSocketAddress("proxy.example.local", 8080)))
    .build();
```

- JVM 설정(`-Dhttps.proxyHost` , `-Dhttp.nonProxyHosts` 등)을 따르게 하려면 `.proxy(null)` 을 줍니다.

7.4 HTTPS (사설 인증서)

게이트웨이 앞에 HTTPS 프록시를 두고 사내 사설 인증서를 쓰는 경우입니다.

1. 사설 루트 인증서를 신뢰 저장소로 만듭니다.

```
keytool -importcert -noprompt -alias corp-root -file corp-root-ca.crt \
  -keystore /app/conf/tts-truststore.jks -storepass changeit
```

2. 그 저장소로 `SSLSocketFactory` 를 만들어 넘깁니다.

```
KeyStore ts = KeyStore.getInstance("JKS");
try (InputStream in = new FileInputStream("/app/conf/tts-truststore.jks")) {
    ts.load(in, "changeit".toCharArray());
}
TrustManagerFactory tmf =
          TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
tmf.init(ts);
SSLContext ssl = SSLContext.getInstance("TLS");
ssl.init(null, tmf.getTrustManagers(), null);

CoplandTtsClient tts = CoplandTtsClient.builder()
    .endpoints("https://tts.example.local")
    .apiKey(System.getenv("COPLAND_TTS_KEY"))
    .sslSocketFactory(ssl.getSocketFactory())
    .build();
```

- 인증서의 이름과 주소가 다르면 `.hostnameVerifier(...)` 로 확인 방식을 바꿀 수 있습니다. 모든 이름을 통과시키는 확인기는 쓰지 않습니다.
- JVM 전체 신뢰 저장소(`-Djavax.net.ssl.trustStore`)에 넣었다면 SDK 설정은 필요 없습니다.

7.5 로그

SDK는 `java.util.logging` 의 로거 `ai.copland.tts` 를 씁니다.

수준	남는 때
`WARNING`	게이트웨이 연결 실패, 응답 전에 연결이 끊김 (주소와 예외)
`FINE`	HTTP 오류(429·502·503·504)로 다음 서버로 넘어갈 때

`java.util.logging`으로 FINE까지 보려면 다음처럼 설정합니다.

```
# logging.properties (-Djava.util.logging.config.file=... 로 지정)
handlers = java.util.logging.ConsoleHandler
java.util.logging.ConsoleHandler.level = FINE
ai.copland.tts.level = FINE
```

SLF4J(Logback·Log4j2)를 쓰는 WAS는 `jul-to-slf4j` 브리지로 받습니다.

```
// 애플리케이션 시작 때 한 번
SLF4JBridgeHandler.removeHandlersForRootLogger();
SLF4JBridgeHandler.install();
```

```
<!-- logback.xml -->
<contextListener class="ch.qos.logback.classic.jul.LevelChangePropagator">
  <resetJUL>true</resetJUL>
</contextListener>
<logger name="ai.copland.tts" level="DEBUG"/>
```

- Spring Boot(기본 Logback)는 이 브리지를 스스로 겁니다. `logging.level.ai.copland.tts=DEBUG` 만 주면 됩니다.
- `FINE`은 SLF4J의 `DEBUG`로 옮겨집니다.
- WAS 로그에는 `TtsException.getRequestId()`를 함께 남깁니다. 게이트웨이 로그와 맞춰 볼 때 씁니다.

7.6 스레드 안전과 싱글턴

- `CoplandTtsClient`는 스레드 안전합니다. **애플리케이션에 하나**만 만들어 공유합니다.
  - 요청마다 새로 만들면 장애 전환 상태(쉬는 서버)와 번갈아 보내기 순서가 공유되지 않습니다.
- `SpeechRequest`는 불변이라 공유해도 됩니다.
- `SpeechStream`은 한 스레드에서 읽고 닫습니다.
- 연결은 JDK `HttpURLConnection`의 keep-alive를 씁니다. 끝까지 읽고 닫은 연결은 다시 씁니다.
  - JDK가 서버마다 남겨 두는 유휴 연결 수는 기본 5개입니다(`-Dhttp.maxConnections`). 동시 요청이 많은 WAS는 늘려 두면 연결 재수립이 줄어듭니다.

8. 문제 해결

증상	원인	조치
401 `invalid_api_key`	키를 안 넣었거나 틀림, 게이트웨이 키 파일에서 지워짐	키 값을 확인합니다. 앞뒤 공백은 SDK가 지웁니다. 게이트웨이 관리자에게 키가 살아 있는지 묻습니다
403 `admin_required`	일반 키로 `usage()` · `license()` · `reloadDictionaries()` · `reloadKeys()` 호출	관리자 키로 만든 클라이언트를 따로 씁니다. 관리자 키는 웹 서버에 두지 말고 운영 도구에서만 씁니다
403 `ip_not_allowed`	게이트웨이 허용 대역에 WAS 주소가 없음	게이트웨이 관리자에게 WAS 주소 등록을 요청합니다. 메시지에 막힌 주소가 나옵니다
403 `license_expired`	게이트웨이 라이선스 만료 + 유예 기간 지남	게이트웨이 관리자에게 알립니다. `tts.health()` 의 상태가 `license_expired` 로 보입니다
429 `channel_limit`	라이선스 동시 채널을 모두 쓰는 중	SDK가 다른 서버로 다시 보냅니다. 자주 나면 동시 요청 수를 줄이거나 채널 증설을 검토합니다
429 `overloaded`	합성 대기열이 가득 참(첫 소리가 1초 넘게 밀릴 상황)	SDK가 다른 서버로 다시 보냅니다. 자주 나면 서버 증설을 검토합니다
429 `daily_quota`	시험 엔드포인트의 하루 100회 초과	한국 자정 뒤 다시 씁니다. 남은 횟수는 `SpeechStream.getQuotaRemaining()` 으로 봅니다. 자사 게이트웨이로 옮깁니다
503 `restarting`	합성 워커 재기동 중	잠시 뒤 자동 복구됩니다. 다른 서버가 있으면 SDK가 그쪽으로 보냅니다
504 `synthesis_timeout`	문장 하나 합성이 30초를 넘음	서버 과부하입니다. 게이트웨이 관리자에게 알립니다
`connection_failed`	주소·포트 오류, 방화벽, 서버 중지, 응답 전 연결 끊김	`tts.health()` 나 `java -jar ... --health` 로 서버별 상태를 봅니다
`response_timeout`	응답이 `readTimeoutMillis` 안에 오지 않음. 주로 긴 글을 `synthesize()` 로 받을 때	`stream()` 으로 바꾸거나 `readTimeoutMillis` 를 늘립니다(7.2)
`IOException: 음성을 받는 중 끊김`	흘려 받는 중 네트워크가 끊기거나 서버가 멈춤	다시 보내지 않습니다(중복 재생 방지). 필요하면 화면에서 "다시 듣기"로 새 요청을 보냅니다
`SSLHandshakeException: PKIX path building failed`	신뢰하지 않는 인증서	7.4대로 신뢰 저장소를 지정합니다
브라우저에서 재생이 늦게 시작	중간 프록시·WAS 필터가 응답을 모아서 보냄	압축(gzip) 필터·버퍼링을 `/tts` 에서 끕니다. `transferTo` 가 청크마다 flush합니다

8.1 Java 8에서 copland.ai 인증서 오류

시험 엔드포인트(`https://copland.ai/tts/sandbox`)를 오래된 Java 8에서 부르면 다음 오류가 날 수 있습니다.

```
javax.net.ssl.SSLHandshakeException: ... PKIX path building failed ... unable to find valid
certification path to requested target
```

- 원인: Java 8 초기 빌드(8u101 이전)는 copland.ai 인증서의 루트(ISRG Root X1)를 모릅니다.
- 조치: JDK를 최신 8 업데이트로 올립니다. Temurin 1.8.0_504에서는 정상 동작을 확인했습니다.
- JDK를 올릴 수 없으면 ISRG Root X1 인증서를 신뢰 저장소에 넣고 7.4처럼 지정합니다.

8.2 흘려 받는 중 끊김

- `stream()` 이 돌아온 뒤의 끊김은 `TtsException` 이 아니라 `SpeechStream.read()` 의 `IOException` 입니다.
- 서블릿에서는 이미 응답 헤더를 보냈으므로 HTTP 오류로 바꿀 수 없습니다. 연결을 닫아 브라우저에 끊김을 알립니다.
- 받은 바이트 수(`getBytesRead()`)와 요청 번호를 로그에 남기고, 게이트웨이 로그에서 같은 `rid` 를 찾습니다. 끊긴 요청은 게이트웨이 로그 끝에 `aborted` 또는 `error` 로 남습니다.

9. 명령줄 도구 (Cli)

jar에 들어 있는 설치 확인용 도구입니다. 게이트웨이 설치 직후 WAS 장비에서 연결을 확인할 때 씁니다.

```
java -jar copland-tts-client-0.2.1.jar [인자] "읽을 문장"
```

인자	기본	설명
`--url <주소[,주소…]>`	환경 변수 `COPLAND_TTS_URL`	게이트웨이 주소. 쉼표로 여러 대 (필수)
`--key <키>`	환경 변수 `COPLAND_TTS_KEY`	API 키
`--voice <화자>`	`F1`	화자
`--rate <Hz>`	`32000`	샘플레이트
`--out <파일>`	`out.wav`	저장할 파일
`--stream`	끔	흘려 받고, 첫 소리 지연과 전체 시간을 출력합니다
`--normalize`	끔	합성하지 않고 정규화 미리보기만 출력합니다
`--health`	끔	모든 서버의 상태를 출력합니다
`--help` , `-h`	–	사용법을 출력합니다

종료 코드는 다음과 같습니다.

- `0` : 성공. `--health` 면 모든 서버가 정상
- `1` : 합성 실패, 또는 `--health` 에서 비정상 서버가 하나 이상
- `2` : `--url` 이나 문장이 없음, 또는 인자 값이 빠짐

키를 명령 기록에 남기지 않으려면 환경 변수로 넘깁니다.

```
export COPLAND_TTS_URL=http://10.0.0.11:9300,http://10.0.0.12:9300
export COPLAND_TTS_KEY=ctts_xxxxxxxxxxxxxxxxxxxxxxxxxx
```

9.1 상태 확인

```
java -jar copland-tts-client-0.2.1.jar --url http://localhost:9300,http://localhost:9399 --health
```

예상 출력: (`9399` 는 꺼진 서버, 종료 코드 1)

```
http://localhost:9300 ok v0.3.4 workers=2
http://localhost:9399 unreachable (Connection refused (Connection refused))
```

9.2 정규화 미리보기

```
java -jar copland-tts-client-0.2.1.jar --url http://localhost:9300 --key $COPLAND_TTS_KEY \
  --normalize "KB국민카드 32,500원, cloud 서비스"
```

예상 출력:

```
케이비국민카드 삼만 이천오백 원, cloud 서비스
사전 후보(영문): [cloud]
```

9.3 합성

```
java -jar copland-tts-client-0.2.1.jar --url http://localhost:9300 --key $COPLAND_TTS_KEY \
  --stream --voice M1 --rate 16000 --out cli.wav "안녕하세요. 상담원 연결 전 안내 말씀 드립니다."
```

예상 출력:

```
cli.wav ← http://localhost:9300 첫 소리 268ms, 전체 687ms, 169472바이트, 문장 2,
requestId=70095c0c5b29495a
```

`--stream` 없이 실행하면 전체 받기로 저장합니다.

```
cli2.wav 51ms 96998바이트
```

키가 없거나 틀리면 다음처럼 끝납니다(종료 코드 1).

```
실패: API 키가 없거나 맞지 않음 [status=401, code=invalid_api_key, requestId=0476cbcf19014078,
endpoint=http://localhost:9300]
```

- 명령줄 도구는 사전 선택·SSML 강제·PCM 형식을 지원하지 않습니다. 그런 확인은 SDK 코드로 합니다. 값을 받는 인자 뒤에 값을 빠뜨리면 사용법과 함께 끝납니다(종료 코드 2).

```
java -jar copland-tts-client-0.2.1.jar --url http://localhost:9300 --key
```

```
인자 값이 빠졌다: --key
사용: java -jar copland-tts-client.jar --url http://host:9300[,http://host2:9300] [--key KEY]
...
```

10. 관련 문서

- 게이트웨이 설치: 설치 매뉴얼
- 게이트웨이 운영(API 키·사전·로그): 운영 매뉴얼
- HTTP API(다른 언어·curl): API 개발자 매뉴얼
- 라이선스 발급: 라이선스 발급 포털 사용 안내
- 온라인 데모: https://copland.ai/tts/demo/
- 시험 엔드포인트: https://copland.ai/tts/sandbox
- 문서: https://copland.ai/tts/docs/
- 서드파티 라이선스: 패키지의 `NOTICE.md`

문의: COPLAND 기술지원 (연락처는 계약서 참조)
