



문서 기준일 2026-10-28

API 명세서

(주)플로터랩



목 차

시작하기	3
인증	3
요청 형식	4
비동기 작업(작업 ID 폴링)	5
오류 응답	6
처리 시간·재시도	7
데이터 처리와 보관	8
버전과 상태 확인	8
발전 분석시트 입력	10
연동 순서	10
결과 읽는 법	11
분석시트 입력 작업 만들기 (POST /api/power/extract)	11
작업 상태·결과 조회 (GET /api/power/extract)	14

CLIP API

이 문서는 CLIP 설치본에서 사용할 수 있는 API 를 담고 있습니다. 예제의 주소는 실제 설치 주소로 바꿔 사용하십시오.

시작하기

CLIP HTTP API 는 발전 분석시트 입력을(를) 고객 시스템에서 호출할 수 있도록 제공하는 API 입니다. 이 문서는 공통으로 적용되는 규칙과 기능별 호출 방법을 설명합니다.

모든 주소는 `https://<CLIP 설치 주소>` 을 기준으로 합니다. 실제 연동 시에는 담당자가 안내한 서버 주소로 바꿔 사용하십시오.

- 모든 요청은 고객 서버에서 호출해 주십시오. 브라우저나 모바일 앱에서 직접 호출할 수 없습니다(CORS 미지원).
- 인증이 필요한 모든 요청에는 `x-clip-key` 헤더로 API 키를 보내야 합니다. 인증 세부 사항은 다음 섹션을 참고하십시오.
- 파일을 보내는 요청은 `multipart/form-data` 를, 그 밖의 요청은 `application/json` 을 사용합니다.
- 일부 기능은 작업을 만든 뒤 작업 ID 로 결과를 조회하는 방식입니다. 자세한 내용은 비동기 작업 섹션을 참고하십시오.

연동을 시작할 때는 아래 순서로 서버 상태를 먼저 확인하는 것을 권장합니다. 상태 확인 경로는 인증 없이 호출할 수 있습니다.

첫 호출 예	CURL
<pre># 1. 서버 동작 확인 (인증 불필요) curl -s https://<CLIP 설치 주소>/api/health # 2. 키 상태와 잔여 한도 확인 curl -s https://<CLIP 설치 주소>/api/key-status -H "x-clip-key: YOUR_API_KEY"</pre>	

첫 번째 요청은 `{"status":"ok"}` 응답이면 서버가 정상 동작하는 것입니다. 두 번째 요청에서 키가 유효하면 요금제, 사용 가능한 기능, 한도 정보가 응답으로 옵니다.

인증

API 키는 모든 인증 요청의 `x-clip-key` 헤더로 보냅니다. 키 값을 요청 본문이나 주소의 쿼리 문자열에 넣지 마십시오.

```
curl -X POST https://<CLIP 설치 주소>/api/power/upload -H "x-clip-key: YOUR_API_KEY" -F "file=@/path/to/document.pdf"
```

API 키는 담당자에게 요청하여 발급받습니다. 키마다 사용할 수 있는 권한이 다르므로, 아래 표에서 연동할 기능에 필요한 권한을 확인하십시오.

기능	필요한 권한	비고
키 상태·잔여 한도 확인	키 활성화·만료 상태	GET https://<CLIP 설치 주소>/api/key-status

키에는 추가 기능(분류, 키 추출, 개인정보 탐지) 사용 여부가 설정되어 있을 수 있습니다. 설정이 꺼진 기능을 요청하면 403 이 옵니다.

인증 실패는 다음과 같이 구분됩니다.

- 키 헤더가 없거나 비어 있으면 401 이 오며, 본문의 `code` 값이 `api_key_required` 입니다. 배포 설정에 따라 키 헤더 없이도 요청이 처리될 수 있으나, 연동은 항상 키를 보내는 것을 전제로 하십시오.
- 키가 잘못되었거나 비활성·만료되었거나 권한이 없으면 401 또는 403 이 옵니다. 기능 API 응답의 `error` 문자열로 세부 사유를 확인할 수 있습니다. 다만 키 상태 확인 API(GET /api/key-status)는 업스트림 사유를 감추고 `{"error":"Upstream error 401"}` 처럼 상태 코드만 돌려주므로 401 과 403 을 구분할 수 없습니다. 이 API 의 502 본문은 `Failed to fetch: ...` 형식입니다.
- 분당 또는 일일 사용 한도를 넘으면 429 가 옵니다.

키 누락 응답 (401)

JSON

```
{"error":"우측 상단에 API Key 를 입력해 주세요.", "code":"api_key_required"}
```

API 키를 브라우저, 모바일 앱, 공개 저장소 등 클라이언트 쪽 코드에 넣지 마십시오. 키가 노출되었다고 판단되면 즉시 담당자에게 폐기와 재발급을 요청하십시오.

요청 형식

파일을 보내는 API 는 multipart/form-data 형식을 사용하며 파일 필드 이름은 `file` 입니다.

파일이 없는 POST 요청은 application/json 본문을 사용합니다. `Content-Type` 헤더를 `application/json` 으로 지정하고 JSON 본문을 보내십시오. 작업 조회는 GET 요청이며 본문이 없습니다.

```
curl -X POST https://<CLIP 설치 주소>/api/power/upload -H "x-clip-key: YOUR_API_KEY" -F "file=@/path/to/document.pdf"
```

기능별로 받는 파일 형식이 다릅니다. 아래 목록은 권장 형식이며, 목록에 없는 형식을 보내면 처리가 실패하거나 결과 품질이 떨어질 수 있습니다.

기능

지원 파일 형식

파일 크기와 페이지 수 상한은 계정 설정에 따릅니다. 상한을 넘는 파일은 413 으로 거절됩니다. 정확한 상한 값은 담당자에게 확인하십시오. 이미지 해상도는 기본적으로 자동 축소되어 처리됩니다. 파일 크기, 페이지 수, 픽셀 상한을 넘으면 413 으로 거절됩니다.

파일 업로드 요청을 보낼 때는 Content-Type 을 직접 지정하지 마십시오. HTTP 클라이언트가 multipart 경계 값을 포함한 헤더를 자동으로 만들도록 두어야 합니다.

비동기 작업(작업 ID 폴링)

문서 처리는 시간이 오래 걸리므로 여러 기능이 같은 방식의 비동기 작업으로 동작합니다. 기본 흐름은 다음과 같습니다.

1. 파일을 업로드 API 로 보내면 `fileId` 를 응답으로 받습니다. 업로드 응답의 HTTP 상태는 200 입니다.
2. 처리 API 를 POST 로 호출하면 `{"jobId": "..."}` 형태의 응답을 받습니다. 이 값이 작업 ID 입니다.
3. 같은 처리 API 주소에 `?job=작업ID` 를 붙여 GET 으로 호출합니다. 3초 간격으로 조회하십시오.
4. 응답의 `status` 가 `pending` 이면 다시 조회하고, `done` 또는 `error` 이면 조회를 멈춥니다.

처리 API 의 요청 본문과 페이지 단위 호출 방식은 기능별 문서를 따르십시오.

status 값	의미	처리 방법
<code>pending</code>	처리 중입니다.	3초 뒤에 다시 조회합니다.
<code>done</code>	처리가 끝났습니다.	결과는 <code>result</code> 또는 <code>results</code> 필드에 있습니다. 필드 이름은 기능에 따라 다릅니다.
<code>error</code>	처리에 실패했습니다.	본문의 <code>status</code> 는 항상 문자열 <code>"error"</code> 이고, <code>error</code> 필드에 사유가 옵니다. 실패 여부는 조회 응답의 HTTP 상태 코드로 판단하십시오. 처리·검토 API 는 처리 엔진의 상태 코드를 그대로 쓰며(알 수 없으면 502).

status 값	의미	처리 방법
unknown (HTTP 404)	작업이 없거나 완료되었거나 서버 재시작으로 사라졌습니다.	작업을 처음부터 다시 만들어야 합니다.

`done` 과 `error` 응답은 한 번 읽으면 서버에서 삭제됩니다. 응답을 받은 즉시 저장하십시오. 같은 작업을 다시 조회하면 404 unknown 이 옵니다. 응답이 유실되었다면 같은 작업을 재조회하지 말고 작업을 처음부터 다시 만들지 여부를 판단하십시오.

모니터링이나 상태 점검에 HEAD 요청을 사용하지 마십시오. HEAD 요청도 같은 조회 처리로 들어가 결과를 소비하므로, 이후 GET 조회는 404 unknown 이 될 수 있습니다.

- 작업 정보는 서버 메모리에만 있으며, 마지막 상태 변경 후 1시간이 지나면 만료됩니다.
- 서버가 재시작되면 진행 중이던 작업은 모두 사라지며 조회 시 404 unknown 이 옵니다.
- 여러 대의 서버를 앞단 로드밸런서로 묶은 환경에서는 작업을 만든 서버와 다른 서버로 조회될 수 있습니다. 이 경우에도 404 unknown 이 옵니다.

작업 ID 는 결과를 조회하는 값이므로 외부에 노출하지 마십시오.

작업 생성 요청(POST)이 200 을 돌려주었다고 해서 키가 유효하다는 뜻은 아닙니다. 키가 무효하거나 권한·한도가 부족하면 작업은 만들어지고, 실패 사유는 조회 단계에서 401, 403, 413, 429 등으로 나타납니다. 연동 전에 GET https://<CLIP 설치 주소>/api/key-status 로 키 상태를 먼저 확인하십시오.

작업 생성 요청을 다시 보내면 새 작업이 만들어져 처리 횟수가 중복으로 소모됩니다. 재시도 규칙은 처리 시간·재시도 섹션을 따르십시오.

오류 응답

오류 응답의 본문 형식은 API 마다 다릅니다. 공통된 형식이 없으므로, 먼저 HTTP 상태 코드로 분류하고 본문에서 code 또는 status 값, 그리고 error 문자열을 확인하십시오. 거절한 상위 시스템의 세부 필드(부족한 권한 이름, 허용량 등)는 전달되지 않습니다.

형식	본문 예	해당 API
키 누락	<code>{"error": "...", "code": "api_key_required"}</code>	인증이 필요한 모든 API
입력 오류 (400)	<code>{"error": "..."} </code>	필드가 없거나 형식이 맞지 않을 때입니다. 메시지는 엔드포인트마다 다릅니다(각 엔드포인트의 오류 표 참고).

형식	본문 예	해당 API
처리 오류 (상태 값 없음)	<code>{"error": "..."} </code>	파일 업로드, 근거 페이지 이미지, 키 상태 확인
작업 조회 실패	<code>{"status": "error", "error": "..."} </code>	비동기 작업 조회
작업 없음	<code>{"status": "unknown"} </code>	비동기 작업 조회 (HTTP 404)

처리 오류 예 (상태 값 포함)	JSON
<code>{"error": "처리 중 오류가 발생했습니다.", "status": 502}</code>	

상태 코드	의미	대응
400	입력 필드가 없거나 형식이 맞지 않습니다.	필드 이름과 형식을 확인하십시오. 같은 요청을 재시도하지 마십시오.
401	API 키가 없거나 잘못되었거나 비활성·만료되었습니다.	<code>code</code> 가 <code>api_key_required</code> 이면 키 헤더 누락입니다. 그 밖이면 키 값과 상태를 담당자에게 확인하십시오. 재시도하지 마십시오.
403	키 권한이 부족하거나 설정이 꺼진 기능을 요청했습니다.	필요한 권한을 담당자에게 확인하십시오. 재시도하지 마십시오.
404	작업이 없거나(<code>unknown</code>) 파일을 찾을 수 없습니다.	작업이 만료되었거나 재시작으로 사라졌는지 확인하십시오. 파일 ID 는 같은 키로 업로드한 것만 사용할 수 있습니다.
413	파일 크기, 페이지 수, 이미지 픽셀 상한 중 하나를 넘었습니다.	파일을 줄이거나 나누어 보내십시오. 상한은 담당자에게 확인하십시오. 재시도하지 마십시오.
429	분당 또는 일일 사용 한도를 넘었습니다.	60초 이상 기다린 뒤 재시도하십시오. 일일 한도는 날짜가 바뀌면 초기화되며, 기준은 한국 시간 오전 9시입니다.
5xx	서버 또는 문서 처리 엔진의 일시적인 오류입니다.	지수 백오프로 최대 2회 재시도하십시오. 이 경우 본문이 JSON 이 아닐 수 있습니다.

처리 시간·재시도

문서 1건의 판정이나 추출은 보통 수 초에서 수 분이 걸립니다. 문서 길이와 페이지 수, 스캔 여부에 따라 차이가 큼니다. 클라이언트 타임아웃은 20분 이상으로 설정하는 것을 권장합니다.

- 동시 요청은 문서 4~8건 이내로 제한하는 것을 권장합니다. 많이 보내면 한도에 빨리 도달하고 대기 시간이 길어집니다.
- 요청 1건이 내부적으로 여러 번의 처리 호출로 나뉘므로, 사용량 한도는 보낸 요청 수보다 빠르게 소모될 수 있습니다.

- 비동기 작업 조회는 3초 간격으로 하고, 전체 대기 시간에도 상한을 두어 상한을 넘으면 실패로 처리하십시오.

상황	재시도 여부	대기 방법
네트워크 오류 또는 5xx (조회 GET, 파일 업로드)	최대 2회	지수 백오프 (예: 2초, 4초)
작업 생성 POST 의 네트워크 오류	응답을 받지 못한 경우에만	5xx 를 포함해 응답을 받았다면 재시도하지 마십시오. 재시도하면 작업이 중복 생성됩니다
429	재시도합니다	60초 이상 기다린 뒤 재시도합니다. 응답에 Retry-After 헤더가 없으므로 고정 대기 시간을 사용하십시오
그 밖의 4xx	재시도하지 않습니다	요청 내용이나 키 권한을 고쳐야 합니다

작업 생성 요청(POST)을 재시도하면 새 작업이 생겨 처리 비용이 중복될 수 있습니다. 응답을 받지 못한 경우에만 재시도하고, 이미 받은 작업 ID 는 다시 생성하지 마십시오.

데이터 처리와 보관

업로드한 원본 파일과 OCR 로 추출한 텍스트는 문서 처리 엔진으로 전송되어 처리됩니다. 연동 전에 전송 경로가 고객 정책에 맞는지 보안 담당 부서와 확인하십시오.

- 기본 모드**에서는 업로드한 원본 파일이 문서 처리 엔진의 저장소에 남습니다. 파일은 API 키 단위로 격리되어 저장되며, 기간이 지나도 자동으로 삭제되지 않습니다.
- 저장된 원본을 삭제해야 하면 담당자에게 요청하십시오.

응답 JSON 과 고객 서버의 로그에도 판정 대상 문서의 내용이 들어 있을 수 있습니다. 개인정보와 기밀 문서를 다룬다면 고객 측 보관 기간과 접근 권한 정책을 함께 적용하십시오.

버전과 상태 확인

이 문서는 2026-10-28 기준입니다. API 경로에는 버전 번호가 없으며(예: /v1 없음), 서버 버전을 응답에서 확인할 수 없습니다. 연동 테스트 전에 담당자에게 배포된 버전을 확인하십시오.

- 응답에 새 필드가 추가될 수 있습니다. 모르는 필드는 무시하도록 파서를 작성하십시오.
- 변경 사항은 담당자를 통해 확인하십시오.

서버 동작 확인은 인증 없이 호출할 수 있습니다. 이 경로는 서버가 응답하는지만 확인하며, 문서 처리 엔진의 연결 상태나 API 키 상태는 확인하지 않습니다.

상태 확인	CURL
<pre>curl -s https://<CLIP 설치 주소>/api/health</pre>	

응답은 {"status":"ok"} 입니다. 키 상태와 잔여 한도는 `GET https://<CLIP 설치 주소>/api/key-status` 에 `x-clip-key` 헤더를 보내 확인하십시오.

기능

발전 분석시트 입력

발전소 ZLD(무방류) 수질 분석시트 — ZLD 전처리수 분석(양식 A), 증발농축기·결정화기 분석(양식 B) — 의 손글씨 스캔에서 지점·시각별 측정값과 일일 점검값을 뽑고, 양식에 인쇄된 관리기준을 넘은 값을 표시합니다. 값마다 원문 위치를 함께 돌려줍니다.

연동 순서

1. **작업 만들기** — `POST /api/power/extract` 에 분석시트 파일들(`files`)을 보내 `jobId` 를 받습니다(HTTP 202).
2. **결과 조회** — `GET /api/power/extract?job=...` 를 같은 키로 3초 간격 호출합니다.
3. **값 사용** — `done` 이면 `result.files[].sheets[]` 를 양식(`sheet_type`)과 날짜(`date`)로 구분해 읽습니다. 값마다 원문 위치(`evidence`)가 있어 근거를 보여 줄 수 있습니다.

전체 흐름 예 (Python, requests)

PYTHON

```
import time
import requests

BASE = 'https://<CLIP 설치 주소>'
HEADERS = {'x-clip-key': 'YOUR_API_KEY'}

def extract(paths, wait_limit=3600):
    files = [('files', (p.split('/')[1], open(p, 'rb')))] for p in paths]
    r = requests.post(BASE + '/api/power/extract', headers=HEADERS, files=files, timeout=600)
    r.raise_for_status() # 202
    job = r.json()[0]['jobId']
    started = time.time()
    while time.time() - started < wait_limit:
        time.sleep(3)
        s = requests.get(BASE + '/api/power/extract', headers=HEADERS, params={'job': job}, timeout=60)
        body = s.json()
        if body['status'] == 'pending':
            continue
        if body['status'] == 'done':
            return body['result']
        raise RuntimeError(f"{s.status_code} {body.get('error') or body['status']}")
    raise TimeoutError(job)

result = extract(['2026년 5월 13-31.pdf'])
```

```

for f in result['files']:
    for sheet in f['sheets']:
        for loc in sheet['locations']:
            for s in loc['samples']:
                print(sheet['date'], loc['location'], s['time'], s['values'])
            for o in sheet['out_of_spec']:
                print('기준 초과', sheet['date'], o['item'], o['value'], o['spec'])

```

결과 읽는 법

양식(sheet_type)	지점(location)	항목
A · zld_pretreatment	fgd_waste_water , reaction_clarifier_pond (process: primary_sedimentation · secondary_coagulation), pre_treated_water	pH·전도도·Ca·Mg·SO ₄ ·Cl. 지점마다 하루 최대 6회(칸 가로 3 × 2줄, 첫 줄 원→오른 다음 둘째 줄), 시각은 손으로 기입. 둘째 줄 칸의 값은 위 열 항목의 이어지는 측정값입니다. 비고란은 daily_checks
B · evap_crystallizer	evaporator_a , evaporator_b , crystallizer_a , crystallizer_b	pH·TS·TDS·TSS·SG 를 09:00·16:00·21:00·05:00 에 측정, Cl 은 하루 1회(09:00)

- 시트 날짜는 윤전일입니다. 양식 B 의 05:00 측정에는 day_offset: 1 이 붙습니다(실제 측정은 다음 날 새벽).
- 연도는 파일명에 "2026년" 처럼 있으면 그것을 씁니다. 시트에 인쇄된 연도가 틀린 경우가 있습니다. 손글씨 원문은 date_raw 에 남습니다.
- 값이 숫자가 아닌 칸은 values 에서 빠지고 flags 에 원문과 사유가 담깁니다. 빈 칸은 키가 없습니다.
- out_of_spec 은 양식에 인쇄된 관리기준을 넘은 값입니다(Evaporator·Crystallizer 의 pH·TDS·TSS, Pre-Treated Water 의 Ca·Mg). Cl 은 기준 표기(%)와 열 단위(mg/L)가 맞지 않아 현장 확인 전까지 검사하지 않습니다.
- page 는 있는데 boxes 가 없으면 그 쪽에서 읽었지만 정확한 자리는 찾지 못한 값입니다.

엔드포인트

POST /api/power/extract

분석시트 입력 작업 만들기

ZLD 분석시트 스캔 파일을 보내 지점·시각별 값 추출 작업을 만들고 jobId 를 받습니다.

꼭마다 양식(A·B)을 가려 지점·시각별 값으로 바꿉니다. 업로드 단계 없이 이 요청에 파일을 바로 보냅니다. 한 파일에 양식 A·B 시트가 날짜별로 섞여 있어도 꼭마다 나눕니다.

처리 시간 · 작업 생성은 즉시 끝납니다. 파일마다 전체를 한 번에 처리하며 수십 쪽 파일은 수 분이 걸릴 수 있습니다.

요청 · MULTIPART/FORM-DATA

이름	위치	타입	필수	설명
x-clip-key	header	string	필수	API 키입니다. 결과를 조회할 때도 같은 키를 보내야 합니다. 예: YOUR_API_KEY
files	form	file	필수	분석시트 파일입니다. 같은 이름으로 여러 번 보내면 여러 파일을 한 작업으로 처리합니다. PDF·TIFF·PNG·JPG·WEBP 를 받습니다. 연도는 파일명 ("2026년 ...")에서 읽으므로 기간이 드러나는 이름을 권장합니다. 예: 2026년 5월 13-31.pdf

응답 · 200

작업이 만들어지면 HTTP 202 로 응답합니다.

필드	타입	설명
jobId *	string	
total *	integer	받은 파일 수입니다.

응답 예	JSON
<pre>{ "jobId": "7b0c2f4e-9a1d-4e3b-8c5f-2d6a1b3c4e5f", "total": 1 }</pre>	

오류

상태	언제	본문 예
400	files 가 없을 때입니다.	{"error": "서류가 없습니다. files 필드로 파일을 보내 주세요."}
401	x-clip-key 헤더가 없을 때입니다.	{"error": "우측 상단에 API Key 를 입력해 주세요.", "code": "api_key_required"}

참고

- 서류 원본은 처리 엔진에 저장하지 않습니다. 처리하는 동안만 전달되고 남지 않습니다.
- 처리 엔진의 일시 오류(5xx)는 파일마다 2번까지 자동으로 다시 시도합니다.

cURL

CURL

```
# 1) 작업 만들기 — 파일 여러 개를 한 번에(files 를 반복)
curl -X POST 'https://<CLIP 설치 주소>/api/power/extract' \
  -H 'x-clip-key: YOUR_API_KEY' \
  -F 'files=@2026년 5월 13-31.pdf'
# → 202 {"jobId":"...", "total":1}

# 2) 결과 조회 — 같은 키로, 3초 간격
curl 'https://<CLIP 설치 주소>/api/power/extract?job=JOB_ID' -H 'x-clip-key: YOUR_API_KEY'
```

Python (requests)

PYTHON

```
import time
import requests

BASE = 'https://<CLIP 설치 주소>'
HEADERS = {'x-clip-key': 'YOUR_API_KEY'}

def extract(paths, wait_limit=3600):
    files = [('files', (p.split('/')[-1], open(p, 'rb')))] for p in paths]
    r = requests.post(BASE + '/api/power/extract', headers=HEADERS, files=files, timeout=600)
    r.raise_for_status() # 202
    job = r.json()['jobId']
    started = time.time()
    while time.time() - started < wait_limit:
        time.sleep(3)
        s = requests.get(BASE + '/api/power/extract', headers=HEADERS, params={'job': job}, timeout=60)
        body = s.json()
        if body['status'] == 'pending':
            continue
        if body['status'] == 'done':
            return body['result']
        raise RuntimeError(f'{s.status_code} {body.get("error") or body["status"]}')
    raise TimeoutError(job)

result = extract(['2026년 5월 13-31.pdf'])
for f in result['files']:
    for sheet in f['sheets']:
        for loc in sheet['locations']:
            for s in loc['samples']:
                print(sheet['date'], loc['location'], s['time'], s['values'])
        for o in sheet['out_of_spec']:
            print('기준 초과', sheet['date'], o['item'], o['value'], o['spec'])
```

```
const BASE = 'https://<CLIP 설치 주소>';
const headers = { 'x-clip-key': 'YOUR_API_KEY' };

async function extract(files) {
  const form = new FormData();
  for (const f of files) form.append('files', f); // File 또는 Blob
  const r = await fetch(BASE + '/api/power/extract', { method: 'POST', headers, body: form });
  if (r.status !== 202) throw new Error((await r.json()).error);
  const { jobId } = await r.json();
  for (;;) {
    await new Promise((ok) => setTimeout(ok, 3000));
    const s = await fetch(BASE + '/api/power/extract?job=' + jobId, { headers });
    const body = await s.json();
    if (body.status === 'pending') continue;
    if (body.status === 'done') return body.result;
    throw new Error(s.status + ' ' + (body.error || body.status));
  }
}
```

GET /api/power/extract

작업 상태·결과 조회

jobId 로 진행률을 확인하고, 끝나면 시트별 값을 받습니다.

3초 간격으로 조회하십시오. 작업을 만든 키와 같은 키를 `x-clip-key` 헤더로 보내야 합니다. 다른 키로 조회하면 없는 작업과 같게 404 가 옵니다.

처리 시간 · 조회는 즉시 응답합니다.

요청

이름	위치	타입	필수	설명
x-clip-key	header	string	필수	작업을 만든 키입니다. 예: YOUR_API_KEY
job	query	string	필수	작업 만들기 응답의 jobId 입니다. 예: 7b0c2f4e-9a1d-4e3b-8c5f-2d6a1b3c4e5f

응답 · 200

HTTP 200 이면 status 가 pending(진행률 포함) 또는 done(결과 포함)입니다.

object | object | object | object

```
{
  "status": "done",
  "result": {
    "files": [
      {
        "fileName": "2026년 5월 13-31.pdf",
        "pageCount": 1,
        "sheets": [
          {
            "page": 1,
            "sheet_type": "evap_crystallizer",
            "date": "2026-05-31",
            "date_raw": "2026년 5월 31일",
            "locations": [
              {
                "location": "evaporator_a",
                "samples": [
                  {
                    "time": "09:00",
                    "values": {
                      "ph": 6.64,
                      "tds_pct": 13.16,
                      "cl_mg_l": 65500
                    }
                  },
                  {
                    "time": "05:00",
                    "day_offset": 1,
                    "values": {
                      "ph": 6.62,
                      "tds_pct": 13.04
                    }
                  }
                ]
              },
              {
                "location": "crystallizer_a",
                "samples": [
                  {
                    "time": "09:00",
                    "values": {
                      "ph": 5.53,
                      "tds_pct": 43.55
                    }
                  }
                ]
              }
            ]
          }
        ]
      }
    ]
  }
}
```

```

],
"flags": [],
"out_of_spec": [
  {
    "path": "locations[1].samples[0].values.tds_pct",
    "item": "tds_pct",
    "value": 43.55,
    "spec": "42% 이하"
  }
],
"evidence": {
  "locations[0].samples[0].values.ph": {
    "raw": "6.64",
    "key": "evaporator_a_ph",
    "page": 1,
    "boxes": [
      [
        0.31,
        0.22,
        0.35,
        0.24
      ]
    ]
  }
}
],
"otherPages": [],
"pinKeys": [],
"warnings": []
}
],
"failed": []
}
}

```

오류

상 태	언제	본문 예
404	jobId 를 모르거나, 다른 키로 조회했거나, 만료(1시간) 되었거나, 서버가 재시작되었을 때입니다. 작업을 다시 만드십시오.	{"status":"unknown"}
401	x-clip-key 헤더가 없을 때입니다.	{"error":"우측 상단에 API Key 를 입력해 주세요.", "code":"api_key_required"}

참고

- 결과는 끝난 뒤 1시간 동안 여러 번 조회할 수 있습니다.
- 일부 파일만 실패하면 작업은 `done` 이고 실패한 파일은 `result.failed` 에 담깁니다.
- 작업 정보는 서버 메모리에 있어 서버가 재시작되면 사라집니다(404 `unknown`).

예제

cURL

CURL

```
curl 'https://<CLIP 설치 주소>/api/power/extract?job=JOB_ID' -H 'x-clip-key: YOUR_API_KEY'
```

