# AEGIS Fetcher Logic & Self-Recovery System

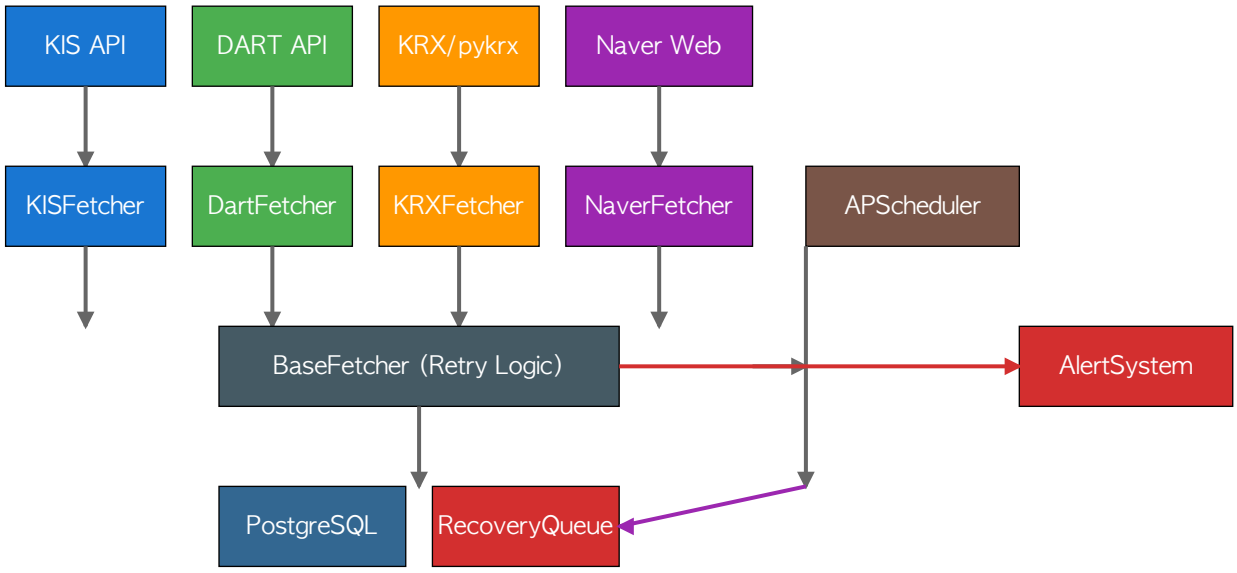
Version: v1.2.0 (Phase 38.1 Complete) | Generated: 2025-12-04 06:05

## 1. 시스템 개요

AEGIS Fetcher 시스템은 KIS, DART, KRX, Naver 등 다양한 외부 데이터 소스에서 주식 관련 정보를 수집합니다. 네트워크 오류, API 장애, Rate Limit 등 다양한 실패 상황에서 자동으로 복구하고 데이터 무결성을 보장하는 것이 핵심 목표입니다.

| 목표 | 설명 | 우선순위 |
|---|---|---|
| 데이터 무결성 | DB 저장 실패 시 재시도하여 데이터 손실 방지 | 최상 |
| 자동 복구 | 일시적 오류 발생 시 자동으로 재시도 | 상 |
| 장애 알림 | 복구 실패 시 사용자에게 즉시 알림 | 상 |
| Rate Limit 준수 | API Rate Limit 초과 방지 | 중 |
| 로깅 & 모니터링 | 모든 fetch 작업 추적 및 분석 | 중 |

## 2. Fetcher 아키텍처




## 3. Fetcher 상세 명세

### 3.1 KIS Fetcher (한국투자증권)

| 항목 | 내용 |
|---|---|
| 모듈 위치 | fetchers/kis/ |
| 주요 기능 | 실시간 시세 조회, 잔고 조회, 매매 실행 |

| | |
|---|---|
| 인증 방식 | OAuth 2.0 (App Key + Secret → Access Token) |
| 토큰 유효기간 | 24시간 (자동 갱신) |
| Rate Limit | 초당 20건 (권장) |
| 주요 에러 | 401 (토큰 만료), 429 (Rate Limit), 네트워크 오류 |
| 재시도 전략 | 토큰 만료 → 즉시 갱신, 429 → 지수 백오프 |

### 3.2 DART Fetcher (전자공시)

| 항목 | 내용 |
|---|---|
| 모듈 위치 | fetchers/dart_official/ |
| 주요 기능 | 기업 개요, 재무제표, 주요 주주, 임원 정보 |
| 인증 방식 | API Key (DART_API_KEY) |
| Rate Limit | 분당 1000건 |
| 주요 에러 | 401 (API Key 오류), 429 (Rate Limit), 데이터 없음 |
| 재시도 전략 | 429 → 1분 대기 후 재시도, 데이터 없음 → 스킵 |

### 3.3 KRX Fetcher (한국거래소)

| 항목 | 내용 |
|---|---|
| 모듈 위치 | fetchers/krx/ |
| 주요 기능 | 일별 OHLCV, 투자자별 매매동향 (수급) |
| 라이브러리 | pykrx, FinanceDataReader |
| Rate Limit | 없음 (웹 크롤링 기반, 1초 간격 권장) |
| 주요 에러 | 네트워크 오류, 데이터 없음 (휴장일) |
| 재시도 전략 | 네트워크 오류 → 지수 백오프, 휴장일 → 스킵 |

### 3.4 Naver Fetcher (네이버 금융)

| 항목 | 내용 |
|---|---|
| 모듈 위치 | fetchers/naver_finance/ |
| 주요 기능 | 뉴스, 컨센서스, 재무제표, 동종업계 비교, 리포트 |
| 인증 방식 | 없음 (비공식 API/웹 크롤링) |
| Rate Limit | 없음 (IP 차단 위험, 1~2초 간격 권장) |
| 주요 에러 | DNS 실패, HTTP 403/503, 파싱 실패 |
| 재시도 전략 | DNS/네트워크 → 지수 백오프, 403 → User-Agent 변경 |

### 3.5 Naver Fetcher 표준 인터페이스 (Phase 38.4)

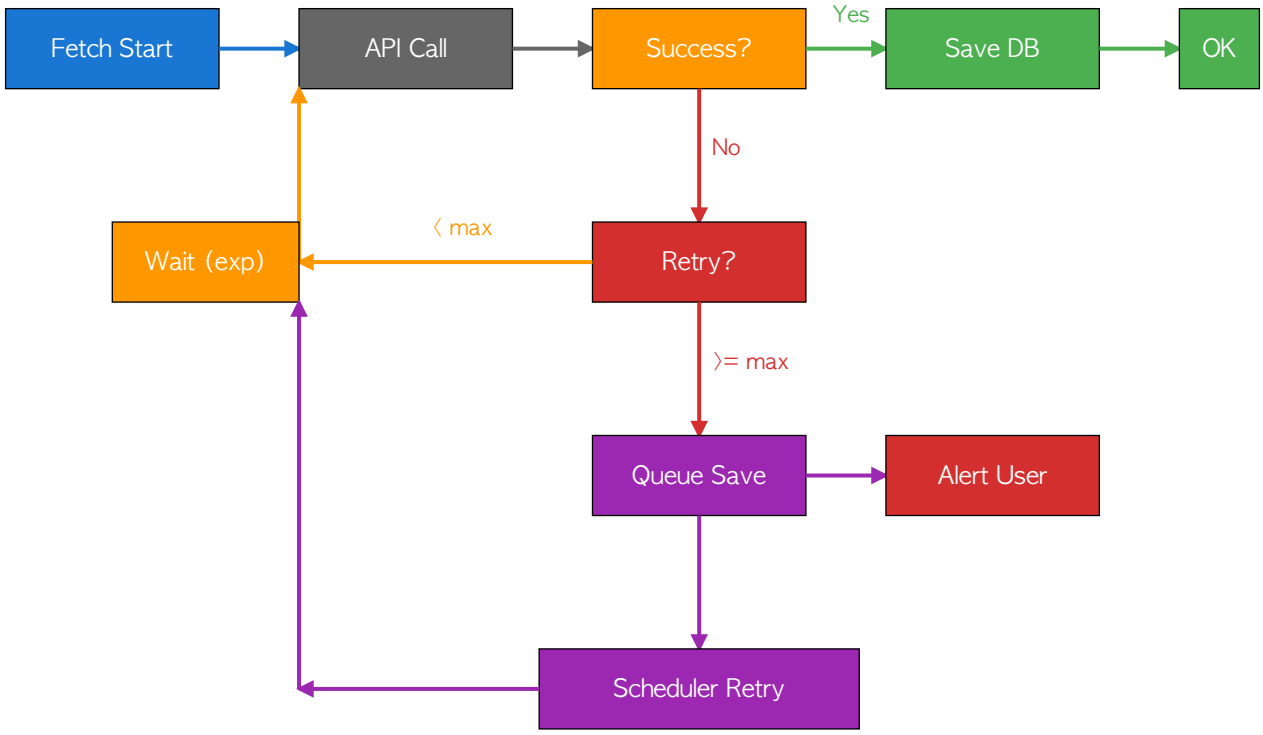
모든 Naver Fetcher가 BaseFetcher를 상속하여 동일한 인터페이스를 제공합니다. 이를 통해 target_codes 파라미터로 특정 종목을 지정하여 수집할 수 있습니다.

| Fetcher | 상속 | 표준 호출 | 레거시 호출 |
|---|---|---|---|
| NaverNewsFetcher | BaseFetcher | fetch(target_codes=[...]) | - |
| NaverConsensusFetcher | BaseFetcher | fetch(target_codes=[...]) | - |
| NaverReportFetcher | BaseFetcher | fetch(target_codes=[...]) | fetch_reports(code, pages) |

표준 인터페이스 사용 예시:

```
# ■■ ■■ ■■ ■■
    fetcher = NaverNewsFetcher()
    fetcher.fetch(target_codes=['316140', '005930'])
    # ■■ ■■ ■■■■ ■■
    fetcher = NaverConsensusFetcher()
    fetcher.fetch(target_codes=['316140'])
    # ■■ ■■ ■■■ ■■ (■■ ■■)
    fetcher = NaverReportFetcher()
    fetcher.fetch(target_codes=['316140'], pages=5)
    # ■■■ ■■ (■■ ■■)
    fetcher = NaverReportFetcher(db=session)
    fetcher.fetch_reports('316140', pages=3)
```

## 4. 자가 복구 플로우




## 5. 재시도 전략 (Exponential Backoff)

지수 백오프(Exponential Backoff)는 실패 시 대기 시간을 지수적으로 증가시키는 전략입니다. 이는 일시적인 장애 상황에서 서버에 과부하를 주지 않으면서 복구를 시도합니다.

| 시도 횟수 | 대기 시간 | 누적 시간 | 상황 |
|---|---|---|---|
| 1차 | 1초 | 1초 | 일시적 오류 가능성 |
| 2차 | 2초 | 3초 | 네트워크 지연 가능성 |
| 3차 | 4초 | 7초 | API 장애 가능성 |
| 4차 | 8초 | 15초 | Rate Limit 가능성 |
| 5차 | 16초 | 31초 | 심각한 장애 |
| 실패 | - | - | RecoveryQueue 저장 + 알림 |

## 6. Recovery Queue DB 설계

재시도 최대 횟수 초과 시 실패한 작업을 Recovery Queue에 저장하여 다음 스케줄러 실행 시 자동으로 재시도합니다.

| 컬럼명 | 타입 | 설명 | 필수 |
|---|---|---|---|
| id | Integer (PK) | 고유 식별자 | O |
| fetcher_name | String(50) | KISFetcher, DartFetcher 등 | O |
| target_code | String(20) | 종목코드 (예: 005930) | O |
| task_type | String(50) | price, news, consensus 등 | O |

| | | | |
|---|---|---|---|
| params_json | Text | 작업 파라미터 (JSON) | - |
| error_message | Text | 마지막 에러 메시지 | - |
| retry_count | Integer | 재시도 횟수 | O (기본 0) |
| max_retries | Integer | 최대 재시도 횟수 | O (기본 5) |
| status | String(20) | pending/processing/completed/failed | O |
| created_at | DateTime | 생성 시각 | O |
| updated_at | DateTime | 마지막 업데이트 | O |
| next_retry_at | DateTime | 다음 재시도 예정 시각 | - |

## 7. 에러 유형별 처리 전략

| 에러 유형 | 원인 | 처리 방법 | 재시도 |
|---|---|---|---|
| NetworkError | DNS 실패, 연결 끊김 | 지수 백오프 후 재시도 | O |
| TimeoutError | API 응답 지연 | 타임아웃 증가 후 재시도 | O |
| RateLimitError | API 호출 초과 | 대기 후 재시도 (최소 1분) | O |
| AuthError (401) | 토큰 만료/인증 실패 | 토큰 갱신 후 재시도 | O |
| NotFoundError (404) | 데이터 없음 | 스킵 (정상 케이스) | X |
| ServerError (5xx) | API 서버 장애 | 지수 백오프 후 재시도 | O |
| ParseError | 응답 파싱 실패 | Queue 저장 + 로깅 | △ |
| DBError | DB 저장 실패 | Queue 저장 + 재시도 | O |

## 8. 스케줄러 통합 계획

스케줄러 통합 계획은 별도 문서로 관리됩니다

→ 상세 내용: reports/AEGIS/Schedule_Logic.pdf

Schedule_Logic.pdf에는 다음 내용이 포함되어 있습니다:

| 항목 | 설명 |
|---|---|
| 시스템 Cron | 1분, 10분, 20분, 30분, 3시간 주기 작업 |
| 문서 생성 | Dashboard, Report 자동 생성 스케줄 |
| Fetcher 스케줄 | KIS, KRX, DART, Naver 데이터 수집 주기 |
| Recovery Worker | 10분마다 실패 작업 재시도 |
| Gemini API | Rate Limit (15 RPM, 1500 RPD, 1M TPM) |
| 이메일 알림 | 에러 발생 시 즉시 알림 + 20분 주기 대시보드 |
| fswatch 연동 | AEGIS 문서 자동 동기화 |

이렇게 문서를 분리함으로써 중복을 피하고 단일 진실 공급원(Single Source of Truth)을 유지합니다.

## 9. 데이터 신선도 검증 (Freshness Check)

수집된 데이터가 오래되었거나 누락된 경우 자동으로 재수집을 트리거합니다.

| 데이터 유형 | 신선도 기준 | 검증 시점 | 재수집 트리거 |
|---|---|---|---|
| 뉴스 | 24시간 이내 | 30분 cron | 마지막 뉴스 〉 24h |
| 컨센서스 | 7일 이내 | 3시간 cron | consensus.date 〉 7d |
| 수급 데이터 | 당일 | 장 마감 후 | 오늘 데이터 없음 |
| 일별 시세 | 전일 | 09:00 | 어제 데이터 없음 |
| 실시간 시세 | 5분 이내 | 1분 cron | timestamp 〉 5min |

## 10. 알림 시스템 (Alert System)

| 알림 레벨 | 조건 | 알림 채널 | 예시 |
|---|---|---|---|
| INFO | 정상 완료 | 로그만 | Fetch 완료: 10건 |
| WARNING | 부분 실패 | 로그 + 대시보드 | 3/10 종목 실패 |
| ERROR | 전체 실패 | 로그 + 대시보드 + 이메일 | API 인증 실패 |
| CRITICAL | 시스템 장애 | 모든 채널 + SMS | DB 연결 불가 |

## 11. 구현 계획

| 단계 | 작업 | 예상 소요 | 상태 |
|---|---|---|---|
| Phase 1 | BaseFetcher에 retry 로직 추가 | 완료 | DONE |
| Phase 2 | FetcherRecoveryQueue 모델 생성 | 완료 | DONE |

| | | | |
|---|---|---|---|
| Phase 3 | RecoveryWorker 스케줄러 작업 추가 | 완료 | DONE |
| Phase 4 | KIS/DART/KRX Fetcher 통합 | 완료 | DONE |
| Phase 5 | Naver Fetcher 통합 | 완료 | DONE |
| Phase 6 | 데이터 신선도 검증 로직 | 완료 | DONE |
| Phase 7 | 이메일 알림 시스템 | 완료 | DONE |

## 12. BaseFetcher 코드 설계

BaseFetcher 핵심 로직:

```
class BaseFetcher:
    MAX_RETRIES = 5
    BASE_DELAY = 1  # seconds
    def fetch_with_retry(self, target_code, task_type, **kwargs):
        for attempt in range(self.MAX_RETRIES):
            try:
                result = self._fetch_impl(target_code, **kwargs)
                return result
            except RetryableError as e:
                delay = self.BASE_DELAY * (2 ** attempt)
                time.sleep(delay)
            except NonRetryableError as e:
                self._save_to_queue(target_code, task_type, str(e))
                raise
        # Max retries exceeded
        self._save_to_queue(target_code, task_type, "Max retries exceeded")
        self._send_alert(f"Fetch failed: {target_code}")
```

## 13. RecoveryWorker 코드 설계

RecoveryWorker 핵심 로직:

```
class RecoveryWorker:
    def process_queue(self):
        pending_tasks = db.query(FetcherRecoveryQueue).filter(
            status='pending',
            next_retry_at <= datetime.now()
        ).all()
        for task in pending_tasks:
            fetcher = self._get_fetcher(task.fetcher_name)
            try:
                fetcher.fetch(task.target_code)
                task.status = 'completed'
            except Exception as e:
                task.retry_count += 1
                if task.retry_count >= task.max_retries:
                    task.status = 'failed'
                    self._send_critical_alert(task)
                else:
                    task.next_retry_at = datetime.now() + timedelta(
                        seconds=60 * (2 ** task.retry_count)
                    )
            db.commit()
```

## 14. Email Alert System (Phase 38.1)

Phase 38.1에서 추가된 이메일 알림 시스템은 Fetcher 실패 및 Recovery 실패 시 자동으로 이메일을 발송하여 즉각적인 대응이 가능하도록 합니다.

### 14.1 환경 설정 (.env)

| 환경변수 | 설명 | 예시 |
|---|---|---|
| ALERT_EMAIL_ENABLED | 이메일 알림 활성화 | true |
| ALERT_SMTP_HOST | SMTP 서버 | smtp.gmail.com |
| ALERT_SMTP_PORT | SMTP 포트 | 587 |
| ALERT_SMTP_USER | 발송 이메일 | your@gmail.com |
| ALERT_SMTP_PASSWORD | Gmail 앱 비밀번호 | 16자리 앱 비밀번호 |

| | | |
|---|---|---|
| ALERT_EMAIL_TO | 수신자 이메일 | admin@example.com |
| ALERT_EMAIL_FROM | 발신자 표시명 | AEGIS 〈your@gmail.com〉 |

### 14.2 알림 발송 시점

| 발송 시점 | 알림 레벨 | 내용 |
|---|---|---|
| Fetcher 최대 재시도 초과 | ERROR | 종목코드, 에러 유형, 에러 메시지 |
| RecoveryWorker 최종 실패 | ERROR | 실패 작업 목록 (최대 10건) |
| 데이터 신선도 경고 | WARNING/ERROR | 오래된 데이터 종목 목록 |
| 시스템 장애 | CRITICAL | 장애 내용 및 상태 |

### 14.3 중복 알림 방지

같은 종류의 알림이 30분 내에 반복 발송되는 것을 방지합니다. alert_key를 기반으로 최근 발송 기록을 관리하며, 쿨다운 기간 내에는 동일 알림을 무시합니다.

### 14.4 Gmail 앱 비밀번호 설정 방법

1. Google 계정 로그인 (myaccount.google.com)
2. 보안 〉 2단계 인증 활성화
3. 보안 〉 앱 비밀번호 생성
4. '앱 선택' 〉 '기타' 〉 'AEGIS' 입력
5. 생성된 16자리 비밀번호를 ALERT_SMTP_PASSWORD에 설정

### 14.5 모듈 구조

| 파일 | 설명 |
|---|---|
| alerting/__init__.py | 패키지 초기화 |
| alerting/email_alert.py | EmailAlert 클래스, 발송 로직 |
| fetchers/base_fetcher.py | BaseFetcher에서 알림 연동 |
| .env | SMTP 설정 환경변수 |

### 14.6 설정 완료 상태

| 항목 | 상태 | 비고 |
|---|---|---|
| alerting 모듈 | 완료 | alerting/email_alert.py |
| BaseFetcher 연동 | 완료 | 최대 재시도 초과 시 알림 |
| RecoveryWorker 연동 | 완료 | 최종 실패 시 알림 |
| .env SMTP 설정 | 완료 | Gmail 앱 비밀번호 사용 |
| 이메일 발송 테스트 | 성공 | 2025-12-02 18:39 확인 |

### 14.7 테스트 이메일 샘플

아래는 실제 발송된 테스트 이메일 형식입니다:

```
Subject: [AEGIS ERROR] AEGIS ■■■ ■■
    From: AEGIS System
    To: joungwonlim@gmail.com
    ■ ERROR: AEGIS ■■■ ■■
    ■■■■: 2025-12-02 18:39:00
    ■ ■■■■ AEGIS ■■ ■■■ ■■■■■■.
```

[redacted].