

# 정산 시스템 설계서

'22.08.12

김동석

1. 개요	2
2. 해결해야 할 문제	3
3. 시스템 목표	3
4. 시스템 특징	3
5. 분석	3
5.1. 업무 정의	3
5.2. AS-IS	5
5.2.1. 마운드 미디어 음원 transaction flow 와 정산	5
5.2.2. 포크라노스 정산 시퀀스	6
5.2.3. 오즈 정산 시퀀스	7
5.2.4. 오즈 정산 유스케이스	9
5.3. 데이터 흐름	9
5.4. TO-BE	9
6. 설계	9
6.1. 공통코드	10
6.2. 네이밍	10
6.3. 공통 자바 패키지 구조	10
6.4. 컨테이너 자바 패키지 구조	10
6.5. 컨테이너 다이어그램	11
6.6. 유스케이스 정의	13
6.6.1. 액터	13
6.6.2. 유스케이스	13
6.7. 도메인 모델	16
6.8. 컴포넌트	18
6.8.1. 백엔드 Back-End	18
6.8.1.1. mybatis CRUD	18
6.8.1.2. 보안	18
6.8.1.3. 시나리오 테스트	19
6.8.2. 파일 매니저	19
6.8.2.1. 파일 저장 정책	19
6.8.2.2. 업로드	19
6.8.2.3. 다운로드	19
6.8.3. 계산기 SDS(Sum-Divide-Subtract???)	20
6.8.4. 메시지 발송기 Message Sender(alpha 범위 아님)	20
6.9. 클래스 설계	20
6.9.1. 공통 🖱️ doxygen 문서 보기	20
6.9.2. 백엔드 🖱️ doxygen 문서 보기	21
6.9.3. 파일 매니저 🖱️ doxygen 문서 보기	21
6.9.3.1. 로더	21
6.9.3.2. 파일 생성	23

6.9.4. 계산기 SDS(Sum-Divide-Subtract)	24
6.9.4.1. 업무	24
6.10. RESTful API	24
6.10.1. 규약	24
6.10.2. 정산 API 목록	24
6.11. 데이터 유효성 검사	24
6.12. DB	32
6.12.1. fractional seconds rounding problem	32
<b>7. 단어 사전</b>	<b>32</b>
7.1. 금액	33
7.2. 그 외	33
<b>8. 기타</b>	<b>35</b>

# 1. 개요

1. 정산 : 판매사(DSP), 중개사로 부터 받은 상품별 판매자료를 취합하여 수수료를 차감한 금액(로열티)을 권리자에게 지급하는 과정
2. 주요 기능
  - a. 권리자, 계약 관리
  - b. 카탈로그 관리
  - c. 수익분배 관리
  - d. 판매 데이터 관리
  - e. 정산 처리
3. 사용자 : 경영운영실, 브랜드 담당자
4. 접속주소
  - a. 운영
    - i. front-end : <https://sett-alpha.moundmedia.net/>
    - ii. back-end : <https://sett-alpha-backend.moundmedia.net/swagger-ui/index.html>
  - b. 스테이징  
개발 완료 이후 셋팅 예정
  - c. 개발
    - i. front-end : <http://10.1.40.53:9122>
      1. 반영하기

```
$ cd ~/dev/sett/stg/front-end
$ ./runStg.sh
```
    - ii. back-end : <http://10.1.40.53:9121/v0.1/swagger-ui/index.html>
      1. 반영하기


```
$ cd ~/dev/sett/stg/back-end
$ ./runStg.sh
```
5. 소스 저장소
  - a. 공통 java 라이브러리 : [https://gitlab.com/dx\\_team/settlement/sett-common](https://gitlab.com/dx_team/settlement/sett-common)
  - b. front-end : [https://gitlab.com/dx\\_team/settlement/front-end](https://gitlab.com/dx_team/settlement/front-end)
  - c. back-end : [https://gitlab.com/dx\\_team/settlement/back-end](https://gitlab.com/dx_team/settlement/back-end)
  - d. file manager : [https://gitlab.com/dx\\_team/settlement/file-manager](https://gitlab.com/dx_team/settlement/file-manager)
  - e. 계산기 : [https://gitlab.com/dx\\_team/settlement/calculate](https://gitlab.com/dx_team/settlement/calculate)
6. 참고
  - a. [DTT 정산시스템 Agenda Draft.](#)
  - b. [SDS\(Sum Divide Subtract\) : 자체 정산 P.O.C](#)

# 2. 해결해야 할 문제

1. OCMS의 제약사항
  - a. 기능 수정/추가 불가능
  - b. 엑셀 기반 : 실 데이터는 엑셀로 관리 함. 업무 난이도 증가, 파일 중복/유실 문제 발생(참고 : [\(POCLANOS\) 매뉴얼](#), [음원 정산 프로세스](#), [OZZ 관리자 정산 운영 매뉴얼](#))
  - c. 데이터 활용이 힘들
  - d. 담당자 PC에 설치 필요. PC자원을 많이 사용

2. 관리자 정산서 발송 문제 : pc의 아웃룩으로 발송됨
  - a. 시간이 오래 걸림. 발송 중 다른 업무 수행 불가
  - b. 오류처리는 발송자가 직접 처리해야 함. 실수 발생 가능성 높음

### 3. 시스템 목표

1. 업무 편의성 향상
2. 빠른 처리(업로드, 계산, 알림) 속도
3. 정확한 계산
4. 데이터(매출/정산) 온라인화
5. 확장성 제공
  - a. [매출/인세 관리 시스템](#)으로 발전 고려 : 참고  [마운드 미디어 시스템 컨텍스트 다이어그램](#)
  - b. 국제 표준 적용(ex. [DDEX sales/usage reporting \(DSR\)](#))
  - c. 통계 추가

### 4. 시스템 특징

1. 대용량 : 유튜브 '22.05판매 로그 데이터 700만건(광고, 레드 포함)
2. 판매 데이터 포맷이 DSP별로 상이함

## 5. 분석

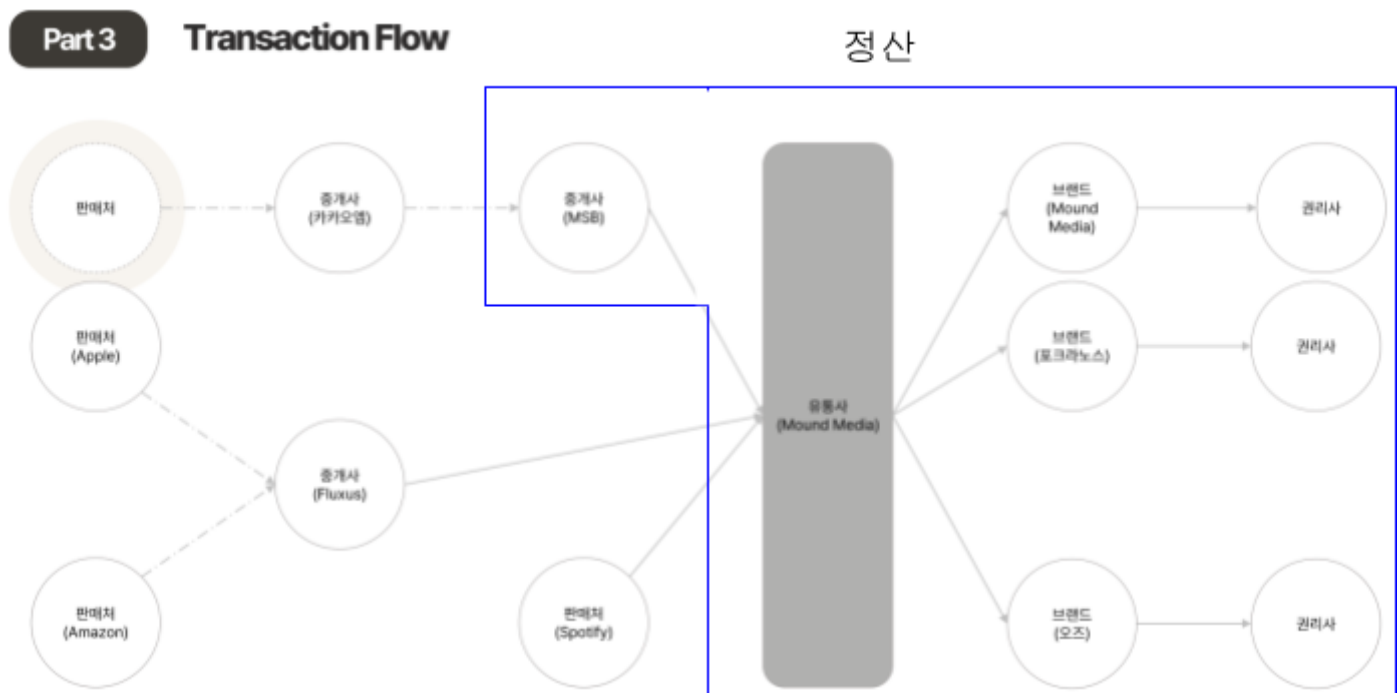
### 5.1. 업무 정의

1. 정산
  - a. 정의 : 중개사에서 제공하는 정산자료(에셋별 판매액)을 기반으로 관리자들의 수익금(?)을 확정하는 행위.
  - b. 주기 : 매달 정해진 특정일
  - c. 주체 : 해당 브랜드 담당자
2. 관리자 : 에셋 소유자. 수익금(?) 지급 대상자. 관리자는 에셋을 양도할 수 있다(관리이관) 즉, 시점에 따라 관리자가 다를 수 있다.
3. 에셋
  - a. 정의 : 판매 대상 상품
  - b. 상하관계 : 에셋은 부모-자식 관계로 다른 에셋에 속할 수 있다. 하지만 하나의 에셋이 여러 부모에 속하지는 않는다.
  - c. 수수료율
    - i. 정의 : 수수료를 산정하는 기준.
    - ii. 에셋에만 있는 속성.
    - iii. 부모-자식-형제 에셋의 수수료율은 서로 상관 관계가 없다.
4. 수수료 : 판매액에서 차감할 금액. 판매액에 수수료율을 곱해서 산정. 모두 합치면 브랜드의 수익이 됨
5. 수수료 계산 시 수수료율 선택 기준  
기본적으로 에셋의 수수료율을 적용하여 산정하지만 정산 시점에 아래의 경우가 있다면 별도의 수수료율이 적용되어 있으므로 해당 수수료율을 적용.

- a. 중개 계약 : 중개사에게 특정 에셋의 유통을 의뢰하는 경우(?). 에셋의 판매 지역(국내, 국외, 특정 국가)에 따라 다르게 수수료율 설정 가능.
  - b. 선급 계약 : 권리자에게 대부를 해주고 원금을 수익금에서 자동 차감
    - i. 수수료율 적용 기간
      - 1. 대부금이 모두 상환되었어도 선급 수수료율을 특정 기간까지 적용할 수 있다.
      - 2. 미상환 대부금이 있어도 에셋의 기본 수수료가 적용되도록 할 수 있다.
    - ii. 선급은 여러번 가능하고 총 선급액은 누적된다.
    - iii. 선급금은 수익금에서 차감하는 방식으로 자동 상환되지만 다른 방식으로 상환도 가능하다.
6. 수익 분배
- a. 정의 : 특정 에셋의 수익을 다른 권리자들에게 지급하는 행위
  - b. 분배 금액 : 판매액에 사전 정의된 비율을 곱해서 산정되는 금액
7. 이월
- a. 정의 : 권리자에게 수익금의 전부 혹은 일부를 전달할 수 없는 경우 다음달의 수익금에 합산하는 절차
  - b. 유형
    - i. 입금액(?)이 최소 입금액 미달
    - ii. 세금계산서 미발행(권리자가 사업자인 경우)
    - iii. 입금실패
    - iv. 그 외

## 5.2. AS-IS

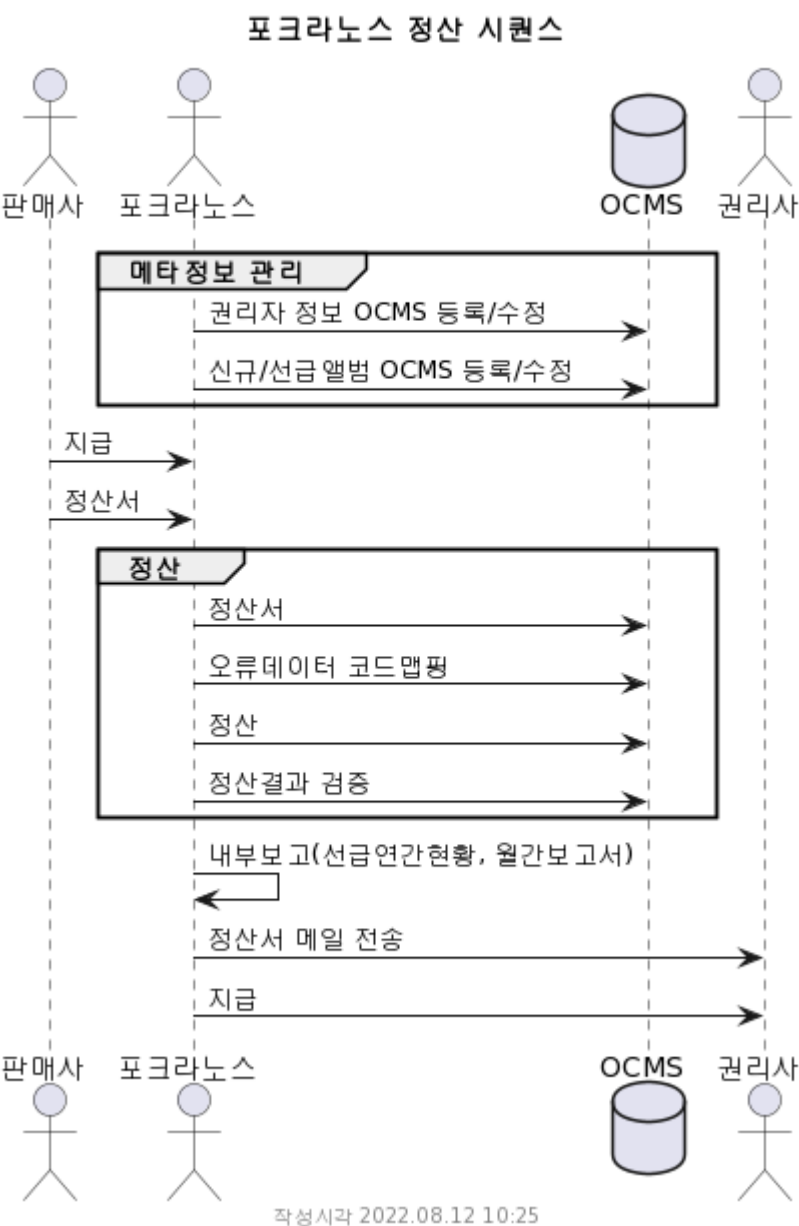
### 5.2.1. 마운드 미디어 음원 transaction flow 와 정산



[출처 : [Digital Transformation Team Business](#)]

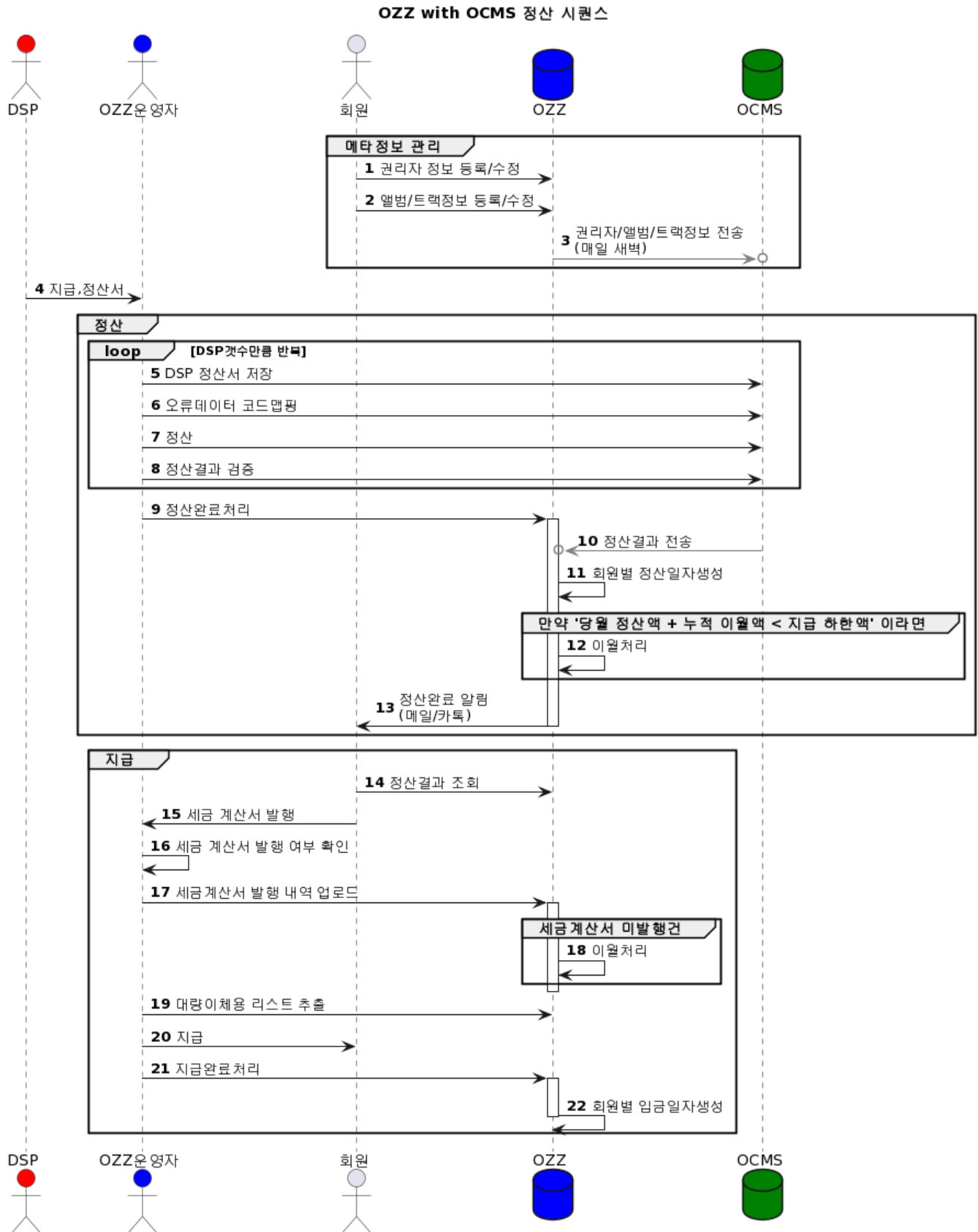
5.2.2. 포크라노스 정산 시퀀스

Warning:  
대외비



### 5.2.3. 오즈 정산 시퀀스

Warning:  
대외비

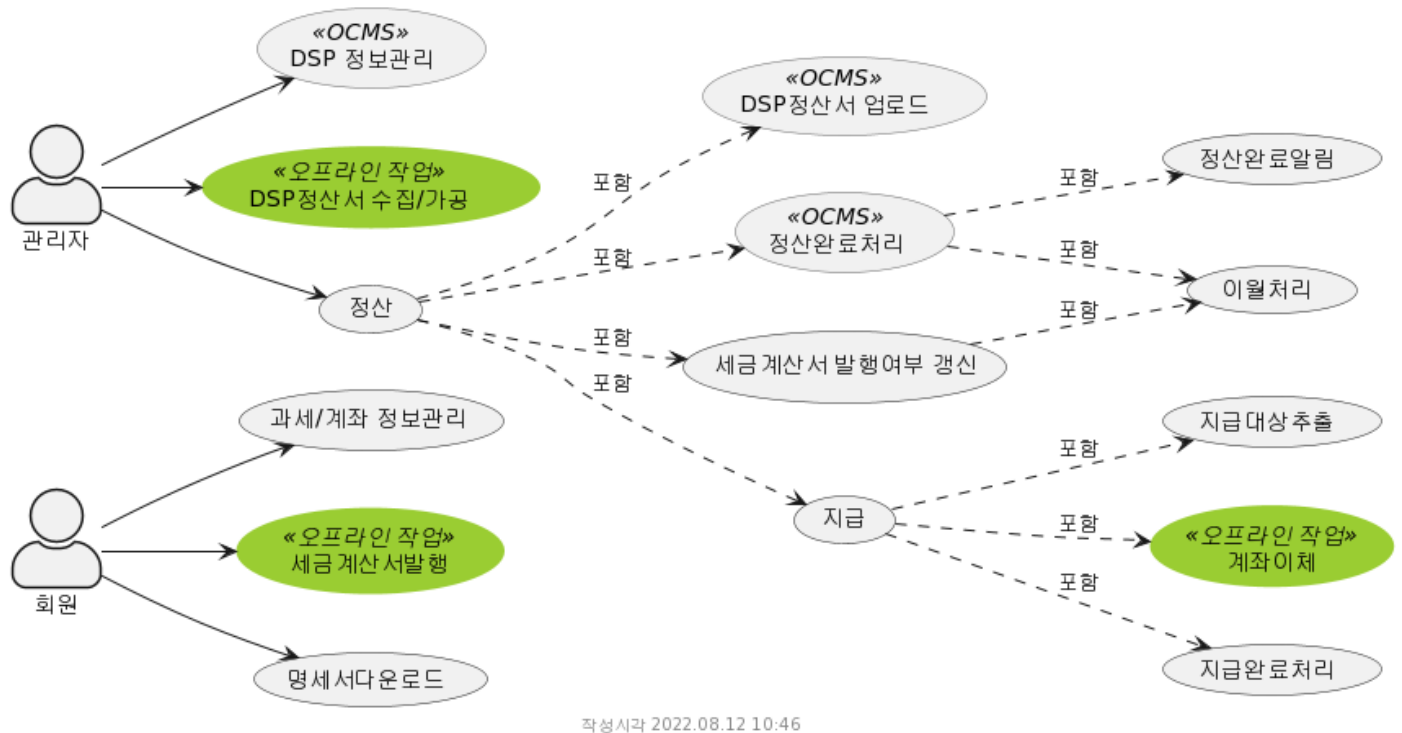




## 5.2.4. 오즈 정산 유스케이스

Warning:  
대외비

OZZ 정산 업무 유즈케이스



## 5.3. 데이터 흐름

1. 업로드 : DSP별로 수신한 판매 데이터 파일
2. 정제 : 무의미한 판매 데이터 제거(매출액 없음, 식별 불가능한 데이터...). 자체 포맷으로 변환
3. 저장 : 판매 데이터를 통일된 형식으로 동일 저장소에 저장
4. 대사
5. 월별 합산 : asset별 + item별 + DSP별
6. 정산 : 분배,상계(선지급),징수,이월
7. ~~알림 : 최종 정산 데이터(alpha의 범위 아님)~~
8. 지급 : 입금 결과 데이터(세금 계산서 미발급 시 이월)
9. 통계

## 5.4. TO-BE

## 6. 설계

※ 화면을 제외한 모든 컨테이너는 java, spring, mybatis기반임

## 6.1. 공통코드

1. ISO 3166(국가코드), 4217(통화)와 같은 국제표준 코드는 거의 변경되지 않는 정보이므로 각각 컨테이너에서 관리 함(호출 부하와 개발공수를 줄일 수 있음)
  - a. Java : <https://mvnrepository.com/artifact/com.neovisionaries/nv-i18n>
  - b. Json : <https://github.com/vijinho/ISO-Country-Data>
2. 코드 설계 *TODO*

## 6.2. 네이밍

※ [마운드 미디어 개발 가이드](#) 참조

※ 컨테이너 별칭

1. 공통 : comm
2. 백엔드 : api
3. 파일 매니저 : file
4. 메시지 발송기 : msg
5. 계산기 : calc

## 6.3. 공통 자바 패키지 구조

Package tree			Description
com.mm.sett. <b>comm</b>			기본 package
	conf		설정 정보 관련 package
	controller		Spring MVC controller 관련 package
	domain		도메인 모델 관련 package
	service		비즈니스 로직 관련 package
	mapper		Mybatis DB 쿼리 mapping 관련 package
	enum		공통 enum

## 6.4. 컨테이너 자바 패키지 구조

Package tree			Description
com.mm.sett.{컨테이너 별칭}			정산 기본 package
	conf		설정 정보 관련 package
	controller		controller 관련 package
	domain		도메인 모델 관련 package

	service		비즈니스 로직 관련 package
	mapper		Mybatis DB 쿼리 mapping 관련 package
	comm		공통 로직 관련 package

## 6.5. 컨테이너 다이어그램

※ 컨테이너 : 소스 저장소/개발/반영/이중화의 단위, 컨테이너들은 서로 직/간접적으로 연동

※ 데몬 : Job큐(테이블)을 통해 작업요청을 송수신

※ 데몬 개발을 편하게 하기 위해 개발환경에서는 각각 기능에 대해 웹기반 api를 띄워서 개발

1. 프론트 엔드(Web) : 관리 화면
2. 백 엔드(Spring/REST)
  - a. 파일 업로드 : DSP 판매 데이터(정산서) 필터링, 브랜드/채널 코드 추가, 원본 DB 저장
  - b. 파일 다운로드 : 각종 파일생성/분할/압축 (정산서 파일, GRID 파일 생성 제외)
3. 파일 매니저(daemon) : 스케줄러
  - a. 원본 파일 정보 DB로 저장, 대사
  - b. 정산서 파일, GRID 파일 생성
4. 계산기(daemon) : 스케줄러
  - a. 매출 합산, 분배, 징수, 관리자 정산서 생성 요청(to 파일 매니저)
  - b.
- ~~5. 메서지 발송기(daemon) : email, kakao talk, sms 등으로 관리자, 운영자 등 사용자에게 알림 발송~~
6. 스케줄러 : 주기적 작업 실행
  - a. 매일마다 : 계약 자동 연장
 

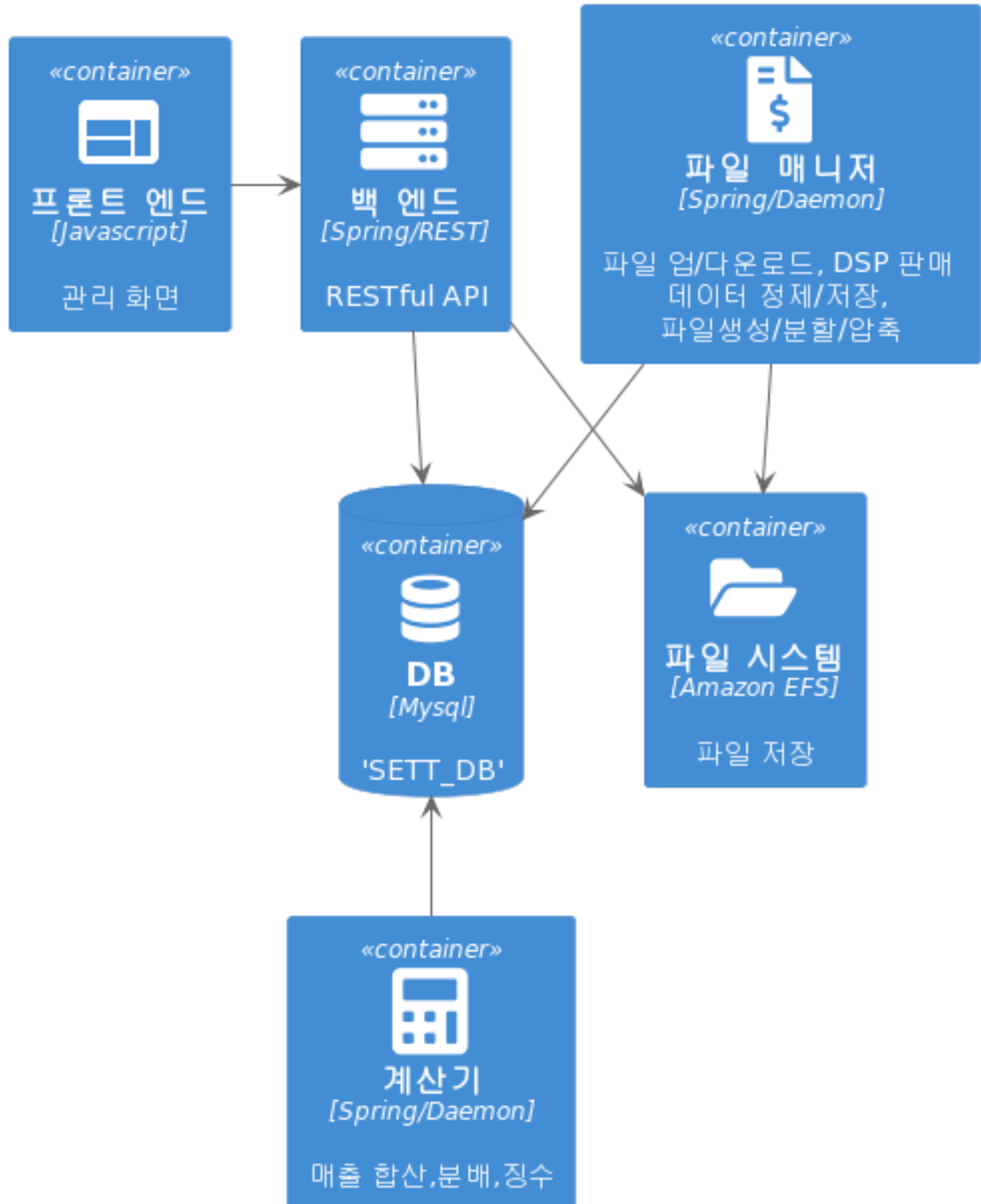
👉 이것 외에 더 있는가. 단순히 주기적인 작업을 하기 위해 컨테이너를 따로 띄우는 것보다 백엔드에서 처리할 수 있는지 고려 필요
7. 파일 시스템 : Amazon EFS, ~~S3~~
  - a. EFS 셋팅 완료 : /data
8. DB : MySQL

Warning:

대외비

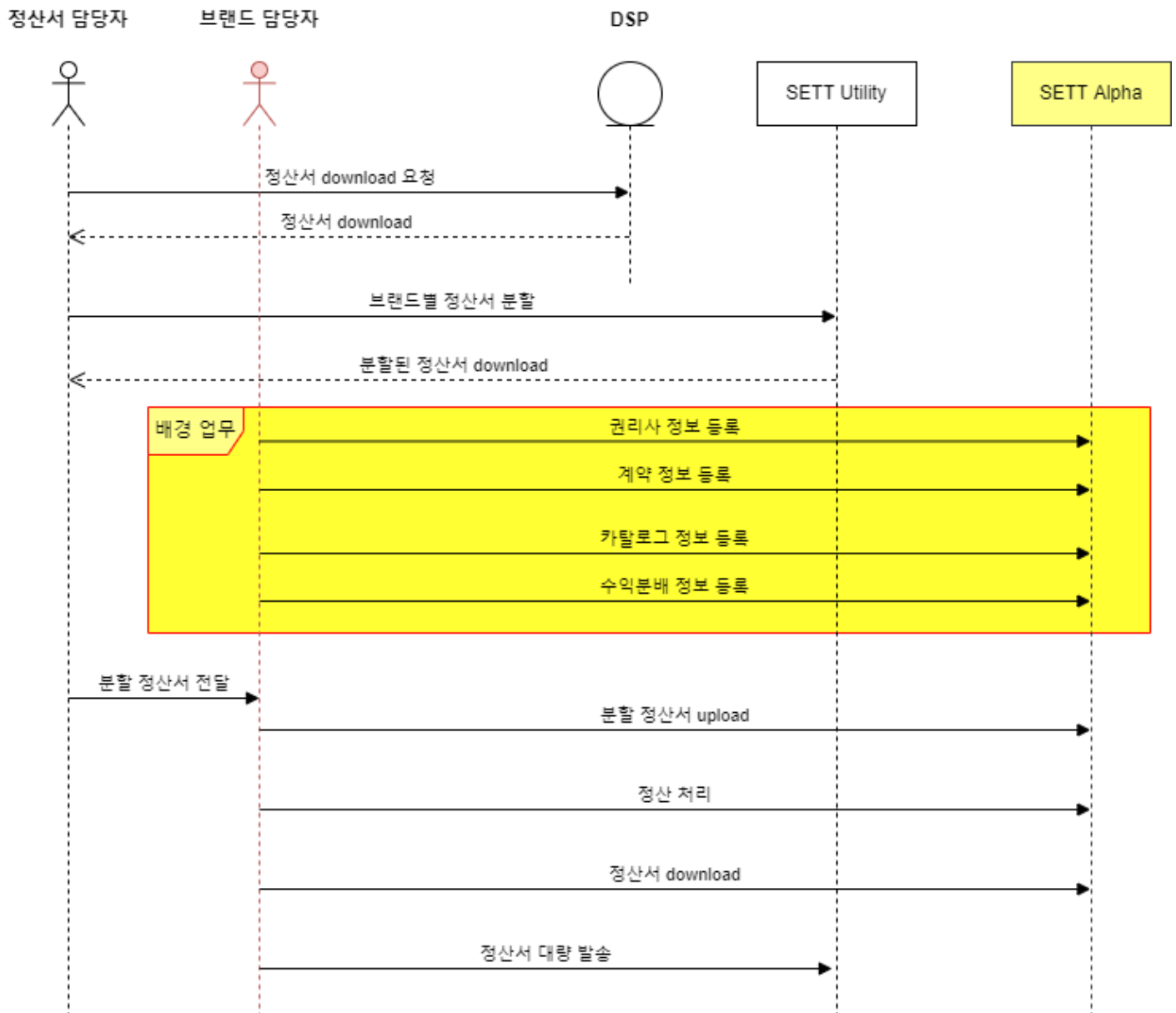
※ 화살표는 의존성을 나타냄

## Mound Media 정산시스템 컨테이너



작성시각 2022.09.26 02:39

## 6.6. 유스케이스 정의



### 6.6.1. 액터

※ 직/간접적으로 시스템을 이용할 주체(사용자, 내/외부 시스템)

※ 브랜드 : 포크라노스, 오즈, MSB

1. 브랜드 담당자 : 포크라노스, 오즈, MSB

2. 정산 유틸리티 : 판매 데이터(DSP 정산서)를 브랜드별로 분할, 관리자 정산서 발송

### 6.6.2. 유스케이스

※ 구현할 업무 범위 내에서 일어나는 일들(화면/메뉴와 무관)

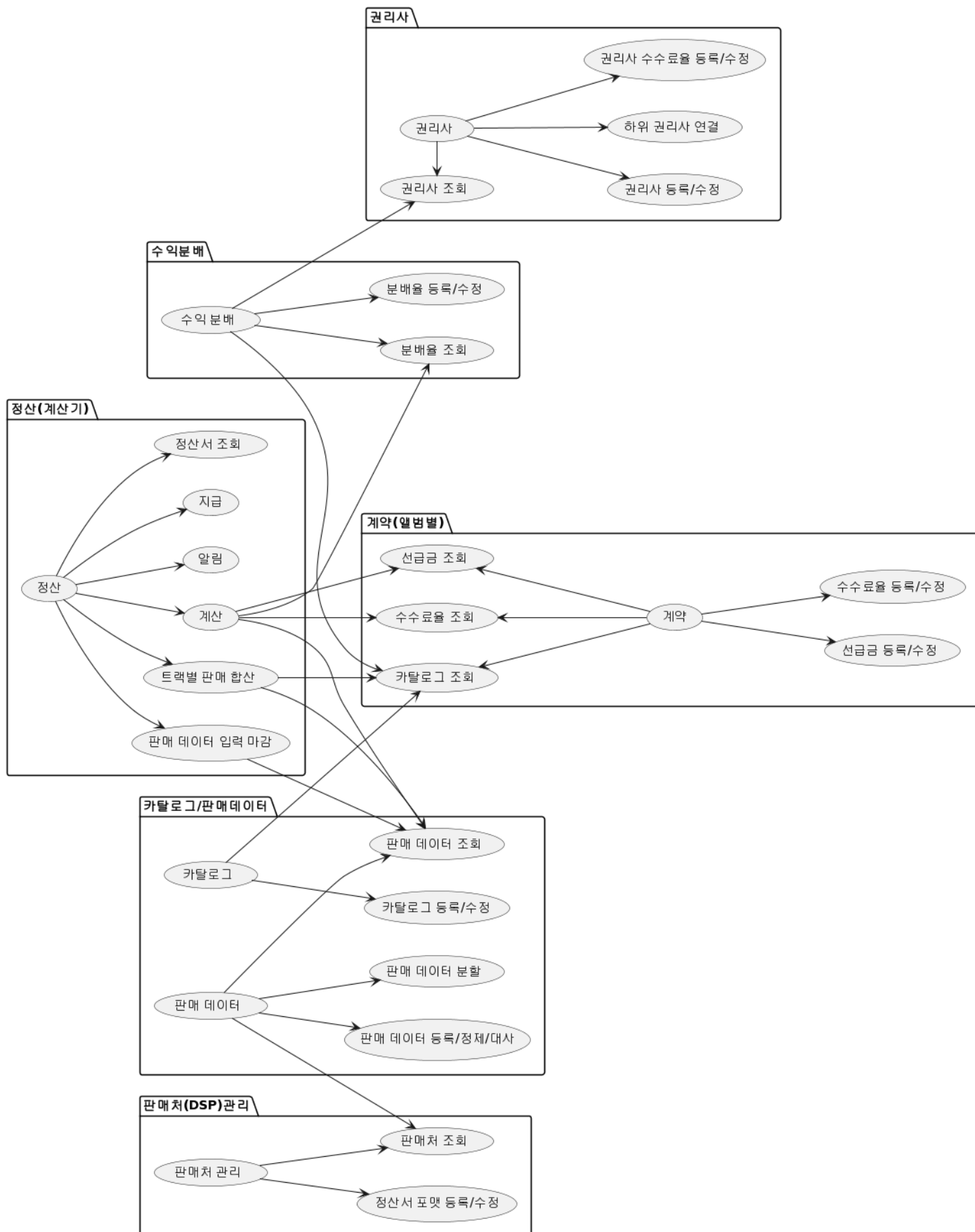
1. 관리자

a. 등록/수정

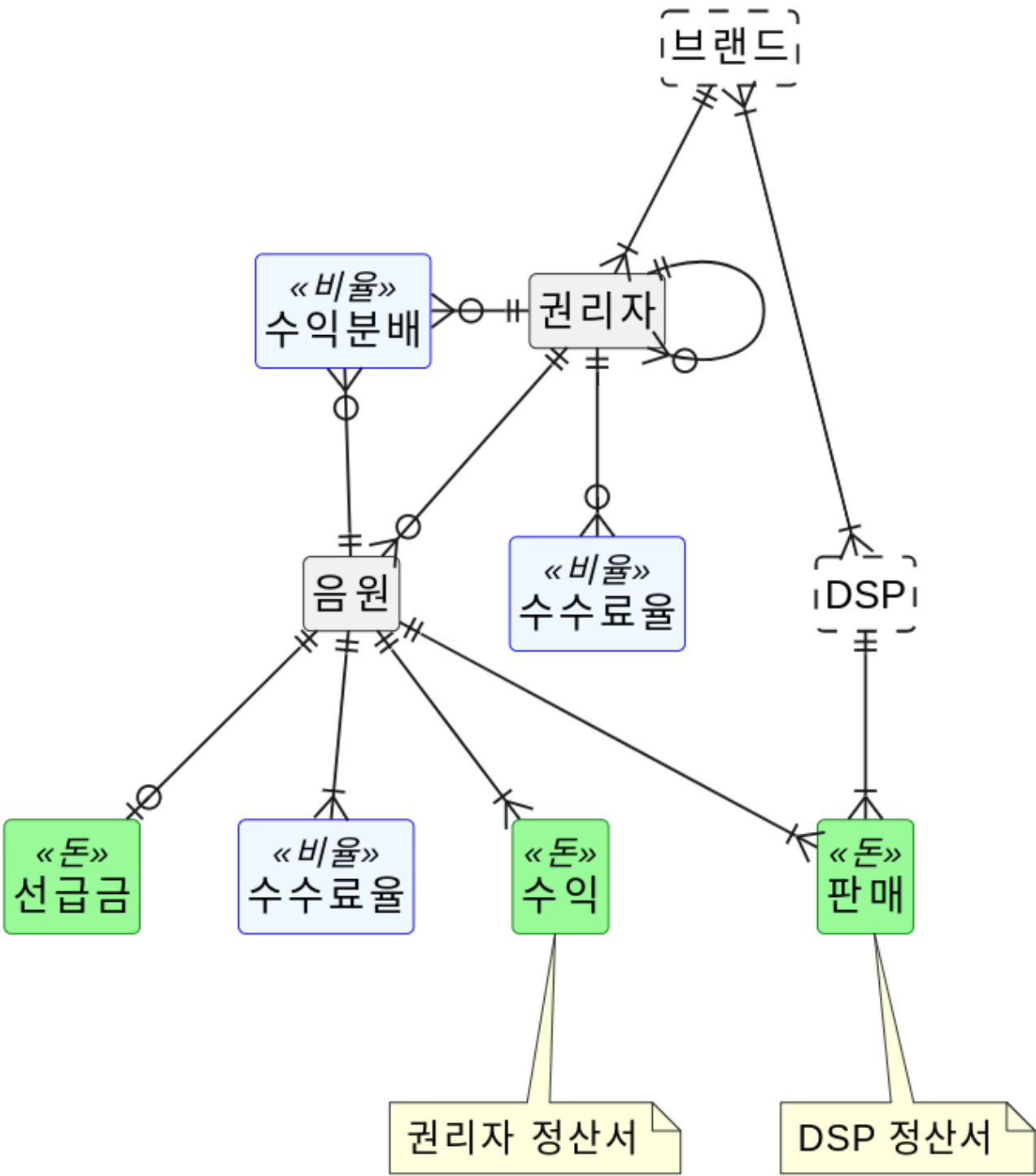
b. 조회

- c. 하위 관리자 연결
  - d. 수수료율 등록/수정
- 2. 계약(앨범별)
  - a. 카탈로그 조회
  - b. 선급금 등록/수정
  - c. 선급금 조회
  - d. 수수료율 등록/수정
  - e. 수수료율 조회
- 3. 카탈로그
  - a. 등록/수정
  - b. 조회
  - c. 관리자 조회 : 트랙과 1:1로 매핑
- 4. 수익 분배
  - a. 카탈로그 조회
  - b. 관리자 조회 : 카탈로그의 관리자 매칭과 무관함
  - c. 분배율 등록/수정
  - d. 분배율 조회
- 5. 판매 데이터(DSP 정산서) (정산월:브랜드:판매처:정산서 = 1:1:1:m 테이블 구조)
  - a. 판매처 조회
  - b. (재)등록/정제(판매데이터 누락 등)
  - c. 대사 : 불일치(매핑 안되는) 트랙, 오류(필수 누락, 수치에 문자, 관리자 매칭누락, 관리자 계좌 누락...) 식별
  - d. 판매 데이터 조회(다운로드)
- 6. 정산(계산기)
  - a. 판매 데이터 입력 마감 : 환율정보, 정산월의 판매데이터 이상있을 시 마감 실패처리
  - b. 트랙별 판매 합산
  - c. 계산 : 수수료 징수/선급 상계/이월(익월에 누적합산)/분배
  - d. 정산서 조회(다운로드) : 이체대상 목록 조회 등
- ~~7. 판매처(DSP) 관리~~
  - ~~a. 조회~~
  - ~~b. 등록/수정~~
  - ~~c. 정산서 포맷 등록/수정~~

유스케이스 다이어그램



정산시스템 도메인 모델





## 6.8. 컴포넌트

TODO 각각의 컨테이너 별로 작성

### 6.8.1. 백엔드 Back-End

#### 6.8.1.1. mybatis CRUD

1. 기본 쿼리/DTO/도메인 모델 생성에 mybatis-generator 사용 🙌 [다운로드](#)

#### 6.8.1.2. 보안

##### 1. 인증

###### a. 전제조건

- i. Spring Security Form Login 사용
- ii. 모든 컨테이너들은 같은 도메인(moundmedia.net)을 사용하여 쿠키 공유

###### b. 인증 순서

- i. front는 back-end의 form login 주소로 API호출
- ii. 인증 성공 시 JSESSIONID 쿠키값을 받음
- iii. 이후의 모든 요청 헤더에 이 쿠키를 전송(도메인이 동일하므로 요청 헤더에 명시적으로 쿠키를 set하지 않더라도 전송됨)
- iv. 세션 타임아웃시 다시 form login API를 호출하여 쿠키를 받아옴

###### c. back-end

미인증시 form로그인 화면으로 리다이렉트(http응답 302) 하지 않고 http응답 401을 반환하도록 수정. 또한 비어있는 response body를 반환해야 함

- i. ExceptionHandlingConfigurer의 AuthenticationEntryPoint 변경 : SecurityConfig.java.configure() 참고
- ii. AjaxAwareLoginUrlAuthenticationEntryPoint.java : ExceptionHandlingConfigurer에서 사용할 AuthenticationEntryPoint
- iii. 참고 🙌 [Spring Security 설정](#)

###### d. front-end

- i. API 요청시 쿠키값을 포함해야 하기 때문에 사용하는 통신객체에 Credentials 설정이 되어야 함 🙌 [CORS 쿠키 전송하기](#)
- ii. 로그인 요청시 -**Method** : Post, **Path** : /login, **Parameter** : FormData (username, password)
- iii. 로그아웃 요청시 - **Method** : Post , **Path** : /logout
- iv. back-end의 설정에 따라 path, formData의 키 이름이 달라질수 있음
- v. form login의 csrf 토큰을 값 가져올 수 없거나 사용하지 않는다면 back-end에서 HttpSecurity의 csrf().disable()가 필요 함

##### 2. SSL

###### a. AWS에서 제공하는 ACM을 통해서 SSL 등록

###### b. 적용 domain list

- i. <https://sett-alpha.moundmedia.net>
- ii. <https://sett-alpha-backend.moundmedia.net>

##### 3. 데이터 암호화

###### a. 대상 : 모든 개인식별정보

b. 방식 : AES / CBC / PKCS5Padding

c. 공통 라이브러리 사용 :

[https://gitlab.com/dx\\_team/settlement/sett-common/-/blob/8ef3ed22cf5daafd6d5d08b94225ad0ee8482c08/lib/src/test/java/com/mm/sett/comm/AES256UtilTest.java](https://gitlab.com/dx_team/settlement/sett-common/-/blob/8ef3ed22cf5daafd6d5d08b94225ad0ee8482c08/lib/src/test/java/com/mm/sett/comm/AES256UtilTest.java)

#### 6.8.1.3. 시나리오 테스트

- karate 결과 :

<http://10.1.40.53:7080/dev/sett/back-end/karate/target/karate-reports/karate-summary.html>

### 6.8.2. 파일 매니저

#### 6.8.2.1. 파일 저장 정책

TODO 결정 필요

1. 년도별로 폴더 생성?
2. 파일 종류별로 폴더 생성?
3. 저장시 이름을 저장 시점으로 변경할지? 파일 확장자는 그대로 둘지

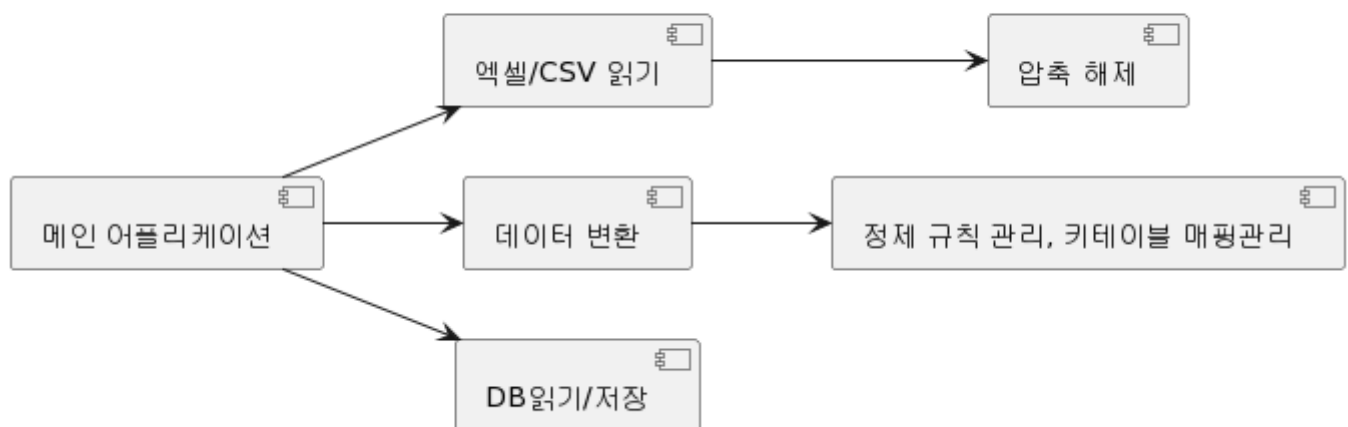
#### 6.8.2.2. 업로드

※ 대용량(유튜브 700만건)처리 필요

1. 엑셀/CSV 읽기 : 제약사항(xls처리 불가)
2. 압축 해제
3. 정제 규칙 관리, 키테이블 매핑관리
4. DB읽기/저장(원본테이블, 키테이블)

Warning:  
대외비

#### 정산시스템 컴포넌트 : 파일 매니저-업로드



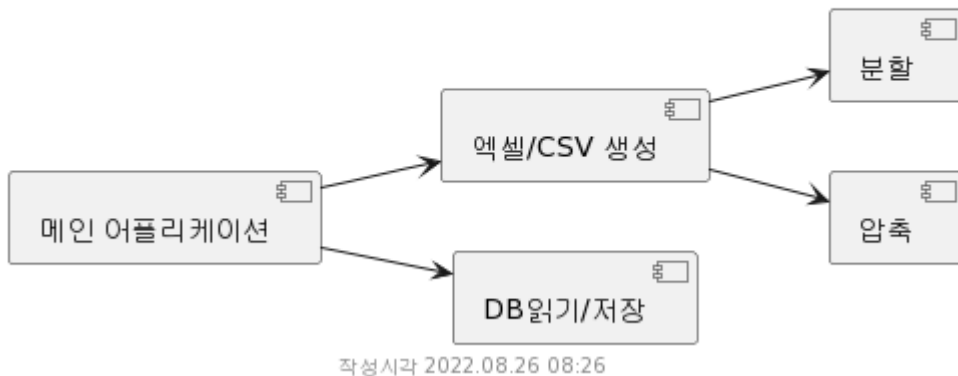
작성시각 2022.08.26 08:24

#### 6.8.2.3. 다운로드

1. 엑셀/CSV/PDF(?) 생성
2. 분할
3. 압축
4. DB읽기/저장

Warning:  
대외비

### 정산시스템 컴포넌트 : 파일 매니저-다운로드



### 6.8.3. 계산기 SDS(Sum-Divide-Subtract???)

1. 선급계약 종료 처리

### 6.8.4. 메서치 발송기 Message Sender(alpha 범위 아님)

## 6.9. 클래스 설계

배포 방안 고려 필요 : Refer to <https://co-de.tistory.com/68>

### 6.9.1. 공통 🙌 [doxygen 문서 보기](#)

1. gitlab 소스 : [https://gitlab.com/dx\\_team/settlement/sett-common](https://gitlab.com/dx_team/settlement/sett-common)
2. 정산 도메인 : com.mm.sett.domain
  - a. BaseDomain.java : 모든 데이터 모델의 최상위 조상(생성자 수정자 추가)
  - b. (Re)Seller.java : 마운드 미디어로 부터 재화를 공급받아 중개/판매하는 자
  - c. SettleYearMonth.java : 정산 년월, 정산 칼렌다
  - d. 데이터 로더 Sales Data Loader : com.mm.sett.domain.loader
    - i. SalesDataLoadJob.java : 판매 데이터 로딩 요청 도메인 객체
    - ii. SalesData.java : 판매 데이터 모델(마운드 미디어 표준)
3. 전사 공통 유틸리티 : com.mm.util
  - a. SysCmdRunner.java : 시스템 명령어 실행
4. 설정 : src/main/resources
  - a. application.yml
5. 공통 mybatis
  - a. 현재 CommFileMapper 만 모듈화 되어 있음
  - b. 설정
    - i. 프로젝트의 mapper-locations 설정의 classpath에 \* 를 넣기

🙌 참고 : <https://stackoverflow.com/a/22801484/4766882>

mybatis:

mapper-locations: **classpath\***:/mybatis/mapper/\*.xml

ii. MyBatisConfig 추가 🙌 참고

[https://gitlab.com/dx\\_team/settlement/back-end/-/blob/c12584777d7e211c97a10f3efc3214d3e4d38a68/src/main/java/com/mm/sett/api/conf/MyBatisConfig.java](https://gitlab.com/dx_team/settlement/back-end/-/blob/c12584777d7e211c97a10f3efc3214d3e4d38a68/src/main/java/com/mm/sett/api/conf/MyBatisConfig.java)

## 6.9.2. 백엔드 🙌 doxygen 문서 보기

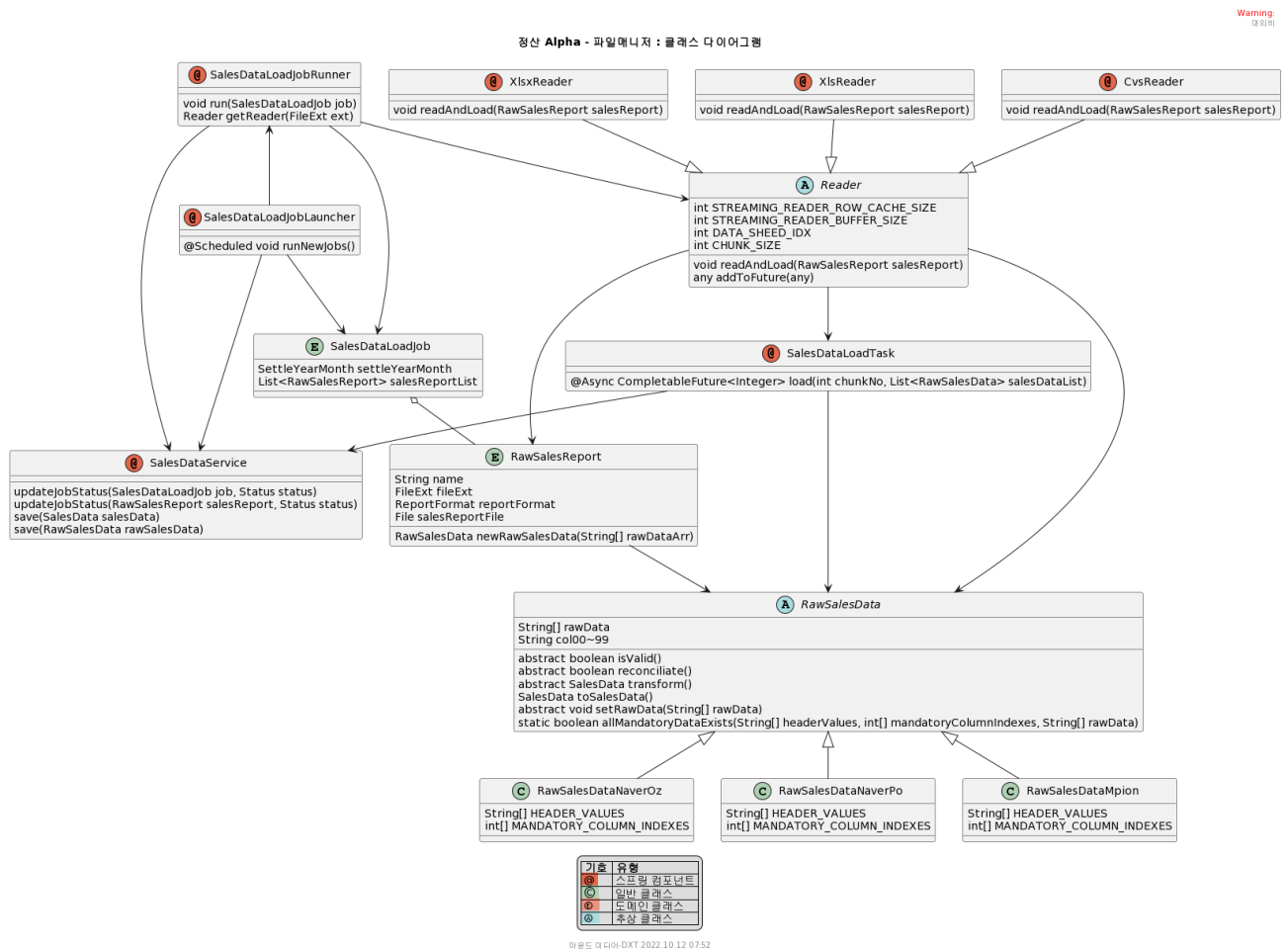
1. gitlab 소스 : [https://gitlab.com/dx\\_team/settlement/back-end](https://gitlab.com/dx_team/settlement/back-end)

## 6.9.3. 파일 매니저 🙌 doxygen 문서 보기

1. gitlab 소스 : [https://gitlab.com/dx\\_team/settlement/file-manager](https://gitlab.com/dx_team/settlement/file-manager)
2. javadoc pdf : [정산Alpha-FileManager-JavaDoc\\_230914.pdf](#)

### 6.9.3.1. 로더

1. 클래스 다이어그램

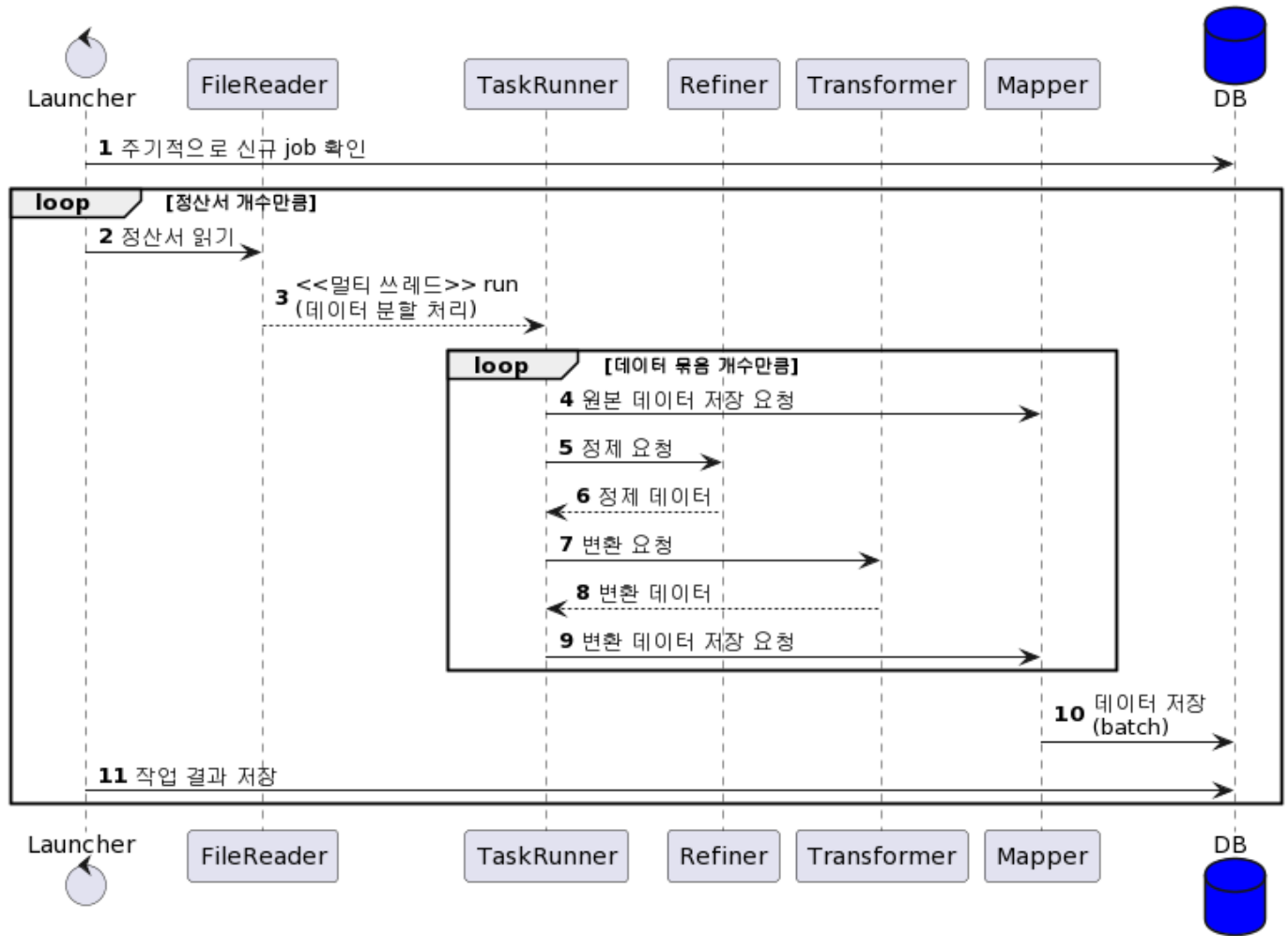


## 2. 메인 애플리케이션

- a. SalesDataLoader : Spring Boot Application
- b. AppProperties : 앱 환경 설정
- c. ConfigNoneClassBody : 클래스 선언이 따로 필요없는 설정
- d. SalesDataLoadJobLauncher : 주기적으로 SalesDataLoadJob를 확인하고 신규 job이 있다면 처리
- e. SalesDataLoadJob : 판매 데이터 로딩 요청 도메인 객체
- f. SalesDataLoadTask : 판매 데이터 로딩 태스크. 일정 크기의 chunk 데이터를 받아서 처리

- g. SalesDataLoadTaskRunner : 판매 데이터 로딩 job 실행.
- 3. 엑셀/CSV 읽기 : 데이터를 묶어서(chunk) SalesDataLoadTask에 전달(멀티 쓰레드)
  - ※ 소스 복잡도를 줄이기 위해 reader가 loader를 바라보도록 함(설계상으로 좋지 않지만)
  - ※ 참고 : [DXT Settlement System IA, 엑셀 컬럼별 인덱스](#)
    - a. ExcelReader
    - b. CsvReader
    - c. RawSalesReport : 셀러로부터 받은 판매 보고서(정산서)
    - d. RawSalesData.java : 셀러로부터 받은 원본 데이터 인터페이스, toSalesData()가 정의되어 있음
    - e. RawSalesData~~~ : 각각 정산서별로 정제/변환 로직 개발. ex) RawSalesDataNaver.java
- ~~4. 정제 : 무의미한 데이터 삭제, 일부 하자가 있는 데이터를 이후 처리 편의를 위해 값 보장~~
  - ~~a. Filter : 유의미한 데이터만 남기기~~
  - ~~b. Refiner : 정제~~
  - ~~c. RuleManager : 정산서별 정제 규칙 관리~~
  - ~~d. Reconciliator : 대서, 보장~~
- ~~5. 변환~~
  - ~~a. Transformer : 정산서별 판매 데이터를 마운드 미디어 표준 형식으로 변환~~
- 6. DB저장
  - ※ 성능향상을 위해 [batch insert](#) 필요
    - a. SalesDataService : sql 매퍼 호출
    - b. SalesDataMapper : Mybatis sql 매퍼
    - c. ~~TypeHandler : 테이블 컬럼에 맞는 형식으로 변환

데이터 로더 Sales Data Loader 시퀀스 다이어그램



작성시각 2022.08.25 06:17

### 6.9.3.2. 파일 생성

1. 메인 애플리케이션
  - a. ReportFileCreator : Spring Boot Application
  - b. AppProperties : 앱 환경 설정
  - c. ConfigNoneClassBody : 클래스 선언이 따로 필요없는 설정
  - d. ReportFileCreateJobLauncher : 주기적으로 ReportFileCreateJob를 확인하고 신규 job이 있다면 처리
  - e. ReportFileCreateJob : 보고서 생성 요청 객체
  - f. ReportFileCreateTask : 보고서 데이터 로딩 태스크. 사전 정의 된 크기의 chunk 데이터 단위로 처리
  - g. ReportFileCreateTaskRunner : 보고서 데이터 로딩 job 실행. 멀티 쓰레드로 실행됨
2. 보고서 데이터 읽기
  - a. ReportService : sql 매퍼 호출
  - b. ReportMapper : Mybatis sql 매퍼
3. 파일 생성 Writer
  - a. ExcelWriter : 엑셀 형식 보고서 저장
  - b. CsvWriter : CSV 형식 보고서 저장

## 6.9.4. 계산기 SDS(Sum-Divide-Subtract)

### 6.9.4.1. 업무

1. 실행 기준
  - a. QUEUE\_CALCULATE\_MANAGER
2. 사전 작업
  - a. 분배를 위한 계약 현행 정보 정리
  - b. 분배를 위한 asset, asset item별 수수료(계약 수수료 포함) 정보 정리
  - c. 분배를 위한 asset, asset item별 분배율 정보 정리
3. 정산 금액 계산
  - a. 수수료, 선급금 상계 실행 금액 계산
4. 분배 금액 계산
  - a. 수수료 차감 후 금액을 분배율 기반으로 정산액 분배
5. 사후 작업
  - a. 선급계약 종료 처리

## 6.10. RESTful API

### 6.10.1. 규약

1. 등록,수정,삭제의 경우 request header의 reqAccountId에 작업자의 아이디를 넣어야 함
2. enum 목록 : 값으로 code를 사용하지 않고 enumName을 사용(code는 DB저장용임)
  - a. 조회 : <http://10.1.40.53:9111/v0.1/v3/api-docs>
  - b.

### 6.10.2. 정산 API 목록

※ [정산 API 목록](#) 참조

## 6.11. 데이터 유효성 검사

※ 컨테이너들이 모델 정의를 공유하여 일관성을 유지하면 개발 공수를 줄일 수 있음.

1. back-end에서 입력 DTO들에 대한 정의(항목별 유형, 유효성 기준 등...)를 제공 🙌 swagger를 이용하면 class에서 자동 생성됨 ex) <http://10.1.40.53:9121/v0.1/v3/api-docs>

ex) 고객 샘플 데이터

```
{
  "taxId": "KO_a1234",
  "countryCode": "KO",
  "email": "pororo@kid.com",
  "age": 12,
  "grade": "A",
```

```

"feeRate": 1.23,
"expireDate": "2023-09-29",
"regNo": "830422-1185600",
"phones": [
  "111-222-3333",
  "444-555-6666"
]
}

```

ex) 고객 모델 정의

```

{
  "title": "고객",
  "required": [
    "countryCode",
    "phones",
    "taxId"
  ],
  "type": "object",
  "properties": {
    "taxId": {
      "title": "납세자코드",
      "maxLength": 10,
      "minLength": 3,
      "type": "string",
      "description": "국가코드로 시작해야 함"
    },
    "countryCode": {
      "title": "국가코드",
      "maxLength": 2,
      "minLength": 2,
      "type": "string",
      "description": "ISO 3166"
    },
    "email": {
      "title": "이메일",
      "type": "string"
    },
    "age": {
      "title": "나이",
      "maximum": 99,
      "minimum": 10,
      "type": "integer",
      "format": "int32"
    },
    "grade": {

```



```

        "title": "등급",
        "type": "string"
    },
    "feeRate": {
        "title": "수수료율",
        "type": "number",
        "format": "float"
    },
    "expireDate": {
        "title": "만기일",
        "type": "string",
        "description": "오늘 이후 날짜",
        "format": "date"
    },
    "regNo": {
        "title": "주민등록번호",
        "pattern":
"^((?:[0-9]{2}(?:0[1-9]|1[0-2])(?:0[1-9]|1[1,2][0-9]|3[0,1]))-[1-4][0-9]{6}$",
        "type": "string"
    },
    "phones": {
        "title": "전화번호",
        "type": "array",
        "description": "목록",
        "items": {
            "title": "전화번호",
            "type": "string",
            "description": "목록"
        }
    }
}

```

2. front-end 측 오류 처리 : DTO정의를 이용하여 JSON 데이터를 생성하고 유효성 체크 수행
  3. back-end 측 오류 처리 : javax.validation으로 검사하고 오류 시 해당 HTTP 상태코드와 오류 정보 제공
- ex)

```

{
    "code": "BINDING_ERROR",
    "status": 500,
    "detail": "Spring MVC Binding 오류",
    "instance": "uri=/dev/postCustomerTest",
    "errors": [
        {
            "codes": [

```

```

        "Size.customer.countryCode",
        "Size.countryCode",
        "Size.java.lang.String",
        "Size"
    ],
    "arguments": [
        {
            "codes": [
                "customer.countryCode",
                "countryCode"
            ],
            "arguments": null,
            "defaultMessage": "countryCode",
            "code": "countryCode"
        },
        2,
        2
    ],
    "defaultMessage": "크기가 2에서 2 사이여야 합니다",
    "objectName": "customer",
    "field": "countryCode",
    "rejectedValue": "Korea",
    "bindingFailure": false,
    "code": "Size"
},
{
    "codes": [
        "Max.customer.age",
        "Max.age",
        "Max.java.lang.Short",
        "Max"
    ],
    "arguments": [
        {
            "codes": [
                "customer.age",
                "age"
            ],
            "arguments": null,
            "defaultMessage": "age",
            "code": "age"
        },
        99
    ],
    "defaultMessage": "99 이하여야 합니다",
    "objectName": "customer",

```

```

    "field": "age",
    "rejectedValue": 9999,
    "bindingFailure": false,
    "code": "Max"
  },
  {
    "codes": [
      "Digits.customer.feeRate",
      "Digits.feeRate",
      "Digits.java.lang.Float",
      "Digits"
    ],
    "arguments": [
      {
        "codes": [
          "customer.feeRate",
          "feeRate"
        ],
        "arguments": null,
        "defaultMessage": "feeRate",
        "code": "feeRate"
      },
      2,
      2
    ],
    "defaultMessage": "숫자 값이 한계를 초과합니다(<2 자리>.<2 자리> 예상)",
    "objectName": "customer",
    "field": "feeRate",
    "rejectedValue": 1.234,
    "bindingFailure": false,
    "code": "Digits"
  },
  {
    "codes": [
      "AssertTrue.customer.validPhones",
      "AssertTrue.validPhones",
      "AssertTrue.boolean",
      "AssertTrue"
    ],
    "arguments": [
      {
        "codes": [
          "customer.validPhones",
          "validPhones"
        ],
        "arguments": null,

```

```

        "defaultMessage": "validPhones",
        "code": "validPhones"
    },
    ],
    "defaultMessage": "비즈니스 로직이 적용된 유효성 검사. 비어있는 전화번호가
있습니다.",
    "objectName": "customer",
    "field": "validPhones",
    "rejectedValue": false,
    "bindingFailure": false,
    "code": "AssertTrue"
},
{
    "codes": [
        "Pattern.customer.regNo",
        "Pattern.regNo",
        "Pattern.java.lang.String",
        "Pattern"
    ],
    "arguments": [
        {
            "codes": [
                "customer.regNo",
                "regNo"
            ],
            "arguments": null,
            "defaultMessage": "regNo",
            "code": "regNo"
        },
        [],
        {
            "arguments": null,
            "defaultMessage":
"^(:?[0-9]{2}(:?0[1-9]|1[0-2])(?:0[1-9]|[1,2][0-9]|3[0,1]))-[1-4][0-9]{6}$",
            "codes": [
"^(:?[0-9]{2}(:?0[1-9]|1[0-2])(?:0[1-9]|[1,2][0-9]|3[0,1]))-[1-4][0-9]{6}$"
            ]
        }
    ],
    "defaultMessage": "주민등록번호 형식이 아닙니다",
    "objectName": "customer",
    "field": "regNo",
    "rejectedValue": "521202-4562181-아기공룡둘리둘리",
    "bindingFailure": false,
    "code": "Pattern"
}

```

```

    },
    {
        "codes": [
            "AssertTrue.customer.validTaxId",
            "AssertTrue.validTaxId",
            "AssertTrue.boolean",
            "AssertTrue"
        ],
        "arguments": [
            {
                "codes": [
                    "customer.validTaxId",
                    "validTaxId"
                ],
                "arguments": null,
                "defaultMessage": "validTaxId",
                "code": "validTaxId"
            }
        ],
        "defaultMessage": "비즈니스 로직이 적용된 유효성 검사. 납세자코드는  
국가코드로 시작해야 합니다",
        "objectName": "customer",
        "field": "validTaxId",
        "rejectedValue": false,
        "bindingFailure": false,
        "code": "AssertTrue"
    },
    {
        "codes": [
            "Future.customer.expireDate",
            "Future.expireDate",
            "Future.java.time.LocalDate",
            "Future"
        ],
        "arguments": [
            {
                "codes": [
                    "customer.expireDate",
                    "expireDate"
                ],
                "arguments": null,
                "defaultMessage": "expireDate",
                "code": "expireDate"
            }
        ],
        "defaultMessage": "미래 날짜여야 합니다",

```

```

    "objectName": "customer",
    "field": "expireDate",
    "rejectedValue": "2020-10-04",
    "bindingFailure": false,
    "code": "Future"
  }
]
}

```

#### 4. DTO 정의 동기화 필요(back-end ➡ front-end)

※ 이슈 : 화면 전환시 마다 동기화 하면 부하가 많이 발생하므로 갱신되는 경우만 동기화

- back-end : api의 응답헤더에 api 버전 번호를 넣음.
- front-end : 모델정의를 가져옴
- front-end : api호출 시 응답헤더의 api 버전 번호를 저장. 이 정보가 이전의 버전 번호와 틀리다면 모델정의를 다시 가져옴

#### 5. 비즈니스로직의 오류가 있는 경우

ex) 특정 문자열로 시작해야 하는 필드인 경우

```

{
  "codes": [
    "AssertTrue.customer.validTaxId",
    "AssertTrue.validTaxId",
    "AssertTrue.boolean",
    "AssertTrue"
  ],
  "arguments": [
    {
      "codes": [
        "customer.validTaxId",
        "validTaxId"
      ],
      "arguments": null,
      "defaultMessage": "validTaxId",
      "code": "validTaxId"
    }
  ],
  "defaultMessage": "비즈니스 로직이 적용된 유효성 검사. 납세자코드는 국가코드로 시작해야 합니다",
  "objectName": "customer",
  "field": "validTaxId",
  "rejectedValue": false,
  "bindingFailure": false,
  "code": "AssertTrue"
}

```

- code : AssertTrue
- field: "valid" + 해당 컬럼명 ex) "field": "validTaxId"
- defaultMessage : 오류 메시지

d. Java 소스 샘플

```
/**
 * @AssertTrue로 유효성 검사를 하기 위한 메소드
 * 반드시 is로 시작(그래야 유효성 검사가 실행)
 */
@Hidden // swagger 문서화 대상이 아님
@JsonIgnore // json응답 대상이 아님
@AssertTrue(message="비즈니스 로직이 적용된 유효성 검사. 납세자코드는
국가코드로 시작해야 합니다")
public boolean isValidTaxId() {
    return taxId.startsWith(countryCode);
}
```

6. 코드(enum) 정의 api : <http://10.1.40.53:9111/v0.1/comm/enums>

※ api-docs의 .components.schemas에도 정의되어 있지만 코드명이 출력되지 않아서 따로 api를 만들

7. 참고

- <https://kapentaz.github.io/java/Java-Bean-Validation-%EC%A0%9C%EB%8C%80%EB%A1%9C-%EC%95%8C%EA%B3%A0-%EC%93%B0%EC%9E%90/#>
- <https://reflectoring.io/bean-validation-with-spring-boot/>
- <https://reflectoring.io/bean-validation-anti-patterns/#anti-pattern-3-using-validation-groups-for-use-case-validations>
- 

## 6.12. DB

### 6.12.1. fractional seconds rounding problem

1. 증상 : Java에서 종료일을 저장하기 위해 그날의 마지막 시각을 사용하고 있음

```
/**
 * endTime은 항상 그날의 마지막 시각으로 변경됨
 * ex)2022-10-05T06:51:20 -> 2022-10-05T23:59:59.999999999
 * @param endTime
 */
public void setEndTime(@NotNull LocalDateTime endTime) {
    //2022-10-05T06:51:20 -> 2022-10-05T23:59:59.999999999
    this.endTime = LocalDateTime.of(endTime.toLocalDate(), LocalTime.MAX);
}
```

하지만 DB에서 반올림 처리 되어 다음날로 저장됨

2. 해결 : jdbc url에 인수 추가

```
url:jdbc:mysql:// ~ ?sendFractionalSeconds=false
```

※참고 : <https://lenditkr.github.io/MySQL/fractional-seconds-rounding-problem/>

- 3.

## 7. 단어 사전

※ DDEX 참고

- 단어 인덱스 : <http://service.ddex.net/dd/MasterIndex/index.html>
- 용어집 : <https://kb.ddex.net/display/HBK/Definitions+and+Terminology>
- 세무/회계 용어집 : <https://www.samili.com/acc/ifrsdic.asp>

### 7.1. 금액

특정 권리자의 특정 정산월의 ...

단어	설명
판매 금액(Sale Amount)	에셋별 판매액 총합(중개사들로부터 수급한 정산자료에서 취합)
정산 총액(Profit Amount)	판매액 - 수수료
선급 금액(Prepaid Amount)	권리자에게 지급된 금액. 추후 정산액에서 차감
선급 상계 금액(Prepaid Return Amount)	선급액 중 수익금에서 차감할 금액 = 상계 잔액 - 정산 총액
최종 정산금(Final Profit Amount)	정산 총액 - 선급 상계 금액
분배 수령액(Revenue Receipt)	타 권리자로 부터 수익 분배받은 금액 = 최종 정산금 - 분배 차감액
분배 차감액(Revenue Payment Amount)	타 권리자에게 수익 분배해준 금액
지급 총액(Payment Amount)	최종 정산금 + 분배 수령액 + 전기 이월 - 차기 이월
차기 이월(Carried-over Amount)	어떤 사유로 인해 현재 달에 지급되지 못한 지급 총액
전기 이월(Forwarded Amount)	어떤 사유로 인해 이전 달에 지급되지 못한 지급 총액
실 지급액(Transfer Amount)	송금할 금액(세후 금액, 지급 총액 ± 세금) - 원단위 절사
최소 입금금액(Minimum Transfer Amount)	이 금액 이하의 지급 총액은 이월한다.
수수료(Fee)	판매액에서 차감할 금액. 판매액에 수수료율을 곱해서 산정. 합치면 브랜드의 수익이 됨
부가가치세(VAT)	<b>Value Added Tax</b> {최종 정산금} X {수익분배율} X {입력 세율}의 원단위 절사
소득세(Income Tax)	{최종 정산금} X {수익분배율} X {입력 세율 X 91%}의 원단위 절사
주민세(Resident Tax)	해당 지역에 거주하는 세대주, 사업소를 둔 법인 및 개인사업자에게 부과하는 세금 {최종 정산금} X {수익분배율} X {입력 세율 X 9%}의 원단위 절사



## 7.2. 그 외

단어	설명
IP assets	지적 자산(물리적 자산을 제외한 모든 것)
Account	
<a href="#">Settlement</a>	정산, 판매사로부터 받은 상품별 판매자료를 취합하여 수수료를 차감한 금액(로열티)을 권리자에게 지급하는 과정
Asset	창작물, 저작권, 용역(공연, 출연...)
Author	원작자
Brand (POC)	마운드 미디어 내부의 판매 조직
Catalog	artwork 목록, 유통(판매)의 대상이 되는 상품
Common Works Registration (CWR)	a standard format for the registration and revision of musical works
Copyright	복제권
DSP	Digital Service Provider. 판매사. 정산 시스템에서는 중개사의 의미로도 쓰임 이 때는 판매데이터를 Mound Media에 제공하는 자의 의미임.
Fee	수수료, 유통에 대한 대가로 판매사가 징수하는 수익(margin/cost)
Fee rate	수수료율
License	상품 유통에 대한 권리를 주는 행위. 이에 대한 대가로 royalty를 지불함
Licensee	라이선스를 받은 자
Licensor	라이선스를 제공한 자
Master use license	
Mechanical license	
Mechanical royalties	음악이 재생산(복제) 될 때마다 발생
Neighbouring rights	저작인접권: 곡을 연주하면서 나오는 권리, 연주자, 가수, 음반 제작자의 권리(미국에서는 이것을 인정하지 않음)
Originator	창작자
Ownership share	
Performance royalties	음악이 공연 될 때마다 발생

Performing Rights Organizations	창작자와 출판업자를 대신해서 사용처에서 라이선스 수입을 모아주는 단체
pitching(song)	수익을 목적으로 업계 관계자들에게 하는 홍보행위 playing songs for publishers, artists or record label people in hopes of getting them to record or help you get a song recorded
product	음원, 음반 등
recoupment	선급공제
revenue	수익
reseller	리셀러
right	권리
rights controllers	저작권 협회, 출판사, 음반사, Music Licensing Companies
rights holder	권리자
royalty (accrual) report	정산(보고)서. sales report와는 다른 개념
royalty(licensing fee)	asset을 사용한 대가. 일반적으로 사용(판매)량revenue에 대한 비율로 산정
sales report	판매 데이터, 상품을 언제 얼마에 몇 건 팔았는지에 대한 정보. royalty report와는 다른 개념
sync placements	영상물에 삽입된 곡(음악이 영상에 sync 맞춰야 함)
vendor	판매처, 셀러(ex. DSP, 중개사)
reconciliation	대사(대조 확인)
Exchange rate	환율
carry over	이월
send money	송금 / 예금

## 8. 기타

### 1. [정산alpha 개발 회고](#)