```java
private List<MotionConfigDTO> getMotionConfig(String cameraId, List<SettingIVADTO> settingIVADTOS,
                                              List<ConfigSaveCameraDTO> configSaveCameraDTOS,
                                              List<TimeRecordDTO> timeRecordDTOS,
                                              List<SourceCameraDTO> sourceCameraDTOS,
                                              SubDeviceDTO subDeviceDTO,
                                              List<OutputCameraDTO> outputCameraDTOs) {
        MotionConfigDTO dto = new MotionConfigDTO();
        SettingIVADTO motionIVA = settingIVADTOS.stream().filter(x ->
                x.getAnalysisEvent() != null && x.getAnalysisEvent().getIdIVA() == 3 && x.getSubDeviceId().equals(cameraId)
        ).findFirst().orElse(null);

        if (motionIVA == null) {
            return null;
        }
        dto.setType(Constants.EVENT_TYPE.MOTION);
        dto.setEnable(motionIVA.getIsMove() == null ? 0 : motionIVA.getIsMove());
        IVAConfigDTO configDTO = new IVAConfigDTO();
        configDTO.setAlarmSystem(motionIVA.getAlarmSystem());
        configDTO.setBetweenAnalyzes(motionIVA.getBetweenAnalyzes());
        configDTO.setIsKeyFrame(motionIVA.getIsKeyFrame());
        configDTO.setIsMove(motionIVA.getIsMove() == null ? 0 : motionIVA.getIsMove());
        configDTO.setIsSaveAnalysis(motionIVA.getIsSaveAnalysis());

        configDTO.setResolution(motionIVA.getResolution());

        configDTO.setUnitFrame(motionIVA.getUnitFrame());


        SourceCameraDTO sourceCameraDTO = sourceCameraDTOS.stream().filter(x ->
x.getObjectId().equals(motionIVA.getSourceCameraId())).findFirst().orElse(null);

        if (sourceCameraDTO != null) {

            OutputCameraDTO outputCameraDTO = outputCameraDTOs.stream().filter(x ->
x.getSourceCameraId().equals(sourceCameraDTO.getObjectId())).findFirst().orElse(new OutputCameraDTO());

            configDTO.setChannelId(subDeviceDTO.getDeviceDTO().getMacAddress().replace(":", "").replace("-", "") + Constants.XYZ +
outputCameraDTO.getId());

        }


        if (motionIVA.getTimeRecord() != null) {

            List<TimeProfileDTO> timeProfileDTOS = new ArrayList<>();

            for (TimeRecordDTO timeRecordDTO : motionIVA.getTimeRecord()) {

                TimeProfileDTO timeProfile = createTimeRecord(timeRecordDTO);

                timeProfileDTOS.add(timeProfile);

            }

            if (motionIVA.getZoneConfig() != null && motionIVA.getZoneConfig().getDetectArea() != null) {

                List<AreaDTO> areaDTOS = new ArrayList<>();

                ZoneConfigDTO zoneConfigDTO = motionIVA.getZoneConfig();
                for (DetectAreaDTO areaDTO : zoneConfigDTO.getDetectArea()) {
                    AreaDTO area = new AreaDTO();
                    area.setType(zoneConfigDTO.getZoneSetting());
                    area.setX(convertAreaNumber(areaDTO.getX1(), zoneConfigDTO.getWidthDetectArea()));
                    area.setY(convertAreaNumber(areaDTO.getY1(), zoneConfigDTO.getHeightDetectArea()));
                    area.setW(convertAreaNumber(areaDTO.getX2(), zoneConfigDTO.getWidthDetectArea()));
                    area.setH(convertAreaNumber(areaDTO.getY2(), zoneConfigDTO.getHeightDetectArea()));
                    area.setOpacity(motionIVA.getZoneConfig().getOpacity());
                    areaDTOS.add(area);
                }
                configDTO.setAreas(areaDTOS);
            }
            configDTO.setTimeRecord(timeProfileDTOS);
            dto.setConfig(configDTO);
        }

        ConfigSaveCameraDTO configSave = configSaveCameraDTOS.stream().filter(x
                -> x.getSourceCameraId().getObjectId().equals(motionIVA.getSourceCameraId()) && x.getIsEvent() == Constants.EVENT_TYPE &&
x.getEventName().equals(Constants.EVENT_TYPE.EVENT_MOTION)).findFirst().orElse(null);

        if (configSave != null) {
```

```
StorageDTO storageDTO = new StorageDTO();
storageDTO.setEnable(Constants.ACTIVE);
Integer retention = Integer.parseInt(configSave.getTimeSave());
if (configSave.getTimeSaveValue().equals(Constants.TIME_PROFILE.DAY)) {
    retention = retention * 24;
    storageDTO.setRetention(retention.toString());
}
if (configSave.getTimeSaveValue().equals(Constants.TIME_PROFILE.MONTH)) {
    retention = retention * 24 * 30;
    storageDTO.setRetention(retention.toString());
}
if (configSave.getTimeSaveValue().equals(Constants.TIME_PROFILE.YEAR)) {
    retention = retention * 24 * 365;
    storageDTO.setRetention(retention.toString());
}
storageDTO.setAfter(configSave.getAfterTimeEvent());
storageDTO.setBefore(configSave.getBeforeTimeEvent());
TimeRecordDTO timeRecordDTO = timeRecordDTOS.stream().filter(x ->
x.getId().equals(configSave.getRecordEventId())).findFirst().orElse(null);
if (timeRecordDTO != null) {
    TimeProfileDTO timeProfile = createTimeRecord(timeRecordDTO);
```

```
            timeProfileDTOS.add(timeProfile);
        }
        dto.setStorage(storageDTO);
    }
    return Arrays.asList(dto);
}
        private TimeProfileDTO createTimeRecord(TimeRecordDTO timeRecordDTO){
                    TimeProfileDTO timeProfile = new TimeProfileDTO();
            timeProfile.setActiveDate(timeRecordDTO.getStartActive().getTime());
            timeProfile.setExpireDate(timeRecordDTO.getEndActive().getTime());
            timeProfile.setTimeZone(Constants.TIME_ZONE_7);
            int type = timeRecordDTO.getCycleTimes().get(0).getType();
            timeProfile.setType(type);
            if (type == Constants.TIME_PROFILE.MONTH || type == Constants.TIME_PROFILE.WEEK) {
                List<Integer> dateList = new ArrayList<>();
                for (CycleTimeDTO cycleTimeDTO : timeRecordDTO.getCycleTimes()) {
                    dateList.add(Integer.parseInt(cycleTimeDTO.getValue()));
                }
                timeProfile.setDateList(dateList);
            }
            if (timeRecordDTO.getPeriodRecords() != null) {
```

```
     List<IntervalDTO> intervalDTOS = new ArrayList<>();
     for (PeriodRecordDTO periodRecordDTO : timeRecordDTO.getPeriodRecords()) {
         IntervalDTO intervalDTO = new IntervalDTO();
         intervalDTO.setStartTime(Utils.convertDateToSecond(periodRecordDTO.getStartTime()));
         intervalDTO.setEndTime(Utils.convertDateToSecond(periodRecordDTO.getEndTime()));
         intervalDTOS.add(intervalDTO);
     }
     timeProfile.setIntevalList(intervalDTOS);
 }
           return timeProfile;
}
```