LLDD BE - Job 8 CreateCompensationDocument

SBP Mall - ระบบประกันรายได้ | Low Level Design Document

แนวทางการใช้เอกสาร

หัวข้อ	คำอธิบาย
วัตถุประสงค์	ใช้เป็นรายละเอียดระดับพัฒนาสำหรับออกแบบ ลงมือพัฒนา ตรวจทาน และทดสอบขอบเขตที่ระบุในฉบับนี้
ลำดับการอ่าน	เริ่มจาก Overview และ Scope จากนั้นตรวจ Field/Validation, Implementation, Contract, Processing Flow, Acceptance Criteria และ Test Checklist ตามลำดับ
ผลลัพธ์ที่คาดหวัง	ผู้อ่านสามารถระบุ input, ขั้นตอนประมวลผล, output, เงื่อนไขผิดพลาด และหลักฐานการทดสอบได้จากเนื้อหาในฉบับเดียว
Java เดิม	ภาคผนวกท้ายฉบับระบุ source file, ช่วงบรรทัด และ code Java เดิมสำหรับตรวจเทียบ behavior ก่อนย้ายระบบ

คำว่า Input, Progress และ Output ในเอกสารนี้หมายถึงข้อมูลตั้งต้น ลำดับการทำงาน และผลลัพธ์ที่ตรวจสอบได้ของขอบเขตที่กำลังพัฒนา

1. Overview

รายการ	รายละเอียด
Track	BE
Estimate	24 ชั่วโมง = implementation 18 + unit test 6 (30%)
Owner	Aphiwit <Bank> Khammoon
Target repository	`SBP/srm-sps-spsap-store-backend` (NestJS + TypeORM · schema `sps_store`) — batch runner ฝั่ง backend ไม่ผ่าน BFF · cron/พารามิเตอร์อยู่ใน backend config (env/config file)
Objective	สร้างเอกสารประกันรายได้อัตโนมัติ: สร้าง compensation_documents จาก impact profile และข้อมูลชดเชยในฐานข้อมูลเดียวกัน แทนการเขียนไฟล์ BPM06001O และ SFTP ไป compensateflow; ไม่เรียก workflow โดยตรง

Common contract reference: ทุกหัวข้อ API/FE ต้องยึด LLDD-BE-API-Common-Contracts.pdf และ LLDD-FE-Integration-Contracts.pdf สำหรับ error/auth/format/pagination/action/RBAC ก่อนลงรายละเอียดเฉพาะหน้าหรือเฉพาะ endpoint

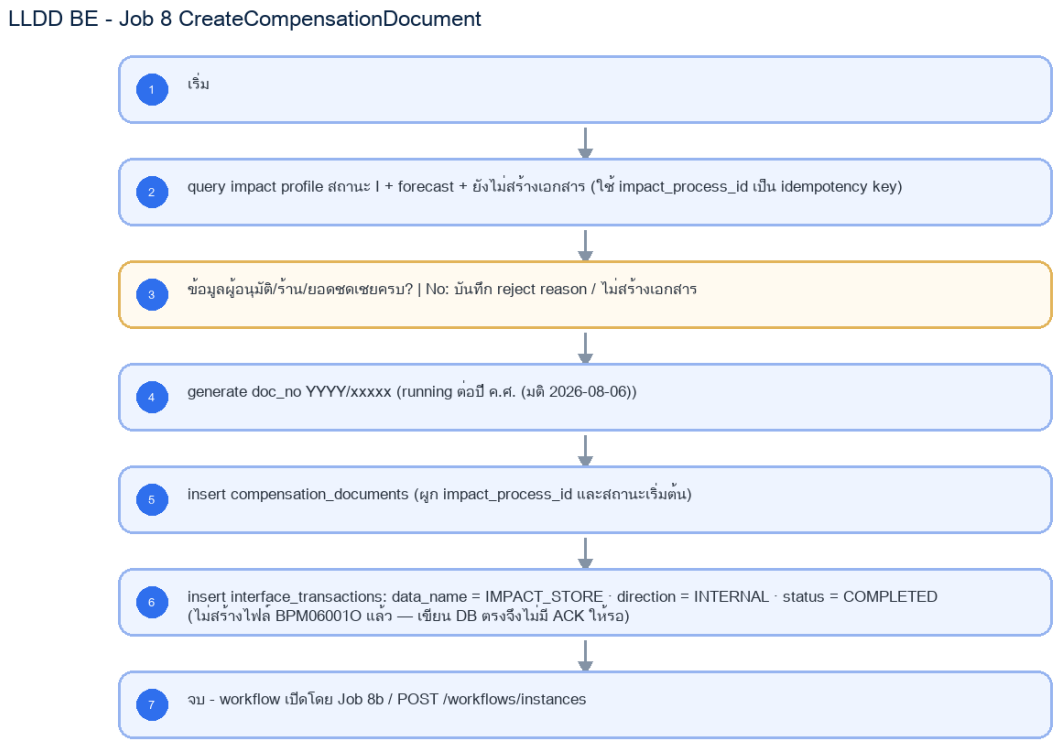
2. Screen / Functional Scope

- Main class/script: document.service.createFromImpact / (internal scheduler / service)
- Phase: C
- Output: compensation_documents (DB)
- Estimate: 18 ชั่วโมง
- พารามิเตอร์/cron อ่านจาก backend config (config file/env) — ไม่มีตาราง job_configs และไม่มีหน้าจอควบคุม (หน้า Flow Batch Job ในกลุ่มเมนู Flow เหลือแค่ Flowchart + Database ที่ใช้ · 2026-08-06)
- Runbook, rerun rule, risk และ history ตามเอกสาร Batch v4.0 · ผลการรันเขียน application log แบบ structured

3. Screenshot Reference

ไม่มีภาพหน้าจอสำหรับหัวข้อนี้ — เป็นเอกสารฝั่ง Backend/Batch ที่ไม่มี UI (ภาพหน้าจอทั้งหมดอยู่ในเอกสารชุด FE)

4. Implementation Flow Diagram (Reference)



รูปที่ 1: Implementation flow reference: LLDD BE - Job 8 CreateCompensationDocument

5. Field, Format, and Validation

Field / UI	Format	Validation	Behavior
กำหนดการรัน (Cron)	30 17 7-31 * *	แก้ไขได้	ใช้รอบเดิม แต่ปลายทางเป็น DB ภายใน
Target table	compensation_documents	ค่าคงที่/แก้ผ่านหน้าจอไม่ได้	สร้าง doc_no YYYY/xxxxx และผูก impact_process_id
เงื่อนไขเลือกข้อมูล	สถานะ I + forecast + ยังไม่สร้างเอกสาร	ค่าคงที่/แก้ผ่านหน้าจอไม่ได้	Gen Flow Gate อยู่ที่ Job 8b / Workflow Engine
ข้อห้ามเชิงสถาปัตยกรรม	ห้ามสร้างไฟล์ BPM06001O, ห้าม SFTP, ห้ามเรียก K2 REST	ค่าคงที่/แก้ผ่านหน้าจอไม่ได้	ใช้ Document Service + DB transaction เท่านั้น

5.9 Input / Progress / Output Contract

Stage	Contract for implementation
Input	Impact-store compensation rows in initial status with workflow sequence values and no prior confirm-receive output.
Progress	update BPM sequence, query eligible impact-store rows, refresh not-OPT data, generate workflow payload, insert confirm-receive rows, upload/export, notify.
Output	Impact-store workflow create payload/output with generated sequence numbers and duplicate guard.

5.90 Job 8 Execution Stages

update BPM sequence, query eligible impact-store rows, refresh not-OPT data, generate workflow payload, insert confirm-receive rows, upload/export, notify.

Order	Service step	Repository	Output / failure contract
1	loadDocumentCandidates	compensationDocumentRepository	คืน metrics และ throw typed error; transaction/rerun ใช้ contract ด้านล่าง
2	allocateDocumentNumbers	compensationDocumentRepository	คืน metrics และ throw typed error; transaction/rerun ใช้ contract ด้านล่าง
3	createCompensationDocuments	compensationDocumentRepository	คืน metrics และ throw typed error; transaction/rerun ใช้ contract ด้านล่าง
4	recordDocumentCreation	compensationDocumentRepository	คืน metrics และ throw typed error; transaction/rerun ใช้ contract ด้านล่าง

5.91 Job 8 Run Evidence

Evidence	Job-specific value	Acceptance
Input identity	Impact-store compensation rows in initial status with workflow sequence values and no prior confirm-receive output.	snapshot input file/business key/period in run record
Output identity	Impact-store workflow create payload/output with generated sequence numbers and duplicate guard.	reconcile input, success, reject and skipped counts
Dedup proof	UNIQUE(impact_process_id) และ UNIQUE(year,running_no); lock running number ต่อปีใน transaction; conflict ต้องคืน/อ้าง doc_no เดิม และยอมให้เลขที่จองกระโดดโดยห้าม reuse	rerun fixture produces no duplicate target business key
Transaction proof	lock เลขรัน + insert document + update process + tracking (direction=INTERNAL) ใน transaction เดียว	injected failure leaves no partial committed state outside documented boundary
Security proof	internal service account เท่านั้น; ห้ามสร้างไฟล์ BPM06001O, ห้าม SFTP และห้ามเก็บ K2 credential	config/log/error contains no plaintext secret

5.92 Legacy Java Source Reference

Legacy file	Line range	Responsibility to carry forward
fcsJar/src/th/co/gosoft/fgi/main/ExportImpactStoreFlowToBPM.java	9-17	Legacy main entrypoint for exporting impact-store flow data.
fcsJar/src/th/co/gosoft/fgi/controller/ExportController.java	518-657	Build impact-store BPM payload, write file, upload, backup, notification.
fcsJar/src/th/co/gosoft/fgi/dao/jdbc/ExportJdbc.java	1654-1692	Query impact-store rows eligible for workflow export.

Line ranges refer to the legacy Java implementation under /Users/bank_mac/gosoft/java/SBP/fcsJar. Use these ranges to preserve business behavior while implementing the target Node job.

5.93 Target Repository and SQL Contract

Contract	Target implementation
Repository	compensationDocumentRepository
Idempotency / dedup	UNIQUE(impact_process_id) และ UNIQUE(year,running_no); lock running number ต่อปีใน transaction; conflict ต้องคืน/อ้าง doc_no เดิม และยอมให้เลขที่จองกระโดดโดยห้าม reuse
Transaction boundary	lock เลขรัน + insert document + update process + tracking (direction=INTERNAL) ใน transaction เดียว

Contract	Target implementation
Security	internal service account เท่านั้น; ห้ามสร้างไฟล์ BPM06001O, ห้าม SFTP และห้ามเก็บ K2 credential

Input / candidate query

```
SELECT p.id AS impact_process_id, p.impacted_store_code, p.impact_month,
      SUM(COALESCE(s.adjust_compensation_amount, s.forecast_compensation_amount, 0)) AS total_compensation_amount
FROM fgi_impact_processes p
JOIN fgi_impact_stores s ON s.impact_process_id = p.id
WHERE p.process_status = 'READY_DOCUMENT'
GROUP BY p.id, p.impacted_store_code, p.impact_month;
```

Write / upsert query

```
INSERT INTO compensation_documents
   (doc_no, year, running_no, impact_process_id, impacted_store_code, impact_month,
    source, status_code, current_section_code, total_compensation_amount, created_by)
VALUES (:doc_no, :year, :running_no, :impact_process_id, :impacted_store_code, :impact_month,
       'FS', '06', '06', :total_compensation_amount, 'JOB-8')
ON CONFLICT (impact_process_id) DO NOTHING;

INSERT INTO interface_transactions
   (run_id, data_name, direction, status, impact_process_id, doc_no,
    business_key, period_key, outbox_status, purge_after, completed_at)
SELECT :run_id, 'DOCUMENT_CREATE', 'INTERNAL', 'COMPLETED', d.impact_process_id, d.doc_no,
      CAST(d.impact_process_id AS VARCHAR), d.impact_month, 'COMPLETED',
      CURRENT_TIMESTAMP + INTERVAL '365 days', CURRENT_TIMESTAMP
FROM compensation_documents d
WHERE d.impact_process_id = :impact_process_id
ON CONFLICT (data_name, direction, business_key, period_key) DO NOTHING;
```

5.94 Target Node Implementation

โครงสร้างนี้ระบุ service/repository เฉพาะงานและต้อง implement ตาม SQL, transaction, idempotency และ security contract ด้านบน โดยทุกขั้นต้องคืน metrics สำหรับ reconcile และ run history

```
export async function runLIddBeJob8Createcompensationdocument(ctx, services) {
  const run = await services.jobRuns.acquire({
    jobNo: "8", period: ctx.period, triggeredBy: ctx.triggeredBy
  });

  try {
    ctx = { ...ctx, runId: run.id, repository: services.compensationDocumentRepository };
    const step1 = await services.loadDocumentCandidates(ctx, undefined);
    const step2 = await services.allocateDocumentNumbers(ctx, step1);
    const step3 = await services.createCompensationDocuments(ctx, step2);
    const step4 = await services.recordDocumentCreation(ctx, step3);
    const result = step4;
    await services.jobRuns.finish(run.id, "SUCCESS", result.metrics);
    return { runId: run.id, status: "SUCCESS", ...result };
  } catch (error) {
    await services.jobRuns.finish(run.id, "FAILED", {
      errorCode: error.code ?? "JOB_FAILED",
      errorMessage: error.message
    });
    throw error;
  }
}
```

5.95 Job 8 Document Number Gap and Rerun Policy

Job 8 ใช้ running number แบบ monotonic ต่อปี ค.ศ. ช่องว่างของเลขเอกสารจาก concurrent rerun หรือ ON CONFLICT เป็นพฤติกรรมที่ยอมรับได้ เพราะเลขที่มีหน้าที่รับประกัน uniqueness ไม่ได้รับประกันความต่อเนื่อง

Case	Required behavior	Evidence / metric
Rerun พบ impact_process_id เดิมก่อนจองเลข	คืน/ข้ามด้วย doc_no เดิมโดยไม่จอง running_no เพิ่มเมื่อ fast lookup พบข้อมูลแล้ว	duplicateExistingCount + existingDocNo
Concurrent worker ชน ON CONFLICT หลังจองเลข	ยอมให้ running_no ที่จองแล้วกลายเป็น gap; ห้ามลด sequence และห้ามนำเลขกลับมาใช้	numberGapCount + conflictedImpactProcessId
Conflict path	อ่าน compensation_documents ด้วย impact_process_id แล้วใช้ d.doc_no เดิมสำหรับ tracking/reconcile	tracking.doc_no ตรงกับเอกสารที่ commit อยู่จริง

Case	Required behavior	Evidence / metric
New document path	insert document และ tracking (direction=INTERNAL) ใน transaction เดียว	createdCount และ trackingCount เพิ่มเท่ากัน
Audit/runbook	อธิบายว่าเลขอาจไม่ต่อเนื่องแต่ต้องไม่ซ้ำและตรวจสอบย้อนกลับได้	ไม่มีขั้นตอน manual reuse หรือ renumber

6. Button / User Action Mapping

Action	Trigger	API / Service	Expected Result
รันตามตารางเวลา	CRON	scheduler → runner (job 8)	อ่าน cron/พารามิเตอร์จาก backend config
รันนอกรอบ (manual/rerun)	CLI	CLI/ops runbook → runner (job 8)	guard ไม่ให้รันซ้อนด้วย distributed lock
แก้พารามิเตอร์/เปิด-ปิด job	CONFIG	แก้ backend config แล้ว deploy	ไม่มี endpoint และไม่มีหน้าจอควบคุม — หน้า Flow Batch Job เป็น reference อย่างเดียว (2026-08-06)
ตรวจผลการรัน	LOG	application log (structured)	ไม่มีตาราง job_run_histories แล้ว · ไฟล์/ACK ดูที่ interface_transactions

7. API Contract

8. Reference DB Mapping (No Database Page Work)

ส่วนนี้เป็นข้อมูลอ้างอิงสำหรับการ implement API/Job เท่านั้น ไม่ใช่งานสร้างหน้า Database, ไม่ใช่งานออกแบบ DB page และไม่ถูกนับเป็น deliverable แยกของ FE/BE

Table / Object	R/W	Usage
document_running_numbers	R/W	ตัวนับเลขเอกสารรายปี — ออก doc_no YYYY/xxxxx (ค.ศ.)
fgi_impact_stores	R/W	อ่าน candidate และอัปเดตสถานะสร้างเอกสาร
fgi_impact_processes	R	hub รอบชดเชย
compensation_documents	W	สร้างหัวเอกสารแทนไฟล์ BPM06001O
interface_transactions	W	tracking ภายใน: direction=INTERNAL · status=COMPLETED (ไม่มี ACK ให้รอเพราะเขียน DB ตรง)

9. Skeleton Code (Batch Job 8)

9.1 ผังไฟล์ที่ต้องสร้าง (Job 8)

โครงไฟล์ของ Job 8 (document.service.createFromImpact เดิม) วางใต้ `src/batch/sbpgi/` ของ store-backend โดยใช้ convention เดียวกับ module ธุรกิจอื่น: inject custom provider `DATA_SOURCE` แล้วยิง raw SQL, repository ประกาศเป็น factory provider ที่ใช้ token string, entity อยู่ใน `src/entitys/`

หมายเหตุสำคัญ — `src/batch/*` ทั้งชุดเป็นของใหม่ที่ยังไม่มีใน store-backend: ปัจจุบัน repo ไม่มีโฟลเดอร์ `src/batch` เลย และแม้จะติดตั้ง `@nestjs/schedule` ไว้แล้วก็ยังไม่มี `@Cron`/`@Interval` แม้แต่จุดเดียว ดังนั้น `runner.ts` / `scheduler.ts` / `cli.js` / `job-failure.notifier.ts` คือ งานตั้งต้นของ Phase แรก ที่ต้องสร้างเองทั้งหมด พร้อม register `ScheduleModule.forRoot()` ใน `app.module.ts` — ไม่ใช่ของเดิมที่ reuse ได้

Path	หน้าที่
src/batch/sbpgi/job-8-create-compensation-document/job-8-create-compensation-document.job.ts	คลาส `CreateCompensationDocumentJob` — `run(ctx)` เรียงตาม flow ของ Job 8 ทีละขั้น, ครอบ transaction, จบด้วย structured log

Path	หน้าที่
src/batch/sbpgi/job-8-create-compensation-document/job-8-create-compensation-document.service.ts	คลาส `CreateCompensationDocumentService` — logic ต่อขั้น (อ่าน/parse/คำนวณ/เขียน) + repository token ที่ inject จาก `DATA_SOURCE`
src/batch/sbpgi/job-8-create-compensation-document/job-8-create-compensation-document.config.ts	คลาส `SbpgiJob8Config` (แบบเดียวกับ `src/config/app.config.ts` — โปรเจกต์นี้ไม่ใช้ `registerAs`) — cron และพารามิเตอร์ทั้ง 4 ตัวของ Job 8 อ่านจาก env/config file (ไม่มีตาราง job_configs)
src/batch/sbpgi/job-8-create-compensation-document/job-8-create-compensation-document.module.ts	NestJS module ผูก job + service + repository provider (factory token string) เข้ากับ `DatabaseModule`
src/batch/runner.ts	ตัวรันกลาง: resolve job ตาม jobNo, กันรันซ้อนด้วย advisory lock, จับ error → แจ้งเตือน, เขียน structured log สรุป (ใช้ร่วมทั้ง 11 job)
src/batch/scheduler.ts	ลงทะเบียน cron จาก config (`SBPGI_JOB8_CRON` = `30 17 7-31 * *`) และรองรับสั่งรันนอกรอบผ่าน CLI/runbook
src/batch/job-failure.notifier.ts	ส่งอีเมลแจ้งผู้ดูแลเมื่อ job ล้มเหลว ผ่าน `EmailLibService` ของ `@gosoft-sbp/email-lib` (log ลง `email_sent` ให้อัตโนมัติ)

9.2 Config Schema ของ Job 8 (backend config / env)

cron ปัจจุบันของ Job 8 คือ `30 17 7-31 * *` (วันที่ 7–31 เวลา 17:30) — ประกาศเป็น `SBPGI_JOB8_CRON` และอ่านตอน bootstrap ของ `scheduler.ts`; ถ้า `enabled=false` scheduler ต้องไม่ลงทะเบียน cron ของ job นี้

```
// src/batch/sbpgi/job-8-create-compensation-document/job-8-create-compensation-document.config.ts
// convention จริงของ store-backend คือคลาส config (`src/config/app.config.ts` ที่ export ผ่าน
// `AppConfigModule` แบบ @Global แล้วอ่าน process.env ตรง ๆ) — โปรเจกต์นี้ **ไม่ได้ใช้ registerAs**
// แม้แต่จุดเดียว จึงประกาศเป็นคลาสให้รีวิว/ทดสอบเหมือน config ตัวอื่น
import { Injectable } from '@nestjs/common';

// TODO: Job 8 ไม่มีตาราง job_configs และไม่มี Job Admin API แล้ว (ตัดสินใจ 2026-08-06)
// TODO: ค่าทุกตัวอ่านจาก env/config file ของ backend เท่านั้น — เปลี่ยนค่า = แก้ config แล้ว deploy
export interface Job8Config {
  /** เปิด/ปิด job รอบถัดไปโดยไม่ต้อง deploy โค้ด */
  enabled: boolean;
  /** cron ของ job นี้ (อ่านตอน bootstrap ของ scheduler.ts) */
  cron: string;
  /** กำหนดการรัน (Cron) — ใช้รอบเดิม แต่ปลายทางเป็น DB ภายใน */
  cron: string;
  /** Target table — สร้าง doc_no YYYY/xxxxx และผูก impact_process_id */
  targetTable: string;
  /** เงื่อนไขเลือกข้อมูล — Gen Flow Gate อยู่ที่ Job 8b / Workflow Engine */
  condition: string;
  /** ข้อห้ามเชิงสถาปัตยกรรม — ใช้ Document Service + DB transaction เท่านั้น */
  param4: string;
  /** ผู้รับอีเมลเมื่อ job ล้มเหลว — เก็บเป็น string คั่น comma ให้ตรง signature ของ
      `EmailLibService.sendMail({ mailTo })` ที่รับ string ไม่ใช่ string[] */
  mailTo: string;
}

@Injectable()
export class SbpgiJob8Config implements Job8Config {
  // TODO: ยืนยันค่า default ทุกตัวกับ Ops ก่อนขึ้น production (ไม่มีหน้าจอแก้ค่าแล้ว)
  enabled = (process.env.SBPGI_JOB8_ENABLED ?? 'true') === 'true';
  cron = process.env.SBPGI_JOB8_CRON ?? '30 17 7-31 * *';
  cron = process.env.SBPGI_JOB8_CRON ?? '30 17 7-31 * *'; // TODO: แก้ผ่าน env/config file แล้ว deploy
  targetTable = process.env.SBPGI_JOB8_TARGET_TABLE ?? 'compensation_documents'; // TODO: ค่าคงที่ทางธุรกิจ —
เปลี่ยนต้องผ่านการอนุมัติ
  condition = process.env.SBPGI_JOB8_CONDITION ?? 'สถานะ I + forecast + ยังไม่สร้างเอกสาร'; // TODO: ค่าคงที่ทางธุรกิจ —
เปลี่ยนต้องผ่านการอนุมัติ
  param4 = process.env.SBPGI_JOB8_PARAM4 ?? 'ห้ามสร้างไฟล์ BPM06001O, ห้าม SFTP, ห้ามเรียก K2 REST'; // TODO: ค่าคงที่ทางธุรกิจ —
เปลี่ยนต้องผ่านการอนุมัติ
  mailTo = process.env.SBPGI_JOB8_MAIL_TO ?? ''; // TODO: ผู้รับอีเมลแจ้ง error คั่นด้วย comma (เดิม: email-lib กลาง (sendEmail) แจ้ง
error/pending ตาม config)
}

// TODO: เพิ่ม SbpgiJob8Config ใน providers/exports ของ AppConfigModule (@Global) เหมือน AppConfig
```

9.3 Job Class — `run(ctx)` ของ Job 8 ทีละขั้นตามผัง

9.3.1 สัญญาของชั้นกลาง (`runner.ts`) + โครง service ของ Job 8

job class อ้าง `JobRunContext` / `JobRunResult` / `JobState` / `JobFailedError` — ทั้งหมดนิยาม ครั้งเดียวใน `src/batch/runner.ts` (ไฟล์ร่วมของทุก job ให้ merge ไม่ใช่เขียนทับ) และ service ต้องมี method ครบตามตารางขั้นตอนด้านล่าง

มิฉะนั้น job class จะเรียก method ที่ไม่มีอยู่

```
// src/batch/runner.ts — สัญญากลางของทุก job (ประกาศครั้งเดียว ใช้ร่วมทั้ง 10 ฉบับ)

export interface JobRunContext {
  jobNo: string;
  period: string; // YYYYMM ของงวดที่รัน
  triggeredBy: string; // 'CRON' | userId ที่สั่งรันนอกรอบ
  params?: Record<string, string>;
}

export interface JobRunResult {
  event: 'job.finish';
  jobNo: string;
  jobName: string;
  status: 'SUCCESS' | 'SKIPPED' | 'SKIPPED_LOCKED' | 'FAILED';
  period: string;
  output: string;
  read: number; written: number; skipped: number; rejected: number;
  durationMs: number;
}

/** counter + ค่าที่ทุกขั้นของ job ใช้ร่วมกัน (service เป็นผู้สร้างผ่าน createState) */
export interface JobState {
  period: string;
  read: number; written: number; skipped: number; rejected: number;
  // TODO: เพิ่ม field เฉพาะของ job นี้ (เช่น rows ที่อ่านมา, path ไฟล์ที่เขียน)
  [key: string]: unknown;
}

/** error ที่ทำให้ job จบเป็น FAILED และส่งอีเมลแจ้งผู้ดูแล */
export class JobFailedError extends Error {
  constructor(public readonly code: string, message: string) { super(message); }
}

/** ใช้ออกจาก transaction เมื่อสาขา NO บอกให้ข้ามงวด/เรคคอร์ด — runner สรุปเป็น SKIPPED ไม่ใช่ FAILED */
export class JobSkippedError extends Error {}
```

```
// CreateCompensationDocumentService — method ที่ job class เรียก (1 method ต่อ 1 ขั้นในตารางด้านบน)
import { Inject, Injectable } from '@nestjs/common';
import type { DataSource, EntityManager } from 'typeorm';
import type { JobRunContext, JobState } from '../../runner';
export type { JobState };

@Injectable()
export class CreateCompensationDocumentService {
  constructor(@Inject('DATA_SOURCE') private readonly dataSource: DataSource) {}

  createState(ctx: JobRunContext): JobState {
    return { period: ctx.period, read: 0, written: 0, skipped: 0, rejected: 0 };
  }

  // query impact profile สถานะ I + forecast + ยังไม่สร้างเอกสาร
  async step02Document(state: JobState, manager?: EntityManager): Promise<void> {
    // TODO: implement
  }

  // ข้อมูลผู้อนุมัติ/ร้าน/ยอดชดเชยครบ?
  async check03Condition(state: JobState): Promise<boolean> {
    return true; // TODO: เงื่อนไขจริงตามผัง
  }

  // generate doc_no YYYY/xxxxx
  async step04Document(state: JobState, manager?: EntityManager): Promise<void> {
    // TODO: implement
  }

  // insert compensation_documents
  async step05Insert(state: JobState, manager?: EntityManager): Promise<void> {
    // TODO: implement
  }

  // insert interface_transactions: data_name = IMPACT_STORE · direction = INTERNAL · status = COMPLETED
  async step06WriteFile(state: JobState, manager?: EntityManager): Promise<void> {
    // TODO: implement
  }

}
```

9.3.2 `run(ctx)` ของ Job 8

ทุกขั้นใน `run()` ตรงกับ flowchart ของ Job 8 หนึ่งต่อหนึ่ง (decision และ error path รวมอยู่ด้วย) — method ที่ต้อง implement ใน service ตามตารางนี้

ลำดับ	ชนิด	ขั้นตอนจากผัง	Method ที่ต้อง implement	เส้นทาง NO / error
1	start	เริ่ม	createState()	-
2	process	query impact profile สถานะ I + forecast + ยังไม่สร้างเอกสาร	step02Document()	throw JobFailedError เมื่อทำไม่สำเร็จ
3	decision	ข้อมูลผู้อนุมัติ/ร้าน/ยอดชดเชยครบ?	check03Condition()	[err] บันทึก reject reason / ไม่สร้างเอกสาร
4	process	generate doc_no YYYY/xxxxx	step04Document()	throw JobFailedError เมื่อทำไม่สำเร็จ
5	process	insert compensation_documents	step05Insert()	throw JobFailedError เมื่อทำไม่สำเร็จ
6	process	insert interface_transactions: data_name = IMPACT_STORE · direction = INTERNAL · status = COMPLETED	step06WriteFile()	throw JobFailedError เมื่อทำไม่สำเร็จ
7	end	จบ - workflow เปิดโดย Job 8b / POST /workflows/instances	summarize()	-

```
// src/batch/sbpgi/job-8-create-compensation-document/job-8-create-compensation-document.job.ts
import { Inject, Injectable, Logger } from '@nestjs/common';
import type { DataSource, EntityManager } from 'typeorm';
import { CreateCompensationDocumentService, type JobState } from './job-8-create-compensation-document.service';
// 4 symbol นี้นิยามใน src/batch/runner.ts (ดูหัวข้อ 9.3.1)
import { JobFailedError, JobSkippedError, JobRunContext, JobRunResult } from '../../runner';

@Injectable()
export class CreateCompensationDocumentJob {
  static readonly jobNo = '8';
  private readonly logger = new Logger(CreateCompensationDocumentJob.name);

  constructor(
    // TODO: DATA_SOURCE = custom provider ที่ route SELECT/WITH ไป slave pool และ write ไป master
    @Inject('DATA_SOURCE') private readonly dataSource: DataSource,
    private readonly service: CreateCompensationDocumentService,
  ) {}

  async run(ctx: JobRunContext): Promise<JobRunResult> {
    const startedAt = Date.now();
    // TODO: state ถือ counter (read/written/skipped/rejected) และค่าจาก job8Config
    const state = this.service.createState(ctx);
    try {
      // === transaction boundary === TODO: DB transaction เดียวครอบ generate doc_no + insert document + tracking
      await this.dataSource.transaction(async (manager: EntityManager) => {
        // ขั้นที่ 2: query impact profile สถานะ I + forecast + ยังไม่สร้างเอกสาร · TODO: ใช้ impact_process_id เป็น idempotency key
        await this.service.step02Document(state, manager);
        // ขั้นที่ 3 (decision): ข้อมูลผู้อนุมัติ/ร้าน/ยอดชดเชยครบ?
        const ok03 = await this.service.check03Condition(state);
        if (!ok03) throw new JobFailedError('JOB8_STEP03', 'บันทึก reject reason / ไม่สร้างเอกสาร');
        // ขั้นที่ 4: generate doc_no YYYY/xxxxx · TODO: running ต่อปี ค.ศ. (มติ 2026-08-06)
        await this.service.step04Document(state, manager);
        // ขั้นที่ 5: insert compensation_documents · TODO: ผูก impact_process_id และสถานะเริ่มต้น
        await this.service.step05Insert(state, manager);
        // ขั้นที่ 6: insert interface_transactions: data_name = IMPACT_STORE · direction = INTERNAL · status = COMPLETED · TODO:
ไม่สร้างไฟล์ BPM06001O แล้ว — เขียน DB ตรงจึงไม่มี ACK ให้รอ
        await this.service.step06WriteFile(state, manager);
      });
      return this.summarize(state, 'SUCCESS', startedAt);
    } catch (error) {
      // TODO: error path ของ Job 8 — ห้ามนำ logic SFTP compensateflow หรือ K2 StartInstance กลับมาใช้ใน target design
      this.logger.error(JSON.stringify({ event: 'job.failed', jobNo: '8', period: ctx.period,
        triggeredBy: ctx.triggeredBy, durationMs: Date.now() - startedAt, error: (error as Error).message }));
      // TODO: แจ้งผู้ดูแลผาน JobFailureNotifier (หัวข้อ 9.6.1) — runner เป็นผู้เรียกให้
      throw error;
    }
  }

  private summarize(state: JobState, status: JobRunResult['status'], startedAt = Date.now()): JobRunResult {
    // TODO: structured log บรรทัดเดียวจบ — ไม่มีตาราง job_run_histories แล้ว (2026-08-06)
    const summary = {
      event: 'job.finish', jobNo: '8', jobName: 'CreateCompensationDocument', status,
      period: state.period, output: 'compensation_documents (DB)',
```

```
      read: state.read, written: state.written, skipped: state.skipped,
      rejected: state.rejected, durationMs: Date.now() - startedAt,
    };
    this.logger.log(JSON.stringify(summary));
    return summary as JobRunResult;
  }
}
```

9.4 การกันรันซ้อนของ Job 8 (PostgreSQL advisory lock)

Job 8 มีข้อควรระวังจาก legacy: ห้ามนำ logic SFTP compensateflow หรือ K2 StartInstance กลับมาใช้ใน target design — runner ล็อกด้วย `pg_try_advisory_lock` ก่อนเริ่มขั้นแรกเสมอ และรอบที่ล็อกไม่ได้ให้จบด้วยสถานะ SKIPPED_LOCKED (ไม่ใช่ FAILED)

```
// src/batch/runner.ts (ส่วนกันรันซ้อน)
import { Inject, Injectable, Logger } from '@nestjs/common';
import type { DataSource } from 'typeorm';

// TODO: ห้ามใช้แถวสถานะ RUNNING ในตารางเป็นตัวกัน (ไม่มีตาราง job_run_histories แล้ว)
// ใช้ PostgreSQL advisory lock ระดับ session แทน — ปลดอัตโนมัติเมื่อ connection หลุด
export const SBPGI_JOB_LOCK_CLASS_ID = 861000; // namespace ของระบบ SBPGI
export const JOB_LOCK_KEYS: Record<string, number> = { '8': 80 /* TODO: เพิ่มครบทั้ง 11 job */ };

@Injectable()
export class BatchRunner {
  private readonly logger = new Logger(BatchRunner.name);
  constructor(@Inject('DATA_SOURCE') private readonly dataSource: DataSource) {}

  async runExclusive<T>(jobNo: string, fn: () => Promise<T>): Promise<T | { status: 'SKIPPED_LOCKED' }> {
    // TODO: ต้องใช้ QueryRunner (connection เดียวบน master) — dataSource.query() ของโปรเจกต์นี้
    // route SQL ที่ขึ้นต้นด้วย SELECT ไป slave pool ทำให้ lock ไปตกที่ replica คนละ connection
    const runner = this.dataSource.createQueryRunner('master');
    await runner.connect();
    const objectId = JOB_LOCK_KEYS[jobNo];
    try {
      const [{ locked }] = await runner.query(
        'SELECT pg_try_advisory_lock($1, $2) AS locked',
        [SBPGI_JOB_LOCK_CLASS_ID, objectId],
      );
      if (!locked) {
        // TODO: รอบนี้ข้ามไปเฉย ๆ ไม่ถือเป็น error และไม่ต้องส่งอีเมล
        this.logger.warn(JSON.stringify({ event: 'job.skipped.locked', jobNo }));
        return { status: 'SKIPPED_LOCKED' };
      }
      return await fn();
    } finally {
      // TODO: ปลด lock ทุกกรณี แล้วคืน connection เข้า pool
      await runner.query('SELECT pg_advisory_unlock($1, $2)', [SBPGI_JOB_LOCK_CLASS_ID, objectId]);
      await runner.release();
    }
  }
}
```

9.5 Repository / SQL หลักของ Job 8

repository ของ Job 8 ประกาศเป็น factory provider (`{provide: 'CREATE_COMPENSATION_DOCUMENT_REPOSITORY', useFactory: (ds) => ds.getRepository(Entity), inject: ['DATA_SOURCE']}`) แล้วยิง raw SQL ตามแบบ module ธุรกิจอื่นของ store-backend (schema `sps_store` มาจาก search_path)

ตาราง	R/W	การใช้งานตามผัง	หมายเหตุ target design
document_running_numbers	R/W	ตัวนับเลขเอกสารรายปี — ออก doc_no YYYY/xxxxx (ค.ศ.)	เขียน SQL ตรงผ่าน DATA_SOURCE
fgi_impact_stores	R/W	อ่าน candidate และอัปเดตสถานะสร้างเอกสาร	เขียน SQL ตรงผ่าน DATA_SOURCE
fgi_impact_processes	R	hub รอบชดเชย	เขียน SQL ตรงผ่าน DATA_SOURCE
compensation_documents	W	สร้างหัวเอกสารแทนไฟล์ BPM06001O	เขียน SQL ตรงผ่าน DATA_SOURCE
interface_transactions	W	tracking ภายใน: direction=INTERNAL · status=COMPLETED (ไม่มี ACK ให้รอเพราะเขียน DB ตรง)	เขียน SQL ตรงผ่าน DATA_SOURCE

```
-- Job 8 CreateCompensationDocument — query หลักที่ต้อง implement
-- TODO: ทุก statement รันผ่าน DATA_SOURCE (SELECT ไป slave, write ไป master) และ
-- write ทั้งหมดต้องอยู่ใน transaction เดียวกับที่ระบุใน 9.3

-- [R/W] document_running_numbers : ตัวนับเลขเอกสารรายปี — ออก doc_no YYYY/xxxxx (ค.ศ.)
-- TODO: อ่าน candidate แบบล็อกแถว กันรอบอื่น/pod อื่นแย่งอัปเดตแถวเดียวกัน
SELECT /* TODO: PK + คอลัมน์ที่ต้องใช้ */
  FROM document_running_numbers
 WHERE /* TODO: เงื่อนไขงวด/สถานะที่ job นี้คัดแถว */ 1 = 1
   FOR UPDATE SKIP LOCKED;

UPDATE document_running_numbers
   SET /* TODO: คอลัมน์สถานะ/ผลคำนวณที่ job นี้เขียน */
       updated_at = NOW(), updated_by = 'JOB8'
 WHERE /* TODO: PK ที่ล็อกไว้ */ id = ANY($1);

-- [R/W] fgi_impact_stores : อ่าน candidate และอัปเดตสถานะสร้างเอกสาร
-- TODO: อ่าน candidate แบบล็อกแถว กันรอบอื่น/pod อื่นแย่งอัปเดตแถวเดียวกัน
SELECT /* TODO: PK + คอลัมน์ที่ต้องใช้ */
  FROM fgi_impact_stores
 WHERE impact_year = $1 AND impact_month = $2 -- TODO: ยืนยันชื่อคอลัมน์งวดกับ Database design
   FOR UPDATE SKIP LOCKED;

UPDATE fgi_impact_stores
   SET /* TODO: คอลัมน์สถานะ/ผลคำนวณที่ job นี้เขียน */
       updated_at = NOW(), updated_by = 'JOB8'
 WHERE /* TODO: PK ที่ล็อกไว้ */ id = ANY($1);

-- [R] fgi_impact_processes : hub รอบชดเชย
-- TODO: เติมเฉพาะคอลัมน์ที่ job ใช้จริง (ห้าม SELECT *) และตรวจว่ามี index รองรับ WHERE นี้
SELECT /* TODO: columns */
  FROM fgi_impact_processes
 WHERE impact_year = $1 AND impact_month = $2 -- TODO: ยืนยันชื่อคอลัมน์งวดกับ Database design
 ORDER BY /* TODO: คีย์ที่ทำให้ลำดับคงที่ */
 LIMIT $3 OFFSET $4; -- TODO: อ่านเป็น chunk กัน memory บวม

-- [W] compensation_documents : สร้างหัวเอกสารแทนไฟล์ BPM06001O
-- TODO: เติมคอลัมน์ payload จริงจาก Database design
INSERT INTO compensation_documents
  (/* TODO: business key + payload + created_by, created_at */)
VALUES (/* TODO: bind params ตามลำดับคอลัมน์ด้านบน */)
ON CONFLICT (source, impacted_store_code, impact_month, new_store_code, round_no) -- unique key จริงตาม DDL ของ
compensation_documents (ห้ามเดา)
DO UPDATE SET /* TODO: คอลัมน์ที่ยอมให้ทับ */
       updated_at = NOW(), updated_by = 'JOB8';
```

9.6 การแจ้งเตือนและการรันซ้ำของ Job 8

9.6.1 อีเมลแจ้งผู้ดูแลเมื่อ job ล้มเหลว

ใช้ `EmailLibService` จาก `@gosoft-sbp/email-lib` ตัวเดียวกับที่ระบบเดิมใช้ (inform-evaluate / external-audit / statement PTT) — ไม่สร้างกลไกส่งเมลใหม่

```
// src/batch/job-failure.notifier.ts
import { Injectable, Logger } from '@nestjs/common';
// ชื่อ method ของ lib ที่ store-backend เรียกจริงคือ `sendMail` (ไม่ใช่ sendEmail) และ
// `mailTo` / `mailCc` เป็น **string** คั่นด้วย comma — ดู evaluation-process.service.ts,
// external-audit.service.ts, statement.service.ts, inform-evaluate.service.ts, performance.service.ts
import { EmailLibService } from '@gosoft-sbp/email-lib';
import type { JobRunContext } from './runner';

@Injectable()
export class JobFailureNotifier {
  private readonly logger = new Logger(JobFailureNotifier.name);
  // TODO: ใช้ lib อีเมลของระบบเดิม — template อยู่ในตาราง email_template และ log ลง email_sent อัตโนมัติ
  // (ตั้งชื่อ property ว่า mailService ตาม call site เดิมทุกที่ใน store-backend)
  constructor(private readonly mailService: EmailLibService) {}

  async notifyFailure(jobNo: string, ctx: JobRunContext, error: Error): Promise<void> {
    // TODO: ผู้รับของ Job 8 เดิมคือ email-lib กลาง (sendEmail) แจ้ง error/pending ตาม config — ย้ายมาเป็น env SBPGI_JOB8_MAIL_TO
    const recipients = (process.env.SBPGI_JOB8_MAIL_TO ?? '').split(',').map((s) => s.trim()).filter(Boolean);
    if (!recipients.length) {
      this.logger.warn(JSON.stringify({ event: 'job.mail.skipped', jobNo, reason: 'NO_RECIPIENT' }));
      return;
    }
    try {
      await this.mailService.sendMail({
        // TODO: emailId = id ของ template EM-07 (แจ้ง error batch) ในตาราง email_template
        emailId: Number(process.env.SBPGI_JOB_FAIL_EMAIL_TEMPLATE_ID),
        mailTo: recipients.join(','), // signature รับ string ไม่ใช่ string[]
        mailCc: '',
        param: {
```

```
        jobNo, jobName: 'CreateCompensationDocument',
        jobTitle: 'สร้างเอกสารประกันรายได้อัตโนมัติ',
        period: ctx.period, triggeredBy: ctx.triggeredBy,
        output: 'compensation_documents (DB)',
        errorMessage: error.message,
        rerunNote: 'idempotent ด้วย impact_process_id; เจอ doc เดิมให้ skip และคืนสถานะ already_created',
      },
    });
  } catch (mailError) {
    // TODO: ส่งเมลไม่สำเร็จห้ามกลบ error เดิมของ job — log แล้วปล่อยผ่าน
    this.logger.error(JSON.stringify({ event: 'job.mail.failed', jobNo, error: (mailError as Error).message }));
  }
 }
}
```

9.6.2 Checklist การ rerun

- กติกา rerun ของ Job 8: idempotent ด้วย impact_process_id; เจอ doc เดิมให้ skip และคืนสถานะ already_created
- ขอบเขต transaction ที่ต้องรักษาเมื่อรันซ้ำ: DB transaction เดียวครอบ generate doc_no + insert document + tracking
- ความเสี่ยงที่ต้องตรวจก่อน/หลังรันซ้ำ: ห้ามนำ logic SFTP compensateflow หรือ K2 StartInstance กลับมาใช้ใน target design
- ตรวจว่ารอบก่อนหน้าไม่ได้ค้าง lock อยู่ (`SELECT * FROM pg_locks WHERE locktype = 'advisory'`) ก่อนสั่งรันนอกรอบ
- สั่งรันนอกรอบผ่าน CLI/runbook เท่านั้น (ไม่มีหน้าจอและไม่มี Job Admin API): `node dist/batch/cli.js --job=8 --period=<YYYYMM>`
- หลังรันซ้ำ ตรวจ output `compensation_documents (DB)` และ log บรรทัด `job.finish` ว่า read/written/skipped/rejected ตรงกับที่คาด
- ถ้ารอบก่อนล้มเหลวกลางทาง ตรวจ `interface_transactions` ของงวดนั้นว่ามีแถวค้างสถานะ READY/PENDING หรือไม่ ก่อนสั่งรันใหม่

10. Processing Flow

Step	Description
1	เริ่ม
2	query impact profile สถานะ I + forecast + ยังไม่สร้างเอกสาร (ใช้ impact_process_id เป็น idempotency key)
3	ข้อมูลผู้อนุมัติ/ร้าน/ยอดชดเชยครบ? | No: บันทึก reject reason / ไม่สร้างเอกสาร
4	generate doc_no YYYY/xxxxx (running ต่อปี ค.ศ. (มติ 2026-08-06))
5	insert compensation_documents (ผูก impact_process_id และสถานะเริ่มต้น)
6	insert interface_transactions: data_name = IMPACT_STORE · direction = INTERNAL · status = COMPLETED (ไม่สร้างไฟล์ BPM06001O แล้ว — เขียน DB ตรงจึงไม่มี ACK ให้รอ)
7	จบ - workflow เปิดโดย Job 8b / POST /workflows/instances

11. Acceptance Criteria

- พารามิเตอร์และ cron อ่านจาก backend config เท่านั้น — เปลี่ยนค่าโดย deploy config ไม่ใช่ผ่าน API/หน้าจอ
- การรันต้องตรวจ enabled flag ใน config และกันรันซ้อนด้วย distributed/advisory lock
- ทุกรอบต้องเขียน application log แบบ structured (เวลา/แถว/ไฟล์/ผล) และ error ต้องส่ง EM-07
- DB/table mapping ใช้เป็น reference สำหรับ implement Job เท่านั้น ไม่ใช่งานสร้างหน้า Database
- รองรับ rerun rule และ risk note ตาม runbook

12. Developer Test Checklist

No	Test
1	รันตามตารางเวลาแล้วผลถูกต้องบน fixture
2	รันนอกรอบผ่าน CLI ได้ผลเดียวกับ cron
3	สั่งรันซ้อนขณะกำลังรัน → runner ปฏิเสธ (lock ทำงาน)
4	แก้ config แล้ว deploy → รอบถัดไปใช้ค่าใหม่
5	job throw error → EM-07 ออก และ log มีบรรทัด error
6	ตรวจผลกระทบตารางตาม R/W mapping reference

13. Unit Test Scope

6 ชั่วโมง (30% ของ implementation 18 ชั่วโมง) · เครื่องมือ: Jest + mock repository/DataSource (ไม่ต่อ DB จริง)

หัวข้อนี้คือ unit test ที่ต้องเขียนคู่กับโค้ด — ต่างจาก *Developer Test Checklist* ซึ่งเป็น scenario ระดับ end-to-end/manual ที่ใช้ตอนตรวจรับ · รายการด้านล่าง derive จาก field/validation, acceptance criteria, endpoint และตารางที่เอกสารนี้เขียน

สิ่งที่ทดสอบ	ประเภท	เกณฑ์ผ่าน
เงื่อนไขเลือกข้อมูล	rule	ใช้กฎกับข้อมูลตัวอย่างแล้วได้ผลตามที่ระบุ — สถานะ I + forecast + ยังไม่สร้างเอกสาร
business rule	logic	พารามิเตอร์และ cron อ่านจาก backend config เท่านั้น — เปลี่ยนค่าโดย deploy config ไม่ใช่ผ่าน API/หน้าจอ
business rule	logic	การรันต้องตรวจ enabled flag ใน config และกันรันซ้อนด้วย distributed/advisory lock
business rule	logic	ทุกรอบต้องเขียน application log แบบ structured (เวลา/แถว/ไฟล์/ผล) และ error ต้องส่ง EM-07
business rule	logic	DB/table mapping ใช้เป็น reference สำหรับ implement Job เท่านั้น ไม่ใช่งานสร้างหน้า Database
business rule	logic	รองรับ rerun rule และ risk note ตาม runbook
`document_running_numbers`, `fgi_impact_stores`, `compensation_documents`	transaction	จำลอง error กลางทาง แล้วยืนยันว่า rollback ครบ ไม่เหลือแถวค้าง (mock DataSource/QueryRunner)
runner	idempotency	รันซ้ำด้วย fixture เดิมต้องไม่เกิดแถวซ้ำ (ON CONFLICT / business unique key ทำงาน)
runner	lock	เรียกซ้อนขณะกำลังรัน ต้องถูกปฏิเสธด้วย advisory lock

- ทุกเคสต้องรันได้โดยไม่ต่อ DB/บริการภายนอกจริง — mock ที่ขอบ repository/client เสมอ
- ข้อความไทยที่ยืนยันในเทสต้องเป็น verbatim ตาม ข้อกำหนดทางธุรกิจ ห้ามพิมพ์ใหม่
- เกณฑ์ผ่านของ CI: ทุกเคสในตารางนี้มี test จริงและผ่านทั้งหมด

ภาคผนวก: Java Source เดิม

Code ในส่วนนี้คัดจาก Java source เดิมโดยตรงและคงข้อความตามต้นฉบับ ใช้เลขบรรทัดที่ระบุหน้าทุก snippet เพื่อตรวจเทียบ business behavior, SQL, transaction และ error handling

J1. ExportImpactStoreFlowToBPM.java — lines 9-17

รายการ	รายละเอียด
Source file	fcsJar/src/th/co/gosoft/fgi/main/ExportImpactStoreFlowToBPM.java
Original lines	9-17
Responsibility	Legacy main entrypoint for exporting impact-store flow data.

```
public class ExportImpactStoreFlowToBPM {
private static ApplicationContext context = new FileSystemXmlApplicationContext("config.xml");
private static ExportController exportController = (ExportController) context.getBean("exportController");
public static void main(String[] args){
LogUtils.info(ExportImpactStoreFlowToBPM.class,"Start => export IMPACT_STORE_INFO to BPM ");
exportController.exportImpactStoreFlowToBPM();
LogUtils.info(ExportImpactStoreFlowToBPM.class,"End => export IMPACT_STORE_INFO to BPM");
}
}
```

J2. ExportController.java — lines 518-529

รายการ	รายละเอียด
Source file	fcsJar/src/th/co/gosoft/fgi/controller/ExportController.java
Original lines	518-529
Responsibility	Build impact-store BPM payload, write file, upload, backup, notification.

```
public void exportImpactStoreFlowToBPM() {
String fileName = "";
  Calendar calStart = Calendar.getInstance(Locale.US);
EmailTemplateInformStatusBean informBean = new EmailTemplateInformStatusBean();
informBean.setSystemCode(FgiConstant.SYSTEM_CODE);
informBean.setJobName(FgiConstant.JOB_NAME_EXPORT_IMPACT_STORE_FLOW_TO_BPM);
informBean.setStartDateTime(calStart.getTime());
informBean.setImportFileName("");
List<Map> errorProcessListMap = new ArrayList<>();
try {
fileInputStream = th.co.gosoft.fcs.utils.FileUtils.openFile("ApplicationResources.properties");
propertiesFile.load(fileInputStream);
```

J2. ExportController.java — lines 646-657

รายการ	รายละเอียด
Source file	fcsJar/src/th/co/gosoft/fgi/controller/ExportController.java
Original lines	646-657
Responsibility	Build impact-store BPM payload, write file, upload, backup, notification.

```
LogUtils.info(getClass(),"--Transaction is Rollback--");
LogUtils.error(getClass(),ex);
informBean.setErrorMsg("" + ex);
informBean.setStatus(FgiConstant.JOB_STATUS_FAIL);
return new Boolean("false");
    }
    }});
if(!result){
return "";
}
return fileName;
}
```

J3. ExportJdbc.java — lines 1654-1665

รายการ	รายละเอียด
Source file	fcsJar/src/th/co/gosoft/fgi/dao/jdbc/ExportJdbc.java
Original lines	1654-1665
Responsibility	Query impact-store rows eligible for workflow export.

```
public List<Map> queryImpactStoreToBPM(){ // to BPM
StringBuilder sql = new StringBuilder();
sql.append(" select si.impact_store_info_id, si.storecode_i, si.name_i, si.opendate_i, si.zone_i, ");
sql.append(" si.branchtype_i, si.juristic_id_i, si.juristic_name_i, si.url_all_map, si.impact_type, ");
sql.append(" si.dv_emp_id, si.dv_fname_th, si.dv_lname_th, si.dv_fname_en, si.dv_lname_en, si.dv_email, ");
sql.append(" si.gm_emp_id, si.gm_fname_th, si.gm_lname_th, si.gm_fname_en, si.gm_lname_en, si.gm_email, ");
sql.append(" si.avp_emp_id, si.avp_fname_th, si.avp_lname_th, si.avp_fname_en, si.avp_lname_en, si.avp_email, ");
sql.append(" si.SBP_DV_EMAIL1, si.SBP_DV_EMP_ID1, si.SBP_DV_FNAME_EN1, ");
sql.append(" si.SBP_DV_FNAME_TH1, si.SBP_DV_LNAME_EN1, si.SBP_DV_LNAME_TH1, si.SBP_GM_EMAIL1, si.SBP_GM_EMP_ID1, ");
sql.append(" si.SBP_GM_FNAME_EN1, si.SBP_GM_FNAME_TH1, si.SBP_GM_LNAME_EN1, si.SBP_GM_LNAME_TH1, ");
sql.append(" si.SBP_VP_EMAIL1, si.SBP_VP_EMP_ID1, si.SBP_VP_FNAME_EN1, si.SBP_VP_FNAME_TH1, ");
sql.append(" si.SBP_VP_LNAME_EN1, si.SBP_VP_LNAME_TH1, ");
```

J3. ExportJdbc.java — lines 1681-1692

รายการ	รายละเอียด
Source file	fcsJar/src/th/co/gosoft/fgi/dao/jdbc/ExportJdbc.java
Original lines	1681-1692
Responsibility	Query impact-store rows eligible for workflow export.

```
sql.append(" and not exists ( ");
sql.append(" select rd.transaction_pk ");
sql.append(" from fgi_confirm_receive_data rd ");
sql.append(" where rd.data_name = 'IMPACT_STORE' ");
sql.append(" and rd.transaction_pk = si.impact_store_info_id ");
sql.append(" and rd.month = si.compensate_month ");
sql.append(" and rd.year = si.compensate_year ");
sql.append(" ) ");
List<Map> impactStoreList = jdbcTemplate.queryForList(sql.toString());
LogUtils.info(getClass(),"query FGI_IMPACT_STORE_INFO for export : "+impactStoreList.size());
return impactStoreList;
}
```