Claims IQ Core — Document Correction Pipeline

Architecture Overview

```
┌─────────────────┐   ┌──────────────────────┐   ┌─────────────────┐
┌──────────────────┐
│ Claim PDF(s)  │───▶│ Claims IQ Core AI │───▶│ Correction │───▶│ PDF Processor │
│ (FNOL, Invoice,│   │ Analysis Engine │   │ JSON Payload │   │ (Nutrient API) │
│ Activity, etc.)│   │              │   │ (canonical) │   │          │
└─────────────────┘   └──────────────────────┘   └─────────────────┘
└──────────────────┘
          │          │          │
          │          │          ▼
          │     ┌──────┴───────┐ ┌──────────────────┐
          │     │ Adapter Layer │   │ Corrected PDF │
          │     │ (Nutrient   │   │ + Annotations │
          │     │  translator) │   └──────────────────┘
          │     └──────────────┘
          ▼
┌──────────────────────┐
│ Cross-Document     │
│ Validation Engine   │
│ (compares fields    │
│  across all docs)   │
└──────────────────────┘
```

Schema Design Principles

1. Product-Agnostic Canonical Format

The correction JSON is **Claims IQ's own standard** — not Nutrient-specific. A thin adapter layer translates to whatever PDF processor is used. This means:

- Swap Nutrient for another tool without changing the AI pipeline
- Version the schema independently
- The format becomes defensible IP

2. Three Correction Capabilities

Capability	Schema Location	Purpose
Text Replacement	`documents[].corrections[]`	Fix typos, dates, phone numbers, values directly in the PDF

Capability	Schema Location	Purpose
Annotations	`documents[].annotations[]`	Non-destructive highlights, comments, flags for human review
Cross-Document Validation	`cross_document_validations[]`	Flag inconsistencies between related claim documents

3. Dual Location Strategy

Every correction and annotation includes a `location` object with two approaches:

- **`bbox`** — Precise coordinates (when OCR or PDF parsing provides them)
- **`search_text`** — Text-based search fallback (more resilient to PDF layout variations)

This is critical because insurance PDFs come from dozens of different systems with wildly different layouts. The search_text approach ensures corrections work even when coordinates shift between document versions.

4. Confidence & Human-in-the-Loop

Every correction carries:

- **`confidence`** (0.0–1.0) — How certain the AI is
- **`requires_human_review`** — Whether to auto-apply or queue for adjuster review
- **`evidence`** — Where the correct value was sourced from

This is essential for insurance compliance. Carriers need an audit trail showing *why* a document was modified and *who/what* approved it.

Correction Types

Type	Description	Example
`typo`	Spelling or character error	"Sampsom" → "Sampson"
`date_error`	Invalid or incorrect date	"3/256/25" → "3/26/25"
`phone_format`	Malformed or truncated phone	"469-394-025" → "469-394-0205"
`name_mismatch`	Name inconsistency across docs	Variant spellings of insured name
`address_error`	Address inconsistency or typo	Missing zip code, wrong street

Type	Description	Example
numeric_error	Wrong number (amounts, counts)	Incorrect deductible amount
missing_value	Required field is blank	Missing claim number
format_standardization	Inconsistent formatting	Date format normalization
data_inconsistency	Value contradicts other data	$0 loss amount with $12K invoice

Annotation Types

Type	Visual	Use Case
highlight	Background color on text	Flag for attention
comment	Sticky note / popup	Add explanation or question
flag	Icon marker	System-generated alert
strikethrough	Line through text	Mark for removal
underline	Line under text	Emphasis

Severity Levels

Level	Meaning	Action
critical	Affects claim validity or payment	Must fix before processing
warning	Data quality issue, potential error	Should fix, review recommended
info	Cosmetic or informational	Optional, good practice

Cross-Document Validation

This is the highest-value feature. The engine compares key fields across all documents in a claim file:

Fields Validated:

- Claim number
- Policy number

- Insured name and contact info
- Date of loss
- Property address
- Loss amounts / payment amounts
- Adjuster information
- Coverage details

Recommended Actions:

- `auto_correct` — High confidence, apply the fix automatically
- `flag_for_review` — Uncertain, needs human decision
- `escalate` — Potential fraud indicator or major discrepancy
- `informational` — No action needed, just documenting consistency

Nutrient Integration Notes

Product Options to Evaluate

Product	Best For	Correction Types
Document Engine (server-side)	Headless PDF processing in pipeline	Text replacement, annotations, form fills
Web SDK	Browser-based adjuster review UI	Interactive annotations, approval workflow
DWS API	Cloud-hosted REST processing	All types via API calls

Adapter Layer Responsibilities

The adapter translates Claims IQ's canonical format to Nutrient's specific API:

1. **Text replacements** → Nutrient Instant JSON or redaction + overlay
2. **Annotations** → Nutrient annotation API (comments, highlights, notes)
3. **Cross-doc flags** → Custom annotation layer or separate report

Key Consideration

Nutrient's "Instant JSON" format (their annotation serialization) is a well-documented spec. The adapter should map our `annotation` objects to their Instant JSON format, while text corrections may need to use their content editor API or a redact-and-overlay approach depending on the PDF structure.

File Manifest

File	Purpose
`claimsiq_correction_schema.json`	JSON Schema (validation spec)
`sampson_claim_corrections.json`	Real example payload for claim 01009792907
`ARCHITECTURE.md`	This document