FULL OPERATIONS, DOCUMENTATION & MANAGEMENT GUIDE

for the Property Risk AI Evaluation System

The property-risk evaluation solution is built as a coordinated, two-workflow system inside n8n. It transforms raw property questionnaires into high-value risk intelligence, evaluates each hazard domain independently with an AI agent, and stores the full risk profile in SQL for underwriting, reporting, and long-term modeling.

The design reduces operational friction by breaking the job into predictable, safe phases. The first workflow receives raw survey data, assigns each question to a risk domain, batches domains to avoid AI timeouts, evaluates each domain independently, and produces a combined master risk report. The second workflow normalizes that data and writes it into SQL using a clean, auditable structure. Each domain's failure or correction is isolated, meaning the whole run is never lost.

This setup is easy to run, maintain, and hand off. Team members can re-run individual domain batches, inspect any intermediate outputs, or push updated mapping rules without breaking the larger system. The workflows log every step, making it simple for engineering, operations, or underwriting teams to collaborate. It is built for iterative improvement — new domains, new scoring models, and new schema elements can be added without system downtime.

In short, the system is stable, scalable, and operationally mature. It was built so teams can maintain it together — not as a fragile one-off automation. Everything about it supports shared ownership and clear accountability.

1. System Overview

There are two workflows:

Workflow A — Webhook JSON → AI Domain Analysis (Batch Mode)

This workflow receives a property questionnaire, expands it into Q/A items, maps each item to a risk domain, creates batches, evaluates each domain through an AI Agent, and assembles a combined master risk report.

Workflow B — Final Report → SQL Writer (Background Task)

This workflow receives the master risk report, normalizes it, breaks it into relational structures, and inserts it into SQL tables.

Both workflows are modular, well-separated, and built to be maintained by a team rather than a single operator.

---

2. End-to-End System Flow

1. Questionnaire Submission

The system starts when a field user submits a full property survey. Workflow A receives this JSON through a webhook. No assumptions are made; the workflow validates structure before proceeding.

2. Expansion into Q/A Items

Every question and answer is broken into its own individual object. This makes domain assignment, analysis batching, and future auditing far easier.

3. Domain Assignment

A ruleset maps each question to Flood, Fire, Wildfire, Security, Construction, Utilities, Maintenance, Business Continuity, CAT, or “Other.” Updating these rules is safe and easy.

4. Domain Grouping

All questions for a single domain are grouped. Each group is independent and can be evaluated without affecting the others.

5. Loop Execution & AI Evaluation

The Loop Over Items node runs each domain batch separately.
The AI Agent analyzes each domain and outputs structured JSON with:

- score
- band

- top weaknesses
- mitigation steps

Failures in one domain don’t affect others.

6. Combined Final Report

Once all domain evaluations complete, Workflow A merges them into a single JSON report covering:

- property details
- combined overall score
- domain-level evaluations

7. SQL Normalization & Insert

Workflow B takes that final JSON and inserts the evaluation into SQL:

- evaluation table
- domain table
- weaknesses table
- mitigation steps table

Everything stays neatly relational and easy to query.

---

3. Daily Operational Procedures

Here’s what a team member would check on a normal day.

Morning Quick Check

- Confirm last night’s executions ran without errors.

- Ensure all expected domains appeared in the combined report.
- Verify SQL insertions occurred (spot-check new rows).

During the Day

- New surveys come in → confirm clean inbound payloads.
- Domain mapping is still behaving as expected.
- AI latency and success rates remain stable.

Before End of Day

- Review DataTable entries and clean up any stale ones.
- Validate no domains were skipped or returned an `"error"` key.
- Export logs for long-term retention if needed.

---

4. Maintenance Cycle

Weekly

- Run a regression sample to verify the full pipeline.
- Review AI cost and timeout alerts.
- Test one domain manually through the Loop Over Items.

Monthly

- Review prompts and mapping rules.
- Clean or archive DataTable rows.

- Validate SQL health: indexing, row counts, relationships.

Quarterly

- Evaluate OpenAI model version upgrades.
- Version-control the workflows.
- Update documentation and onboarding materials.

---

5. WORKFLOW DOCUMENTATION (Node-by-Node Descriptions)

Each node is described in two paragraphs: what it does and how a team should think about it.

Workflow A — Webhook JSON → AI Domain Analysis (Batch Mode)

Below is documentation for every node in the workflow.

---

1. Webhook (Receive Survey)

What it does:
This node receives the full property questionnaire as a JSON payload. It validates that the payload is present and includes the expected keys (address, answers, property info). If the payload is malformed, the webhook returns a 400 response and stops the workflow. This prevents garbage data from polluting the downstream pipeline.

Team considerations:
Anyone maintaining this system should understand that this webhook is the entry gate. If the structure changes — new sections, renamed fields — this is the first place to adjust. The

webhook can also be cloned for staging environments without altering production. If operators report “survey not processed,” this node should be the first place checked.

---

2. Extract Questionnaire Answers

What it does:
This node parses the incoming JSON and expands the answers into a flat array where each item contains `{section, question, answer}`. This makes the data workable for domain assignment and later batching.

Team considerations:
Because all downstream steps rely on this flattened structure, this node is crucial. If a questionnaire format changes (new section numbering, new question fields), updating the extraction logic here keeps everything else stable. This node is “safe to edit” and is where onboarding engineers should begin when new versions of the questionnaire launch.

---

3. Domain Mapping (Code Node)

What it does:
This script reads each Q/A item and assigns it a risk domain. Rules are based on section numbers, keywords, or explicit question IDs. The output is each Q/A with an added `domain` key.

Team considerations:
This is the node most likely to change as your risk framework evolves. Treat it like a “controlled vocabulary.” If you need a new domain, add it here. If a question is mis-mapped, the entire downstream chain will mis-score the property, so keep this node reviewed quarterly. It’s also a good place to add logging if the team needs to audit mapping decisions.

---

4. Group by Domain (Grouping Node)

What it does:
After mapping, this node groups all Q/A items by their assigned domain. It produces an array where each element contains a domain name and an array of questions belonging to that domain.

Team considerations:
This node is where structural issues surface. If a domain suddenly has zero questions, or all questions fall under “Other,” that indicates mapping errors upstream. Team members debugging

missing domains should inspect this node's output first. It's also the choke-point that keeps batches independent.

---

5. Loop Over Items (Batch Processor)

What it does:
This node loops over each domain packet one at a time. It isolates each batch to avoid AI timeouts or overloading the model. The loop triggers one Analysis Agent call per domain.

Team considerations:
The loop is the reliability backbone. When updating workflow behavior, do not change its mode from "iterate over items." Operators should know that domain failures are isolated — if one domain fails, the others complete normally. Team members can re-run a single domain without reprocessing the entire survey.

---

6. AI Analysis Agent (Domain Evaluator)

What it does:
For each domain batch, the AI agent receives a structured JSON containing all questions for that domain. The prompt instructs the model to output structured JSON containing the domain score, band, weaknesses, and mitigation steps. The agent must never produce prose.

Team considerations:
This node represents the dynamic intelligence of the system. Since the model updates over time, team members should periodically re-test results and ensure prompt compliance. When the model fails to produce valid JSON, error branching allows re-running only that domain. If costs spike or latency increases, adjust token limits or fine-tune the batching rules.

---

7. Validate JSON Output (Structured Output Parser)

What it does:
This node checks the AI output against a strict schema. If anything violates the structure — missing keys, unexpected commas, strings not arrays — the node fails and the loop marks the domain as invalid.

Team considerations:
Operators should treat this node as a safety net. If a domain repeatedly fails parsing, inspect both the prompt and the raw AI output in the Execution details. When onboarding team

members, emphasize that this node protects the system from bad AI responses and is essential for maintaining stability downstream.

---

8. Insert Into DataTable (Temporary Storage)

What it does:
Each validated domain result is inserted as a row into the n8n DataTable. The DataTable stores domain evaluations until the loop completes and the final report can be assembled.

Team considerations:
Team members should know that this is the temporary holding zone. If domain results appear missing at final merge time, check this table. Clearing stale rows periodically keeps performance stable. This table acts like a staging database; operators can inspect it to verify that all batches succeeded before merging.

---

9. Retrieve Domain Rows (Get All Rows)

What it does:
After all domain evaluations finish, this node pulls every row from the DataTable to assemble the full list of domain evaluations.

Team considerations:
If not all domains appear here, it means they were never written to the table — usually an AI failure or mapping issue. Troubleshooting always begins by comparing expected domain count vs retrieved rows. This step is critical before final report creation.

---

10. Retrieve Property Info

What it does:
This node extracts the property address, building type, and notes from the original webhook payload. These values anchor the final report.

Team considerations:
If property details appear blank in the final JSON, this is the node to inspect. Any changes in the incoming webhook structure must be mirrored here. It keeps the report readable and tied to a real location.

---

11. Generate Combined Final Report (Code Node)

What it does:
This script merges all domain rows and property details into the final JSON structure with:

- property
- overall
- domains[]

The overall score is calculated from domain-level scores.

Team considerations:
This is the most delicate node in the workflow. Any schema changes (adding new keys, adding new domain fields) must be updated here. Operators should confirm that the final JSON is valid before sending to SQL. If AI outputs change, update this node accordingly.

---

12. POST Final Report to Workflow B

What it does:
This node sends the complete combined JSON to the SQL-writer workflow using an authenticated HTTP call.

Team considerations:
If SQL records are missing, this node is a likely failure point. Check:

- endpoint
- authentication headers
- payload structure

It’s the bridge between the AI pipeline and the storage layer.

Workflow B — Final Report → SQL Writer (Background Task)

---

1. Webhook (Receive Final JSON)

What it does:
This workflow begins when the combined JSON arrives. The webhook accepts the payload and confirms that mandatory fields are present (property, overall, domains).

Team considerations:
This is the first checkpoint in Workflow B. If SQL tables appear empty, confirm that this webhook is receiving data from Workflow A. Use Execution history to track any missing calls.

---

2. Normalize JSON (Code Node)

What it does:
Breaks the final JSON into three categories:

- main evaluation
- domain entries
- sub-rows (weaknesses and mitigations)

This produces predictable arrays regardless of property type or number of domains.

Team considerations:
Everything inserted downstream depends on the shape created here. If SQL begins receiving NULLs or incomplete rows, validate this node first. Any schema changes used by Workflow A must be reflected here.

---

3. Insert Evaluation Row

What it does:
Inserts the primary evaluation row into SQL: address, building type, notes, final score, and risk band.

Team considerations:
Operators must ensure foreign keys (EvaluationId) are returned correctly. If SQL rows appear without associated domains, this is the insertion to check. Also confirm that string sanitation or escaping is applied correctly.

---

4. Extract Domain Rows

What it does:
Loops through each domain and prepares rows for insertion: domain name, domain score, risk band, record ID linking back to the evaluation.

Team considerations:
If domain rows are missing in the SQL output, this node is the place to debug. A mismatch here usually means the combined JSON didn't include all expected domains.

---

5. Insert RiskDomainResult Rows

What it does:
Inserts each domain into SQL. Returns the new DomainId for use by sub-tables.

Team considerations:
Teams should confirm index usage and table constraints here. If the domain table grows large, add indexing on EvaluationId to maintain performance.

---

6. Extract Weakness Items

What it does:
Pulls all topWeakness items across domains, tagging them with their DomainId for insertion.

Team considerations:
If weaknesses disappear from SQL, it's usually because the domain reference wasn't passed through correctly. This node ensures proper foreign key linking.

---

7. Insert Weaknesses

What it does:
Inserts each weakness into SQL with fields for questionId and issue text.

Team considerations:
If SQL throws errors about field size or inconsistent casing, this is where to add normalization. Team members should sanitize strings here for long-term compatibility.

---

8. Extract Mitigation Steps

What it does:
Pulls all 30-day mitigation steps across domains, again mapping them to the correct DomainId.

Team considerations:
If the business wants to analyze mitigation trends, this is the lifeblood table. Confirm stable linking and completeness.

---

9. Insert Mitigation Steps

What it does:
Inserts each mitigation step into SQL. Ensures long-text fields are stored correctly.

Team considerations:
Large text fields can cause SQL errors depending on server settings. Team should check field types (e.g., NVARCHAR(MAX)) and confirm this node matches DB schema.

---

10. Return Success JSON

What it does:
Returns a confirmation payload so Workflow A knows the data was received successfully.

Team considerations:
If Workflow A shows repeated retries, the issue is often here: authentication, timeout, or malformed response. Always confirm the node returns 200 with valid JSON.

---

6. Handoff Strategy for New Team Members

For new engineers:

Start with the domain-mapping logic. It’s where most change happens. Understand the extraction, mapping, batching, and final merge steps before editing anything.

For risk operations staff:

Focus on DataTable inspection and domain-level outputs. This is where incorrect mappings or AI anomalies will surface.

For SQL/reporting analysts:

Focus on Workflow B and the relational structure. Queries should validate completeness: number of domains per evaluation, number of weaknesses, and number of mitigations.

For team leads coordinating the whole system:

Watch the handoff points — where one workflow ends and the other begins. If something goes wrong, it's almost always at a boundary:

- Webhook intake
- Domain mapping
- AI batch returns
- Final JSON merge
- SQL insert

The system is designed for collaborative ownership. No single point requires specialized knowledge.

---

7. TROUBLESHOOTING PLAYBOOK

Issue: Only one domain shows up

Cause: Missing or overwritten DataTable rows.
Fix: Check the DataTable after each AI call. Ensure each Insert node uses `append`, not `replace`.

Issue: "Model output doesn't fit required format"

Cause: AI deviated from JSON schema.
Fix: Strengthen the Structured Output Parser, re-run only the failed domain batch.

Issue: SQL rows partially missing

Cause: IDs not passed through normalization step.
Fix: Inspect the Normalize node's arrays and confirm each domain receives EvaluationId.

Issue: Workflow A shows complete, but Workflow B never fires

Cause: POST node misconfigured.
Fix: Test POST manually, confirm webhook URL, confirm JSON structure.

Issue: Domain mapping wrong

Cause: Rules outdated.
Fix: Update mapping code, rerun regression survey.

Issue: Timeouts in AI Agent

Cause: Payload too large.
Fix: Reduce per-domain batch size or tighten prompt.

Issue: Duplicates in DataTable

Cause: AI retries created extra rows.
Fix: Clear DataTable and re-run. Add uniqueness rules if needed.

Installing N8N Infrastructure Using Docker

Prerequisites

- A Cloudflare account and a registered domain
- Docker Desktop installed on your machine (Mac, Windows, or Linux)
- Cloudflared installed for tunnel creation
- Cribops CLI tool for simplified N8N installation
- Terminal or Command Prompt access

Step 1: Register a Domain with Cloudflare

1. Navigate to https://www.cloudflare.com and log in or create an account.
2. Go to the “Domain Registration” section in your dashboard.
3. Click on “Register a domain.”
4. Search for and purchase a domain name.
5. Once the domain is registered, it will appear in your Cloudflare dashboard for future configuration.

Step 2: Install Required Software

1. Docker Desktop

- Navigate to https://docs.docker.com/get-docker.
- Download Docker Desktop for your operating system.

- Install Docker Desktop and sign in with your Docker Hub account (Google, GitHub, or email).
- Confirm Docker is running before proceeding.

2. Cloudflared

- Navigate to: https://developers.cloudflare.com/cloudflare-one/connections/connect-apps/install-and-setup/installation
- Download and install the Cloudflared client for your OS:
  - macOS via Homebrew:

```
None
brew install cloudflared
```

  - Windows via the MSI installer
- Confirm installation:

```
None
cloudflared --version
```

3. Cribops CLI

- Visit: https://github.com/CloudBedrock/cribops-docs
- Scroll to the Downloads section and select the appropriate installer for your OS.
- macOS: Download and install the `.dmg` file.
- Windows: Download and run the `.msi` installer.
- Verify installation:

```
None
cribops-cli --help
```

Step 3: Run N8N Setup with Cribops CLI

1. Open your terminal or command prompt.
2. Display help to view available commands:

```
None
cribops-cli setup -h
```

3. Run the one-line installation command, replacing placeholders:

```
None
cribops-cli setup --cloudflare --tunnel-name <TUNNEL_NAME>
--domain <TUNNEL_NAME>.<YOUR_DOMAIN>
```

4. **Example:**

```
None
cribops-cli setup --cloudflare --tunnel-name test --domain
test.yourdomain.com
```

5. Follow the prompts to authenticate and verify the domain.

Step 4: Start the N8N Infrastructure

Execute the following command to start all required containers in the background:

```None
docker compose up -d
```

To restart services if needed:

```None
docker compose down
docker compose up -d
```

Step 5: Access N8N in Browser

1. Copy the access URL displayed in the terminal (e.g., `https://test.yourdomain.com`).
2. Open the URL in a browser.
3. Complete the N8N setup form with your name, email, and password.
4. Submit to create your administrator account.

Step 6: Activate N8N License

1. Enter your email address to request a free license key from N8N.
2. Wait for the license key email.
3. In the N8N interface, go to **Usage & Plan** → **Enter Activation Key**.
4. Paste the activation key and click “Activate.”

Installation Complete

N8N is now fully installed and accessible over HTTPS with Docker-managed infrastructure and Cloudflare Tunnel integration.