Experiment Brief

Encoding Health Workflows as Open, Machine-Readable Assets (using n8n)

Purpose

Health systems rely on complex workflows (maternal care, immunization, chronic disease follow-up), but today these workflows are:

- Hard-coded inside applications
- Duplicated across programs and vendors
- Difficult to adapt, reuse, or share

We want to explore a foundational idea:

> **Can common health workflows be encoded as machine-readable, application-agnostic configurations, using a tool like n8n, and published as open Digital Public Goods (DPGs)?**

If successful, these workflows could:

- Be reused across programs and vendors
- Be translated (manually or via AI) into other workflow engines
- Act as shared public infrastructure, not app-specific logic

This is an **exploratory experiment**, not a production build.

What are we trying to do

Using **n8n as a reference tool**, encode **three representative healthcare workflows** and assess:

1. How clearly workflows can be expressed
2. How decoupled they are from applications
3. How reusable and portable they feel
4. How external applications would participate in them

Think of n8n as a **workflow description medium**, not the final platform.

Assumed building blocks (conceptual only)

We do not need to build these—assume they exist and interact via APIs/events.

Concept	Meaning
PHP (Personal Health Profile)	Unified patient record
Workflow Engine	Orchestrates steps based on events
UTM (Unified Task Manager)	Task queue for CHWs, nurses, doctors
CHW Apps	Maternal, NCD, outreach apps
Facility EMR	Doctor / nurse systems
Dashboards	Supervisor / program views

In n8n terms: webhooks, conditionals, functions, schedules, API calls.

Use cases to encode

1. High-Risk Pregnancy + Diabetes Screening

Intent

Automatically trigger diabetes screening for pregnant women and ensure referral and follow-up if high-risk. Additional details here.

Trigger

- Pregnancy registration event

Logic

- Blood sugar ≥ threshold → high-risk pathway

Actions

- Create CHW screening task
- Create doctor referral task
- Notify patient
- Initiate follow-up workflow

Focus

- Branching logic clarity
- Sub-workflow invocation
- Readability as a shared artefact

2. Immunization – Cross-System Tracking

Intent

Ensure facility immunization updates drive timely community outreach. Additional details here.

Trigger

- Immunization recorded in Facility EMR

Actions

- Sync record to child profile
- Compute next due vaccine
- Create CHW outreach task

Focus

- Parameterized schedules
- Reuse across programs
- Config vs hard-coded logic

3. Chronic Disease – Hypertension Follow-Up

Intent

Ensure continuity of care through automated recurring follow-ups. Additional details here.

Trigger

- Hypertension diagnosis recorded

Actions

- Create monthly follow-up tasks
- Capture BP readings
- Sync back to doctor

Focus

- Recurrence handling
- Lifecycle management
- Generalizability to other NCDs

Critical exploration axis: External application participation

Beyond workflow logic, explicitly explore:

> **How does an external application participate in a shared workflow without tight coupling?**

We are *not* integrating real apps—only clarifying the contract.

For each workflow, answer:

1. Triggering

- What event does an app emit?
- What is the minimum payload?

2. Task consumption

- How does an app know a task is assigned?
- What action is expected?

3. Reporting outcomes

- How does the app report completion, partial completion, or failure?
- How does the workflow handle delays or missing updates?

4. Observability

- What workflow state can the app see?
- What is intentionally hidden?

Avoid:

- App-specific assumptions
- Direct DB access
- Dependency on n8n internals

Call out:

- “n8n convenience” vs “generalizable pattern”

What success looks like

We are **not** optimizing for scale, security, or performance.

We care about:

- **Expressiveness** – is the workflow clear?
- **Readability** – can non-developers reason about it?
- **Reusability** – can patterns repeat across programs?
- **Exportability** – JSON, versionable, shareable
- **Translation potential** – can this be re-interpreted elsewhere?

Expected outputs

1. **n8n workflow exports** (JSON)
2. **Short README per workflow**:
    - Trigger
    - Core logic
    - Key assumptions
1. **Simple interaction sketch**:

  - App → Workflow
  - Workflow → App
1. **Minimal interface definitions**:
   - Trigger event shape
   - Task payload shape
   - Completion/update signal
1. **Reflections**:
   - What worked well?
   - What felt limiting?
   - Where did n8n help or hinder?
1. **Recommendation**:
   - Is n8n viable as a reference workflow DSL?
   - What would we standardize if this became a DPG?

Closing note

This is an **exploration, not an evaluation**.

If something feels awkward, let's document it—that insight is as valuable as a working flow.

If done well, this experiment helps answer a larger question:

> *Can workflows themselves become shared public infrastructure for health systems?*

Appendix

Use Case 1: High-Risk Pregnancy with Community-Level Diabetes Screening

Scenario: Pregnant woman registered in Maternal App → triggers diabetes screening via NCD community app → referral to facility doctor if high-risk → doctor confirms high-risk pregnancy → follow-up tasks pushed to CHW.

Configuration:

- **Condition:** Pregnant woman registered → triggers diabetes screening; blood sugar ≥ threshold → referral.
- **Actions:**
  - Create referral task in facility Maternal/NCD system
  - Notify patient via SMS/IVR

    - Doctor evaluates, confirms high-risk pregnancy, sets follow-up plan, updates supervisor dashboard
- **Mapping:**
    - Community: Maternal App ↔ NCD App
    - Facility: Maternal EMR ↔ NCD module
    - Supervisor: Unified Dashboard

Example Execution Flow:

1. **Maternal App** (CHW) registers the woman as pregnant.
2. PHP update triggers the Workflow Engine to initiate a pregnancy workflow.
3. Workflow Engine creates a Diabetes Screening Task in UTM.
4. **NCD App** (CHW) accepts the task and performs screening.
5. **NCD App** updates screening results into the PHP.
6. PHP update triggers the Workflow Engine to evaluate results.
7. **If high sugar:** Workflow Engine creates a referral task in UTM for the doctor.
8. **If high sugar:** Workflow Engine notifies the patient via SMS/IVR.
9. Facility EMR (Doctor) accepts the referral task from UTM.
10. Doctor retrieves the full PHP (pregnancy + screening + referral).
11. Doctor updates the follow-up care plan in PHP.
12. PHP changes trigger the Workflow Engine to initiate a follow-up workflow.
13. Workflow Engine assigns a follow-up task to the CHW (typically **Maternal App** for pregnancy follow-up).
14. Dashboard & Analytics updates with the high-risk pregnancy + diabetes follow-up.

Use Case 2: Immunization – Cross-System Tracking

Scenario: Child immunization records updated at a facility; CHW needs visibility for outreach.

Configuration:

- **Condition:** Immunization event recorded in facility EMR.
- **Action:** Sync record to child profile in CHW App + generate pending task for next vaccine.
- **Mapping:** Facility Immunization System ↔ CHW App

Execution Flow (with Unified Building Blocks):

1. Nurse records the child’s immunization in the Facility EMR.

2. The record syncs to the Unified PHP.
   2a) PHP triggers the Workflow Engine to compute the next vaccine due date.
3. Workflow Engine publishes a pending "next vaccine" task to UTM for **Maternal App**.
4. **Maternal App** fetches the task for household visits.
5. CHW completes the visit and updates immunization status in PHP.

Use Case 3: Chronic Disease – Hypertension Follow-Up

Scenario: Patient diagnosed with hypertension → monthly follow-ups needed by CHW.

Configuration:

- **Condition:** Hypertension diagnosis recorded in EMR.
- **Action:** Auto-generate recurring monthly follow-up tasks for CHW.
- **Mapping:** Facility EMR ↔ CHW App

Execution Flow (with Unified Building Blocks):

1. Doctor diagnoses hypertension and updates the Facility EMR.
2. Workflow Engine detects the diagnosis event.
   2a) Workflow Engine creates recurring monthly follow-up tasks in UTM for **NCD App**.
3. **NCD App** fetches the current due task.
4. CHW records home BP readings in **NCD App** → PHP updates.
5. Data in PHP is visible in the EMR for the doctor.