CLOUDNIX TECHNICAL REPORT · CNX-TR-2026-04

# Engineering Configurable Workflow Platforms

Ajmal Abubacker, Jeya Kirubakaran, Avinash KS, Kiran Reddy and Avinash Nishant

*Cloudnix Software Labs, Bengaluru, India*

CNX-TR-2026-04 · Version 1.0 · November 2026

**Abstract**

Business process software tends to end as a rigid product that does not fit, or as custom code that only developers can change. This report describes a workflow platform engineered so that teams configure their own processes without a release: a process model of departments, projects, workflows and stages with allowed transitions and versioned history; stage forms built from twenty-six field types with a restricted, server-evaluated formula language; two service-level clocks per stage; five views and dashboards over the same cards; function-level permissions and a field-level audit trail; imports in which a language model proposes column mappings under a strict schema and a person confirms them; and field operations on offline-first apps with per-row synchronisation, geofenced visits, inspection questionnaires and reconciled payouts. We conclude with an anonymised case study, a roadmap and a 20-point checklist.

**Keywords:** *business process management; workflow; low-code; forms; formulas; SLA; permissions; audit; data import; JSON Schema; offline-first; geofencing; field operations*

---

**Contents**

1. Executive summary
2. Where process platforms struggle
3. Reference architecture
4. The process model
5. Forms and formulas
6. Stage-level SLAs
7. Views and dashboards
8. Permissions and audit
9. Notifications and API
10. Imports with AI-assisted mapping
11. Offline field capture
12. Geofenced visits
13. Inspection questionnaires, reports and payouts
14. Security and operations
15. Case study: a property-services network
16. Implementation roadmap
17. Checklist: 20 questions
18. Glossary
19. References

## 1 Executive summary

Business process software tends to end in one of two places: a rigid product that does not fit how teams work, or custom code that only developers can change. Both fail as processes evolve. This report describes how we engineered a workflow platform in which teams configure their own processes, stages, transitions, owners, forms, formulas and deadlines, without a release; how managers see the same work through several views and dashboards; how permissions and a field-level audit trail keep shared processes safe; how data is imported with AI-assisted mapping confirmed by a person; and how the same processes are carried into

the field on apps that work without signal, with geofenced visits, inspection questionnaires and validated payouts.

**Principles.**

1. **The process is data.** Stages, transitions and forms are rows, not code.
2. **History is never overwritten.** Every stage visit and form value is versioned.
3. **Deadlines belong to stages.** Each stage has its own response and resolution targets.
4. **One dataset, many views.** Boards, tables, sprints and dashboards read the same cards.
5. **Every change is attributable.** Function-level permissions and field-level audit.
6. **People confirm what AI proposes.** AI maps imports; a person approves the map.
7. **The field works offline.** Capture first, synchronise row by row later.

## 2 Where process platforms struggle

| Problem | Symptom | Pattern (chapter) |
|---|---|---|
| Processes keep changing | Every change waits for a developer | Process as configuration (4) |
| Different data per stage | One giant form, half irrelevant | Stage forms and formulas (5) |
| Late work is invisible | “Overdue” without knowing where it is stuck | Stage-level SLAs (6) |
| Everyone exports to spreadsheets | Reporting outside the system | Views and dashboards (7) |
| Sensitive data in shared processes | Too much access, no trail | Permissions and audit (8) |
| Data migration stalls | Column mapping by hand, repeatedly | AI-assisted import (10) |
| Field work disconnected | Paper forms, no proof of visit | Offline capture and geofencing (11, 12) |

## 3 Reference architecture

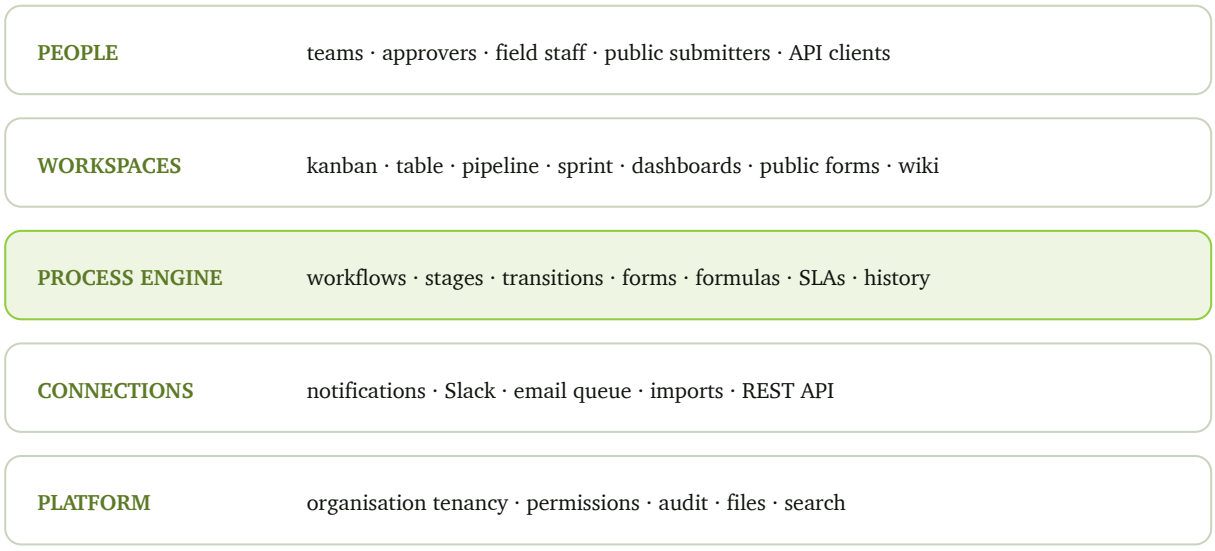



**Figure 1.** The five layers of the workflow platform.

The platform is multi-tenant: every table carries the organisation, and the organisation in the session scopes every query. A demonstration organisation can be routed to a separate database so that prospects never share storage with customers.

## 4 The process model

The hierarchy is organisation → department → project → workflow → stage → card. A workflow names its start stage; each stage lists its allowed next stages, belongs to one of three groups (new, in progress, closed), names the teams and users accountable for it, and can be published as a public form.

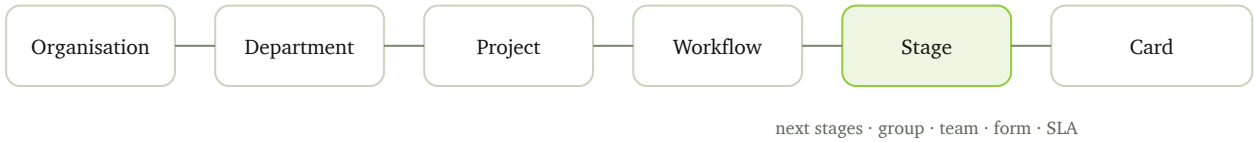



**Figure 2.** The process hierarchy.

### 4.1 *History*

Each card keeps a row per stage visit: who it was assigned to, when, estimated and actual hours, and whether it is the latest visit. Form values are versioned the same way, so the value a field held in an earlier stage is never lost when the card moves on.

### 4.2 *Templates*

Template workflows live in a template organisation and are copied when a project is created; more than a hundred ready-made processes cover sales, hiring, finance, engineering, service and operations.

---

**Design note.** Transitions are an allowed-next list per stage rather than a general rules engine. That keeps processes understandable to the people who design them; conditional routing is added deliberately where a process needs it.

---

## 5 Forms and formulas

Stage forms are built from twenty-six field types over six storage types (text, integer, date, datetime, decimal, long text):

| Group | Types |
|---|---|
| Text | Single line, multi line, rich text, URL, name |
| Numbers | Number, decimal, percentage, currency, formula |
| Choices | Checkbox, radio, dropdown, single select, multi select, tags |
| Dates | Date, date time |
| People | Owner, assigned to, project team |
| Advanced | Lookup, table, file upload, image, section separator |

Fields carry rules: mandatory, minimum and maximum length, prefix and suffix, initial value and visibility.

### 5.1 *Formula grammar*

```
expression := number | {{field}} | function(args) | expression op expression
functions  := if, sum, avg, min, max, abs, round, floor, ceil, sqrt, pow
dates      := date_diff(start, end, unit)   unit ∈ { Y, M, D, YM, YD, MD }
              today()                        # only inside date_diff
blanks     := numeric fields left empty count as 0
```

**Listing 1.** Example exchange or schema.

Formulas are evaluated on the server when a card is saved and previewed in the browser while editing. Restricting date functions to `date_diff` keeps expressions safe and deterministic.

## 6 Stage-level SLAs

Each stage may define two thresholds in hours ($-1$ disables either):

| Clock | Breached when |
|---|---|
| First response | hours in stage ≥ target and the stage's form has not been updated in this stage visit |
| Resolution | hours in stage ≥ target |

```
hours_in_stage = (now − stage_assigned_at).days × 24 + (now − stage_assigned_at).hours
state = resolution_breach ? RED : first_response_breach ? AMBER : ON_TIME
```

**Listing 2.** Example exchange or schema.

Breaches show on every view as a coloured edge with the amount exceeded. Clocks run in calendar hours; business-hours calendars, holidays and pause states are natural extensions for service desks with defined working hours.

## 7 Views and dashboards

Every project offers five views over the same cards: kanban, table (with column choice), pipeline, sprint and dashboard. Filters and sorts can be saved and shared. Dashboard widgets aggregate card and form fields with five functions (sum, count, minimum, maximum, average) into seven chart types (bar, column, pie, line, area, table, counter). Cards can be exported as PDF.

## 8 Permissions and audit

Rights are defined per function and mode and checked at each guarded function (186 call sites). User types are super user, organisation administrator, member, submitter and public user; project teams add administrator, member and public roles. Every change to a card, form value, stage or setting is written to the audit trail with the entity, field, old value, new value, action (added, updated, deleted), actor and time, across twenty-seven kinds of record [5].

## 9 Notifications and API

Six notification types (assignment, mention, project comment, card activity, sprint update, announcement) go in-app and to Slack [7]; email is sent through a queued sender. A REST API described in OpenAPI 3.1 [4] exposes departments, projects and cards with a bearer key per user, stored hashed.

## 10 Imports with AI-assisted mapping

Four connectors bring data in: CSV files, spreadsheets [8] and two HR and accounting systems, through a six-step wizard. The hardest step, matching source columns to destination fields, is proposed by a language model under a strict JSON Schema [6]:

```
{ "mappings": [ { "source_field": string,
                  "destination": "TITLE" | field_id | "NEW" | "SKIP",
                  "display_type": one of 10 importable types } ] }
rules: exactly one TITLE (plain text, not multi-line); ids, images and technical fields → SKIP
```

**Listing 3.** Example exchange or schema.

A person reviews and edits the proposed map before it is saved. Scheduled syncs reuse the saved map without calling the model; destinations that point at deleted fields are reset for review. A heuristic, not the model, guesses the unique match key (id, email, SKU).

## 11 Offline field capture

Field apps are hybrid (one codebase for Android and iOS [9]). An inspector parks an order on the device: the order, its documents and every page image are copied into a local database [10] and a file tree. Records carry flags for synced, created locally and updated.

1. Sync sends one multipart request with changed rows and files.
2. The server replies per row: order, document and page status.
3. Rows marked OK are removed from the device with their files.
4. Failures remain as a checklist the inspector can retry.

Documents are captured with an edge-detecting scanner and converted to PDF; photos are cropped and annotated on the device before upload.

## 12 Geofenced visits

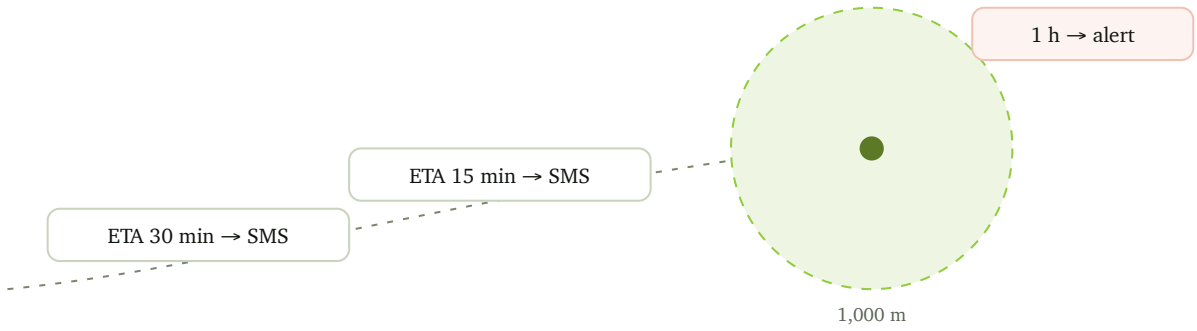



**Figure 3.** A visit tracked against the property's geofence.

```
position = getCurrentPosition(highAccuracy, maxAge 1 h)       # [2]
eta      = distance_matrix(position, property)                 # [3]
if eta ≤ 30 min and not sent_30: sms(customer, eta); sent_30 = true
if eta ≤ 15 min and not sent_15: sms(customer, eta); sent_15 = true
d = distance(position, property)
if d ≤ 1000 m and not arrived:  record_arrival(); arrived = true
if arrived and d > 1000 m:       record_departure()
if arrived and now − arrival ≥ 1 h: alert(office)
```

**Listing 4.** Pseudocode.

Flags are kept per order on the device so each message is sent once.

## 13 Inspection questionnaires, reports and payouts

Each client organisation has its questionnaires; questions belong to sections and are single choice, multiple choice, text or number, each with an option list and a photos-required flag. Answers carry their photos. Questionnaire status moves through new, in progress, completed, rejected and cancelled. A scheduled job renders the PDF report from answers and photos within a minute of submission.

Inspection types carry fees; inspector fees are set per inspector, county and type. Payout batches move through pending, approved and paid or failed, and a batch is rejected unless its total equals the sum of its orders' fees.

## 14 Security and operations

| Area | Control |
|---|---|
| Tenancy | Organisation on every row and query; separate database for demonstrations |
| Identity | Google sign-in [1]; OTP for submitters; secure, HTTP-only session cookies |
| Authorisation | Function and mode rights; 186 checks |
| API | Bearer keys per user, hashed at rest |
| Audit | Field-level, old and new values, 27 kinds of record |
| Uploads | File type checked on import |
| Operations | Queued email and Slack senders; nightly backups; daily log review |

## 15 Case study: a property-services network

*Anonymised. A property-services business coordinating inspections and document retrieval across many counties.*

### 15.1 *The challenge*

Orders arrived by email and file; independent inspectors and abstractors worked in places with poor coverage; clients expected consistent reports; and fees differed by county and job type, paid in bulk.

### 15.2 *What was built*

- Order intake from a mailbox and files, with searchable orders.
- Assignment by county coverage, with the property on a map.

- Offline parking of orders and documents on phones, with per-row synchronisation.
- Geofenced visits with ETA messages, automatic arrival and departure, and a long-stay alert.
- Client-specific questionnaires with marked-up photos, and PDF reports generated on submission.
- Fees by county and job type, and payout batches that must add up before they are paid.

### 15.3 *What it changed*

Office teams see when work actually happened, clients receive reports in one format, and payouts are reconciled before money moves.

## 16 Implementation roadmap

| Phase | Outcome |
|---|---|
| 1. Discovery | Processes mapped: stages, forms, owners, deadlines, reports |
| 2. Pilot | One department live from templates; imports confirmed |
| 3. Scale | Departments, dashboards, permissions, field apps |
| 4. Operate | Process reviews; SLA and audit reports; changes as configuration |

## 17 Checklist: 20 questions

| | |
|---|---|
| 1 | Can a team change its process without a release? |
| 2 | Is every stage visit kept, not overwritten? |
| 3 | Are form values versioned per stage? |
| 4 | Do stages have their own forms and owners? |
| 5 | Are formulas evaluated on the server? |
| 6 | Do stages have response and resolution targets? |
| 7 | Do breaches show on every view? |
| 8 | Do you need business-hours calendars? |
| 9 | Can managers answer questions without exporting? |
| 10 | Are rights checked per function? |
| 11 | Is every change audited with old and new values? |
| 12 | Are API keys personal and hashed? |
| 13 | Does a person confirm every AI-proposed import map? |
| 14 | Do scheduled imports reuse the confirmed map? |
| 15 | Do field apps work without signal? |
| 16 | Is synchronisation per row with a retry list? |
| 17 | Is arrival proven by location, not by a tap? |
| 18 | Are reports generated in one format for every client? |
| 19 | Do payouts reconcile before they are paid? |
| 20 | Who runs the platform after launch? |

## 18 Glossary

| | |
|---|---|
| **Stage** | A step in a workflow with its own form, owners and targets. |
| **Card** | One item of work moving through stages. |
| **First response** | The first update to a stage's form after a card arrives. |
| **Geofence** | A radius around a location used to detect arrival and departure. |
| **Parking** | Copying an order and its files to a device for offline work. |

## 19 References

1. D. Hardt (ed.). *The OAuth 2.0 Authorization Framework*. IETF RFC 6749, 2012.
2. W3C. *Geolocation API*. W3C Recommendation.
3. Google. *Distance Matrix API*, developer documentation.
4. OpenAPI Initiative. *OpenAPI Specification*, v3.1.0, 2021.
5. OWASP Foundation. *Application Security Verification Standard*, v4.0, chapter V7: Error Handling and Logging.
6. JSON Schema. *JSON Schema: Draft 2020-12*.
7. Slack Technologies. *Slack API: Incoming Webhooks and OAuth*, documentation.
8. Google. *Google Sheets API*, developer documentation.
9. The Apache Software Foundation. *Apache Cordova*, documentation.
10. SQLite Consortium. *SQLite Documentation*.

**About this report.** Prepared by the Cloudnix engineering team from systems we have designed, built and operated. Client and product names are withheld; illustrative values are marked as examples. Cloudnix Software Labs, Bengaluru, India · cloudnix.com. This report may be shared freely in its entirety.

| Document control | | | |
|---|---|---|---|
| Report | CNX-TR-2026-04 | Version | 1.0 |
| Issued | November 2026 | Status | Published |
| File | cloudnix-workflow-platforms-v1.0.pdf | | |