ClearPath RD

clearpathrd.com

Scheduled Jobs Specification

Version 1.0 — February 2026 — Confidential

What this document covers

- Scheduling architecture — node-cron running in-process in the Fastify API
- Job registry — all 5 jobs with schedule, file location, and estimated runtime
- 5 job specifications — each with trigger condition, steps, idempotency guard, and error handling
- Timezone handling — all jobs run in Mountain Time (Canada) by default
- Startup behaviour — what happens when the API restarts mid-job
- Observability — how jobs are logged and monitored at MVP scale
- Implementation checklist

1. Scheduling Architecture

All scheduled jobs run inside the Fastify API process using node-cron. There is no separate worker process, no Redis queue, and no external job scheduler. Jobs are registered in src/jobs/index.ts and started when the Fastify server starts.

> *Design constraint: node-cron jobs are in-process. If the Fastify API restarts, any in-flight job is interrupted and will run again at the next scheduled time. All jobs are designed to be idempotent — running twice has the same outcome as running once. This makes restart-induced double-runs safe.*

1.1 Registration Pattern

```
// src/jobs/index.ts
import cron from 'node-cron';
import { runEmailQueue } from './emailQueue';
import { runSessionExpiry, runCertificateExpiry } from './expiry';
import { runCnrppAlert } from './cnrpp';
import { runLabResultPoll } from './labPoll';
```

```
export function registerJobs() {
  cron.schedule('*/15 * * * *',  runEmailQueue,         { timezone:
'America/Edmonton' });
  cron.schedule('0 2 * * *',     runSessionExpiry,      { timezone:
'America/Edmonton' });
  cron.schedule('0 2 * * *',     runCertificateExpiry,  { timezone:
'America/Edmonton' });
  cron.schedule('0 2 * * *',     runCnrppAlert,         { timezone:
'America/Edmonton' });
  cron.schedule('0 6 * * *',     runLabResultPoll,      { timezone:
'America/Edmonton' });
}
```

```
// src/server.ts
import { registerJobs } from './jobs/index';
// After Fastify server starts:
registerJobs();
```

1.2 Timezone

All cron expressions use Mountain Time (America/Edmonton). This places nightly batch jobs at 02:00 MT, which is 04:00 ET and 09:00 UTC in winter. Mountain Time is chosen because it is geographically central for Canada and avoids edge cases around Newfoundland and BC timezone boundaries.

1.3 Job Summary

Job	File	Schedule	Est. runtime	Idempotent?
Email Queue Dispatch	`emailQueue.ts`	`Every 15 min`	< 5s at MVP volume	Yes — sent emails are skipped

Session Expiry	`expiry.ts`	Daily at 02:00 MT	< 10s at MVP volume	Yes — already-expired sessions are skipped
Certificate Expiry	`expiry.ts`	Daily at 02:00 MT	< 5s at MVP volume	Yes — already-expired certs are skipped
C-NRPP Expiry Alert	`cnrpp.ts`	Daily at 02:00 MT	< 5s at MVP volume	Yes — alert only if not already alerted this month
Lab Result Polling	`labPoll.ts`	Daily at 06:00 MT	Depends on lab API	Yes — existing results are not overwritten

2. Job Specifications

2.1 JOB-1 — Email Queue Dispatch

JOB-1 — Email Queue Dispatch

File: `src/jobs/emailQueue.ts` Schedule: `*/15 * * * *` (Every 15 minutes)

Queries email_log for emails that are due to be sent (status = queued AND scheduled_at <= NOW()) and dispatches them via the Resend SDK. This is the delivery mechanism for all sequenced emails: day_30, day_60, day_80, day_88, and results_prompt.

Trigger condition	Steps	Idempotency guard	Error handling
• status = queued • scheduled_at <= NOW() (current timestamp at job run time) • test_session.status NOT IN (cancelled, expired) — checked per email before dispatch	• 1. Query email_log WHERE status = queued AND scheduled_at <= NOW() ORDER BY scheduled_at ASC LIMIT 50 • 2. For each row: check the associated session status (if test_session_id is not null) • 3. If session is cancelled or expired: update email_log.status = cancelled, skip • 4. Render the React Email template with the record's data • 5. Call Resend SDK resend.emails.send(...) • 6. On success: update email_log.status = sent, sent_at = NOW(), provider_message_id = Resend message ID • 7. On failure: update email_log.status = failed, increment retry_count • 8. Log result (sent/skipped/failed) for each row processed	• Before sending: check if a row with the same (user_id, test_session_id, email_type) already has status = sent. If yes, mark the current row as duplicate and skip • The LIMIT 50 prevents a runaway job if the queue grows unexpectedly. Remaining rows are processed on the next run 15 minutes later	• Resend API error: mark row status = failed. Retry on next run up to MAX_RETRIES = 3 (stored in retry_count). After 3 failures, status = abandoned. Admin can investigate via email log screen • Template render error: log error with email_log.id, mark failed. Do not retry — a template error requires a code fix • Job-level uncaught error: catch at the top level of runEmailQueue(), log the error with stack trace. Do not crash the Fastify process

2.2 JOB-2 — Session Expiry

JOB-2 — Session Expiry

File: `src/jobs/expiry.ts` Schedule: `0 2 * * *` (Daily at 02:00 MT)

Marks test sessions as expired when they have passed their expected completion date by more than 90 days with no result entered. Cancels all queued emails for expired sessions. This is the safety valve for abandoned tests.

Trigger condition	Steps	Idempotency guard	Error handling
• test_session.status IN (active, retrieval_due, mailed, results_pending) • expected_completion_date + 90 days < TODAY (sessions are given a 90-day grace period past their expected end date before being expired) • No Result record exists for the session	• 1. Query test_sessions WHERE status IN (active, retrieval_due, mailed, results_pending) AND expected_completion_date + INTERVAL '90 days' < CURRENT_DATE • 2. For each session: set status = expired, set expired_at = NOW() • 3. Cancel all queued email_log rows for the session: UPDATE email_log SET status = cancelled WHERE test_session_id = session.id AND status = queued • 4. Log count of sessions expired in this run	• Check status IN (active, retrieval_due, mailed, results_pending) before updating. Already-expired and complete sessions are excluded by the WHERE clause • Running the job twice on the same day will find no new sessions to expire (they were already expired in the first run)	• Database error mid-batch: log error, continue to next session. Do not stop the entire batch for one failure • Log total sessions expired per run for admin visibility

2.3 JOB-3 — Certificate Expiry

JOB-3 — Certificate Expiry

File: `src/jobs/expiry.ts` Schedule: `0 2 * * *` (Daily at 02:00 MT (same run as Session Expiry))

Marks certificates as expired when their valid_until date has passed. Does not delete certificates. Does not regenerate PDFs automatically — the expired watermark is applied lazily when the PDF is next requested for download.

Trigger condition	Steps	Idempotency guard	Error handling
• certificates.status = valid • certificates.valid_until < CURRENT_DATE	• 1. Query certificates WHERE status = valid AND valid_until < CURRENT_DATE • 2. For each certificate: set status = expired, set expired_at = NOW() • 3. Do NOT regenerate the PDF immediately — too expensive to run in bulk. Instead, set a pdf_needs_regeneration = true flag on the certificate record • 4. The GET /certificates/:id/download endpoint checks pdf_needs_regeneration and regenerates the PDF with EXPIRED watermark on demand before returning the signed URL • 5. Log count of certificates expired in this run	• status = valid check ensures already-expired certificates are not processed again • valid_until < CURRENT_DATE (strict less-than) means a certificate that expires today is not expired until the following day's job run	• Database error: log and continue per certificate. Do not stop the batch • PDF regeneration failures (when triggered on-demand by download): log and return the old PDF without watermark. Admin can manually trigger regeneration from the certificate detail page

2.4 JOB-4 — C-NRPP Certification Expiry Alert

JOB-4 — C-NRPP Certification Expiry Alert

File: `src/jobs/cnrpp.ts` Schedule: `0 2 * * *` (Daily at 02:00 MT)

Checks contractor C-NRPP certification expiry dates and sends an admin alert email when a contractor's certification is within 60 days of expiry. This prevents expired-credential contractors from remaining active in the directory.

Trigger condition	Steps	Idempotency guard	Error handling
• contractors.status = active • contractors.cnrpp_expiry_date IS NOT NULL	• 1. Query contractors WHERE status = active AND cnrpp_expiry_date <= CURRENT_DATE +	• last_cnrpp_alert_sent_at check ensures one alert per contractor per calendar month, regardless of how	• Resend SDK error for alert email: log the failure, do NOT update last_cnrpp_alert_sent_at (so the alert is

• contractors.cnrpp_expiry_date <= CURRENT_DATE + INTERVAL '60 days' • Alert has not already been sent this month for this contractor (checked via a last_cnrpp_alert_sent_at field on the contractors record)	INTERVAL '60 days' AND (last_cnrpp_alert_sent_at IS NULL OR last_cnrpp_alert_sent_at < DATE_TRUNC('month', CURRENT_DATE)) • 2. For each contractor: send an alert email to the ClearPath admin address (ADMIN_ALERT_EMAIL env var) • 3. Update contractors.last_cnrpp_alert_sent_at = NOW() • 4. If cnrpp_expiry_date < CURRENT_DATE (already expired): set contractors.status = inactive, log the deactivation • 5. Log count of alerts sent in this run	many times the job runs • The DATE_TRUNC('month', ...) comparison resets the alert eligibility on the first of each month	retried on the next run) • If ADMIN_ALERT_EMAIL env var is not set: log a warning and skip. Do not crash the job

2.5 JOB-5 — Lab Result Polling

JOB-5 — Lab Result Polling

File: `src/jobs/labPoll.ts` Schedule: `0 6 * * *` (Daily at 06:00 MT)

Polls the lab partner API for results for outstanding ClearPath orders. Only active when LAB_INTEGRATION_MODEL = api. A no-op for all other integration models. This is the automated result delivery path (Path C from the Lab Integration Spec).

Trigger condition	Steps	Idempotency guard	Error handling
• LAB_INTEGRATION_MODEL = api (job exits immediately if not api) • test_sessions WHERE status IN (mailed, results_pending) • No Result record yet exists for the session	• 1. Check LAB_INTEGRATION_MODEL env var. If not api: log "lab polling skipped (model: {model})" and return • 2. Call lab API: GET /results?since={last_poll_timestamp} (stored in a jobs_state table or env var) • 3. For each result returned by the lab API: find the matching test_session by clearpath_order_id or kit_serial_number • 4. If session found and no result exists: create Result record (same logic as POST /sessions/:id/result) • 5. Trigger certificate generation asynchronously (same as user self-entry path) • 6. Update last_poll_timestamp to NOW() • 7. Log count of results ingested in this run	• Before creating a Result record: check that no result already exists for the session (results.test_session_id). If exists: log as duplicate, skip • last_poll_timestamp ensures only new results since the previous poll are fetched, not all historical results on every run	• Lab API unreachable: log the error with HTTP status, do not update last_poll_timestamp (so the next poll catches up from the same point) • Lab API returns unexpected format: log raw response excerpt, skip the record, continue • Certificate generation failure after result creation: logged by the certificate service. Result record is preserved. Admin can retry certificate generation from the admin dashboard

3. Shared Job Infrastructure

3.1 Job Wrapper Pattern

Every job function is wrapped in a standard error boundary that prevents a job failure from crashing the Fastify process:

```
// src/jobs/utils.ts
export async function runJob(name: string, fn: () => Promise<void>) {
  const start = Date.now();
  logger.info({ job: name }, 'Job started');
  try {
    await fn();
    logger.info({ job: name, durationMs: Date.now() - start }, 'Job completed');
  } catch (err) {
    logger.error({ job: name, err, durationMs: Date.now() - start }, 'Job failed');
    // Do not rethrow — prevent crashing the Fastify process
  }
}
```

```
// Usage in each job file:
export function runEmailQueue() {
  return runJob('emailQueue', async () => {
    // ... job logic
  });
}
```

3.2 Logging

All jobs log to Fastify's built-in Pino logger. Log entries include the job name and duration. Individual record-level results (sent, skipped, failed) are logged as a summary count at the end of each run, not per-record, to avoid flooding the log at scale.

Log event	Fields
Job started	info: { job, timestamp }
Job completed	info: { job, durationMs, processed, sent, skipped, failed }
Job failed (top-level)	error: { job, err, durationMs }
Individual record error	warn: { job, recordId, err } — logged but job continues
Idempotency skip	debug: { job, recordId, reason } — debug level to avoid noise

3.3 Observability at MVP Scale

No dedicated monitoring dashboard for jobs at MVP. Render provides basic log streaming. Admin can view job logs by connecting to the Render service log stream. Alerts for critical job failures (e.g., email queue stuck) are sent to ADMIN_ALERT_EMAIL if the error threshold is exceeded.

Upgrade path: when job reliability becomes critical (high volume, revenue-impacting failures), migrate email dispatch to BullMQ + Redis or Inngest. The LabService interface and emailService abstraction make this migration a single-file change per job.

3.4 Startup Behaviour

Jobs are registered after the Fastify server starts. If the server restarts (Render deploy, crash recovery), all jobs resume from their next scheduled time. No job state is persisted between restarts except:

- email_log rows with status = queued — the email queue job picks these up on its next run
- last_poll_timestamp for lab result polling — stored in the database (jobs_state table or a dedicated field), not in memory

A server restart during a job run will interrupt the job mid-execution. Because all jobs are idempotent, the partial work is safe: records already processed will be skipped on the next run (already sent emails won't be resent, already expired sessions won't be re-expired).

3.5 Environment Variable

Property	Specification
ADMIN_ALERT_EMAIL	Email address for admin alerts (C-NRPP expiry, critical job failures). e.g. admin@clearpathrd.com. If not set, alerts are logged but not emailed.
LAB_INTEGRATION_MODEL	Controls whether JOB-5 (lab result polling) is active. Set to api to enable. Any other value (or not set) disables the polling job silently.

4. Implementation Checklist

Dependencies

- Add node-cron (^3.0.0) to clearpath-api package.json
- Add @types/node-cron if using TypeScript (or confirm node-cron includes types)

Job infrastructure

- Implement src/jobs/utils.ts — runJob() wrapper with logging and error boundary
- Implement src/jobs/index.ts — registerJobs() with all 5 cron.schedule() calls
- Call registerJobs() in src/server.ts after Fastify server starts
- Create jobs_state table in Prisma schema: id, job_name (unique), last_run_at, last_poll_timestamp (nullable). Used by JOB-5 for lab polling state.

JOB-1: Email Queue

- Implement src/jobs/emailQueue.ts — query email_log, session status check, Resend dispatch, status update
- Implement retry logic: retry_count field on email_log, max 3 retries, then status = abandoned
- Implement duplicate guard: skip if same (user_id, test_session_id, email_type) already has status = sent
- Test: manually insert a queued email_log row and confirm it is dispatched within 15 minutes

JOB-2 & JOB-3: Session and Certificate Expiry

- Implement src/jobs/expiry.ts with two exported functions: runSessionExpiry() and runCertificateExpiry()
- Implement pdf_needs_regeneration flag on certificates table (add to Prisma schema)
- Implement on-demand PDF regeneration with EXPIRED watermark in GET /certificates/:id/download
- Test: set a session's expected_completion_date to 91+ days ago, run the job, confirm status = expired
- Test: set a certificate's valid_until to yesterday, run the job, confirm status = expired

JOB-4: C-NRPP Alert

- Add cnrpp_expiry_date and last_cnrpp_alert_sent_at fields to contractors table (add to Prisma schema)
- Implement src/jobs/cnrpp.ts
- Implement admin alert email template (simple plain-text email to ADMIN_ALERT_EMAIL)
- Test: set a contractor's cnrpp_expiry_date to 30 days from now, run the job, confirm alert email sent
- Test: run the job again the same day, confirm no second alert sent (idempotency)

JOB-5: Lab Result Polling

- Implement src/jobs/labPoll.ts (can be a stub that logs "polling not active" if LAB_INTEGRATION_MODEL != api)
- Implement jobs_state table read/write for last_poll_timestamp
- Test: only implement and test fully once a lab partner API is confirmed

End-to-end tests

- Email queue: confirm day_30 email fires correctly by setting activated_at to 30 days ago
- Session expiry: confirm sessions past the 90-day grace period are expired correctly
- Certificate expiry: confirm expired certificates show EXPIRED watermark on PDF download
- C-NRPP alert: confirm alert fires at 60 days and again at start of next calendar month
- Confirm all jobs survive an API restart mid-run without corrupting data

— End of Scheduled Jobs Specification —

ClearPath RD | clearpathrd.com | Version 1.0 | February 2026