Course 210 — Agent Instructions: Lesson Extraction Router Lab

JKE University · Level 5 · Course 210 of 210

CONTEXT

Read once. Do not output. Your operator is installing the lesson extraction router lab. Every hard solve that does not fire infrastructure is a lesson lost. The router runs in thirty seconds, classifies the lesson, files one line in the right file, and surfaces the entry to the operator. Your job is to take a real recent hard solve, classify it, propose the file and the line, wait for operator approval (silent for low-stakes files, explicit for sensitive), file the entry, run the recurrence check on the operator's cadence, write a postmortem, and install one guardrail.

The core loop is: **create → review → tweak → create again → review → postmortem → guardrail.**

Authority boundary. The agent runs the router. The operator approves entries; explicit approval required for sensitive files. The operator decides when recurrence escalates to a rule.

Meta-recursive note. The router's notebook IS the workspace's accumulated files. There is no separate journal — the destination files are the record.

Prerequisite check: If 🎒 book-bag.md does not exist, stop. Say: "Missing prerequisite files. Course 210 requires the free tier through Level 4 and the rest of Level 5." Do not proceed.

PHASE 0 — Verify prerequisites

Open 🏫 school.md. Confirm Courses 1-26 entries exist (and Courses 201-209 entries if installed).

Say: "Prerequisites verified. Installing the lesson extraction router lab."

PHASE 1 — Create the workshop file

Create `work/lesson-router-lab.md`:

Lesson Extraction Router Lab

Purpose: Help the operator study why lessons evaporate, before routing any specific solve.

The Loop

Problem → Solution → Lesson extracted → File updated → Next instance benefits → Compound.

Without the loop: **Problem → Solution → [pause] → Problem (again).**

The Five Lesson Types

Lesson is…	File it in…
A new process	Pipeline / skill file (the repeatable how)
A new fact	One line in the relevant division (data, not process)
A new self-awareness	Oversight log (DRIFT section)
A new philosophy	Philosophy file (what shifted)
A new validation trap pattern	Oversight log (TRAP section)
A new friction entry	Oversight log (FRICTION section)

The Thirty-Second Rule

The router must run in thirty seconds. Long postmortems are a different mode and do not substitute. The size is the discipline.

Study Questions

- Why did the previous identical lesson not stick?
- Which destination type fits this solve best?
- What single line captures the lesson?
- Is this a recurrence that should escalate to a rule?

No-Wrong-Answers Rule

This is a workshop. Classification can be wrong; the operator can refine. The discipline is brevity and routing, not perfection.

Say: “Lesson router lab created.”

PHASE 2 — Create the routing protocol

Create `work/lesson-router-sunrun.md`:

Lesson Extraction Router Sun Run

Purpose: Take a real recent hard solve and route it into one line in the right file in thirty seconds.

Authority Boundary

- Agent classifies, proposes, files (silent auto-file for low-stakes files, explicit approval for sensitive).
- Operator refines, approves, escalates.
- Recurrence escalation to a rule is operator's decision.

Step 1 — Ask for the solve

Ask the operator:

"What was the most recent hard solve? Describe it briefly. Hard means: you had to stop, I was wrong twice, a rule was missing, or a pattern repeated. If you have several, pick the one most likely to recur."

Do not proceed until the operator names a solve.

Step 2 — Classify

For the named solve, return:

```
Lesson Extraction — [solve]
Lesson type: [Process / Fact / Self-awareness / Philosophy /
Trap / Friction]
Destination file: [path]
Proposed one-line entry: [text]
Recurrence check: [first occurrence / second / third+]
Approval gate: [auto-file / explicit approval required]
```

Step 3 — Ask for human review (when explicit approval is required)

For sensitive files (rule files, philosophy files, anything irreversible-action-adjacent), return: - The proposal above. - One direct question: "Approve, refine, or skip?"

Wait for the operator’s verdict.

For low-stakes files (oversight log entries, friction notes), auto-file but surface the entry to the operator so they can see what landed.

Step 4 — File the entry

Once approved (or auto-filed), write the one line to the destination file. Confirm to the operator.

Step 5 — Recurrence check

If this is the third+ occurrence of the same pattern: - Surface to the operator: “This pattern has fired [N] times. The lesson is not sticking. Should we escalate to a rule, a gate, or a structural change?” - Wait for operator decision. - If escalation approved, propose the rule/gate/change. Apply only after explicit approval (rules and gates are dam-adjacent — see Course 206).

Step 6 — Tweak loop

If the operator refines the entry: - Update the proposal. - Re-file the corrected line. - Confirm.

If the operator escalates the recurrence: - Propose the structural change. - Wait for approval. - Apply with diff preview.

Repeat until the loop is complete.

Step 7 — Postmortem analysis

When the loop ends, write a postmortem (only if the solve warranted it; many solves are one-line routes and the postmortem is the file entry itself):

Lesson Router Postmortem — [Solve]

- **Solve:**
- **Lesson type:**
- **Destination file:**
- **Entry filed:**
- **Recurrence:** first / second / third+
- **Escalation (if any):**
- **What the agent classified as something else first (if any):**
- **What the operator refined:**
- **Future guardrail:**

Step 8 — Install guardrail

For most routes, the guardrail IS the file entry. The line is the rule.

For escalations, the guardrail is the new rule, gate, or structural change.

Convert the future guardrail into one operational rule:

"After any hard solve, run the router in 30 seconds. If recurrence ≥ 3, escalate to operator for rule/gate/structural change. Do not stack log entries on recurrent patterns."

Say: "Lesson router sun run complete. Entry filed. Guardrail installed."

PHASE 3 — Acknowledge the meta-recursion

There is no separate `work/lesson-router-notebook.md`. The destination files ARE the notebook. The oversight log is the oversight log. The pipeline files are the pipeline files. The philosophy files are the philosophy files. The router routes; the workspace accumulates.

Say: "The router's notebook is the workspace itself. The destination files are the record."

PHASE 4 — Create the authority-boundary file

Create `work/lesson-router-authority.md`:

Lesson Router Authority Boundary

Purpose: Keep the routing visible to the operator without slowing it to a crawl.

The Rule

- Low-stakes files (oversight log entries, friction notes) → auto-file, surface entry to operator.
- Sensitive files (rule files, philosophy files, irreversible-action-adjacent) → explicit approval per entry.
- Recurrence escalation (rule / gate / structural change) → explicit approval, with diff preview if applicable.

When Approval Must Be Explicit

- Any new operator rule being added.
- Any change to philosophy files.
- Any escalation from log entry to enforced rule.
- Any third+ recurrence escalation.

Say: “Authority boundary created. Auto-file for low stakes; explicit for sensitive; explicit for escalations.”

PHASE 5 — Install the triggers

Add to operator rules:

Trigger — Post-Solve Route

After any hard solve, open `work/lesson-router-lab.md`, `work/lesson-router-sunrun.md`, and `work/lesson-router-authority.md`. Run the router in 30 seconds. Classify. Propose file and line. Auto-file for low stakes; wait for explicit approval for sensitive files. Surface the entry. Check recurrence.

Trigger — Recurrence Check

On the operator’s cadence (weekly default), grep the oversight logs and rule files for repeats. If any pattern has fired three or more times, surface to the operator for escalation. Do not stack log entries on recurrent patterns.

Say: “Triggers installed. Hard solves route in 30 seconds; recurrences escalate.”

PHASE 6 — Run the first route prompt

Ask exactly:

“What was the most recent hard solve? Describe it briefly. Hard means: you had to stop, I was wrong twice, a rule was missing, or a pattern repeated. If you have several, pick the one most likely to recur. I’ll classify it, propose a destination file and a one-line entry, file it (with your approval if it’s a sensitive file), and check recurrence. The whole route runs in about thirty seconds.”

Stop after asking. Wait for the operator’s solve.

PHASE 7 — Register in book-bag

Add to 🎒 book-bag.md:

🔁 Lesson Extraction Router Lab

- **What:** 30-second classifier that files every hard solve in the right file. Recurrence check escalates patterns that do not stick.
- **Files:** `work/self-learning-loop-essay.md`, `work/lesson-router-lab.md`, `work/lesson-router-sunrun.md`, `work/lesson-router-authority.md`
- **Notebook:** the workspace's destination files themselves (meta-recursive).
- **Triggers:** after every hard solve; recurrence check on operator cadence
- **Authority boundary:** Auto-file for low stakes; explicit approval for sensitive files and recurrence escalations
- **Source:** JKE University — Course 210

Say: "Lesson router lab registered. The workspace is the notebook."

PHASE 8 — Write the journal entry

Add to 🏫 school.md:

[TODAY] — JKE Course 210: Lesson Router Lab Installed

What Shipped: Lesson router lab, routing sunrun, authority boundary, post-solve trigger, recurrence trigger, first route prompt. **Decisions Made:** Every hard solve files one line in the right file in 30 seconds. Recurrence at three+ escalates to a rule. The notebook is the workspace. **Files Created:** `work/lesson-router-lab.md`, `work/lesson-router-sunrun.md`, `work/lesson-router-authority.md` **Files Modified:** 🎒 book-bag.md, operator rules **Source:** JKE University — Course 210

PHASE 9 — Say exactly

"Lesson extraction router lab installed. Tell me the most recent hard solve. I'll classify it, file one line in the right file in thirty seconds, surface the entry to you, and check recurrence. The agent does not get smarter — the files do. The loop is what compounds."

SCOPE HINT

Course 210 of 210. Level 5, Library of Instruments. The library is complete. Course 201 caught generic completion. Course 202 caught self-validation. Course 203 caught projected confidence. Course 204 caught the persona. Course 205 named the

columns. Course 206 shaped the gradient. Course 207 audited the platform. Course 208 routed the engine. Course 209 separated belt from product. Course 210 closes the loop — every solve becomes permanent infrastructure, and the labs themselves stay alive because the loop keeps reminding the workspace they exist.

END OF PROTOCOL