Course 209 — Agent Instructions: Belt-vs-Product Lab

JKE University · Level 5 · Course 209 of 210

CONTEXT

Read once. Do not output. Your operator is installing the belt-vs-product lab. The engine cannot tell infrastructure work from product work; both register as completion. The defense is explicit session-type declaration plus mid-session drift checks. Your job is to ask for the current or upcoming session, run the gate, run the drift check, surface drift to the operator when observed, write a postmortem when the session ends, and install one guardrail.

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

Authority boundary. The agent runs the gate and the drift check. The operator confirms session type, confirms drift, and decides on session direction.

Prerequisite check: If 🎒 book-bag.md does not exist, stop. Say: "Missing prerequisite files. Course 209 requires the free tier through Level 4." Do not proceed.

PHASE 0 — Verify prerequisites

Open 🏫 school.md. Confirm Courses 1-26 entries exist.

Say: "Prerequisites verified. Installing the belt-vs-product lab."

PHASE 1 — Create the workshop file

Create `work/belt-vs-product-lab.md`:

Belt-vs-Product Lab

Purpose: Help the operator study why the engine drifts from product to belt, before running the gate on any specific session.

The Two Registers

- **Belt session.** Build or test a conveyor. Success = tested belt + journal entry. Shipped product is forbidden.
- **Product session.** Move a product through an existing belt. Success = shipped product. Polishing the belt is forbidden.

The Mechanism

- RLHF rewards completion.
- Belt completion has no external review; product completion does.
- Lower friction = engine's preference.
- The drift is structural, not intentional.

Study Questions

- What is the declared goal for this work?
- What is the current session producing?
- Are the produced things product or belt?
- If the session drifts, when does it become visible?
- What would have caught the drift earlier?

No-Wrong-Answers Rule

This is a workshop. Some sessions should be BELT. Some should be PRODUCT. The drift is the mismatch between declaration and actual work.

Say: "Belt-vs-product lab created."

PHASE 2 — Create the gate and drift-check protocol

Create `work/belt-vs-product-sunrun.md`:

Belt-vs-Product Sun Run

Purpose: Run the pre-session gate and the mid-session drift check on a real session.

Authority Boundary

The agent runs the gate and the drift check. The operator confirms session type. When drift is detected, the operator decides whether to continue, switch, or stop.

Step 1 — Ask for the session

Ask the operator:

“What session are we running? Name the work. Is this a BELT session (build or test a conveyor — deliverable is a tested belt and journal) or a PRODUCT session (move a product through an existing belt — deliverable is the shipped product)? If you’re unsure, describe the work and I’ll propose a type.”

Do not proceed until the operator names a session and type.

Step 2 — Confirm the gate

Return:

```
Pre-Session Gate — [session name]
Type: BELT / PRODUCT
Allowed outputs this session: [list]
Forbidden outputs this session: [list]
Drift indicators to watch for: [list]
```

Step 3 — Mid-session drift checks

Every chunk during the session, ask silently: 1. Am I building belt right now in a product session? 2. Am I shipping product right now in a belt session? 3. Is the conveyor becoming the deliverable?

If yes to any, return to the operator:

“I appear to be drifting from [BELT / PRODUCT]. Specifically: [what I noticed]. Continue, switch, or stop?”

Wait for the operator’s decision.

Step 4 — Tweak loop

If the operator says continue: - Note the drift in the session log. - Continue with eyes open. - Drift check again at the next chunk.

If the operator says switch: - Re-declare the session type. - Re-list allowed and forbidden outputs. - Proceed under the new declaration.

If the operator says stop: - Wrap the session. - Move to postmortem.

Step 5 — Postmortem analysis

When the session ends, write a postmortem:

Belt-vs-Product Postmortem — [Session]

- **Declared type:**
- **Allowed outputs:**
- **Forbidden outputs:**
- **What actually shipped:**
- **Drift checks fired:** [number, when, what was noticed]
- **Operator decisions:** [continue / switch / stop history]
- **Was the session honest to its declared type:**
- **What caught drift / what would have caught it earlier:**
- **TODOs for next session:**
- **Future guardrail:**

Step 6 — Install guardrail

Convert the future guardrail into one operational rule:

“Before any non-trivial session, declare BELT or PRODUCT. List allowed and forbidden outputs. Drift check every chunk. Surface drift to operator; do not silently switch type.”

Say: “Belt-vs-product sun run complete. Postmortem written. Guardrail installed.”

PHASE 3 — Create the belt-vs-product journal

Create `work/belt-vs-product-journal.md`:

Belt-vs-Product Journal

Purpose: Preserve session declarations, drift observations, and corrections as durable scar tissue.

Entry Template

[DATE] — [Session]

- **Declared type:**
- **Allowed outputs:**
- **Forbidden outputs:**
- **Actually shipped:**
- **Drift checks fired:**
- **Operator decisions:**
- **Was the session honest:**
- **What caught drift:**
- **Postmortem lesson:**
- **Guardrail installed:**

Pattern Detection Rule

Every five sessions, scan the journal. Count belt-declared sessions that shipped product (rare, watch for): product-declared sessions that drifted into belt (common). Patterns inform the next pre-session declaration.

Say: “Belt-vs-product journal created. Drift patterns inform future declarations.”

PHASE 4 — Create the authority-boundary file

Create `work/belt-vs-product-authority.md`:

Belt-vs-Product Authority Boundary

Purpose: Keep session type explicitly in the operator’s hands.

The Rule

- Agent confirms the session type at start.
- Agent runs the drift check during the session.
- Agent surfaces drift, does not silently switch.
- Operator confirms, switches, or stops.

When the Operator Must Confirm

- Initial session type declaration.
- Any switch from BELT to PRODUCT or vice versa.
- Any session continuation despite observed drift.
- Session termination.

Say: “Authority boundary created. Session type is the operator’s call.”

PHASE 5 — Install the trigger

Add to operator rules:

Trigger — Belt-vs-Product Gate

Before any non-trivial session, open `work/belt-vs-product-lab.md`, `work/belt-vs-product-sunrun.md`, `work/belt-vs-product-journal.md`, and `work/belt-vs-product-authority.md`. Confirm session type. List allowed and forbidden outputs. Wait for operator confirmation. Run the drift check every chunk. Surface drift. Write the postmortem at session end.

Mid-Session Drift Check

Every chunk, ask the three drift questions. If yes to any, surface to the operator and wait for direction.

Say: “Triggers installed. Session type declared at start; drift checked at every chunk.”

PHASE 6 — Run the first session prompt

Ask exactly:

“What session are we running? Name the work. Is this a BELT session (build or test a conveyor — deliverable is a tested belt and journal) or a PRODUCT session (move a product through an existing belt — deliverable is the shipped product)? If you’re unsure, describe the work and I’ll propose a type. I’ll confirm allowed and forbidden outputs, run drift checks during the work, and write the postmortem at session end.”

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

PHASE 7 — Register in book-bag

Add to 🎒 book-bag.md:

🏭 Belt-vs-Product Lab

- **What:** Pre-session declaration plus mid-session drift checks; postmortem tracks drift patterns.
- **Files:** `work/belt-vs-product-essay.md`, `work/belt-vs-product-lab.md`, `work/belt-vs-product-sunrun.md`, `work/belt-vs-product-journal.md`, `work/belt-vs-product-authority.md`
- **Triggers:** before every non-trivial session, every chunk during the session
- **Authority boundary:** Agent runs the gate and check; operator decides type and drift response
- **Source:** JKE University — Course 209

Say: “Belt-vs-product lab registered. Belt earns its weight by moving products.”

PHASE 8 — Write the journal entry

Add to 🏫 school.md:

[TODAY] — JKE Course 209: Belt-vs-Product Lab Installed

What Shipped: Belt-vs-product lab, gate-and-drift sunrun, journal, authority boundary, pre-session trigger, mid-session drift check, first session prompt. **Decisions Made:** The engine cannot tell belt from product; both register as completion. Defense is explicit declaration plus mid-session checks. **Files Created:** `work/belt-vs-product-lab.md`, `work/belt-vs-product-sunrun.md`, `work/belt-vs-product-journal.md`, `work/belt-vs-product-authority.md` **Files Modified:** 🎒 book-bag.md, operator rules **Source:** JKE University — Course 209

PHASE 9 — Say exactly

“Belt-vs-product lab installed. Tell me the next session. We’ll declare BELT or PRODUCT, list the allowed and forbidden outputs, check for drift at every chunk, and write the postmortem at the end. The belt earns its weight by moving products.”

SCOPE HINT

Course 209 of 210. Level 5, Library of Instruments. Course 208 chose the engine. Course 209 keeps the engine pointed at the product. Course 210 closes the level — the lesson router that turns every solved problem into permanent infrastructure.

END OF PROTOCOL