

BorrowBox dispatcher: a reliable automation

Route a messy lending request into the right queue without asking an AI.

Build

Check

Try independently

The situation and your goal

Situation

A fictional lending desk receives normal requests, urgent requests and records without an item ID. Missing fields go to needs-details, urgent valid requests go to priority, and other valid requests go to normal.

Keep this artifact

A workflow export and a six-case input/output table.

Understand the mechanism

01

Rules can be enough

A known routing policy is easier to inspect as explicit conditions. An LLM would introduce variability without adding a needed capability to this task.

02

Validate before routing

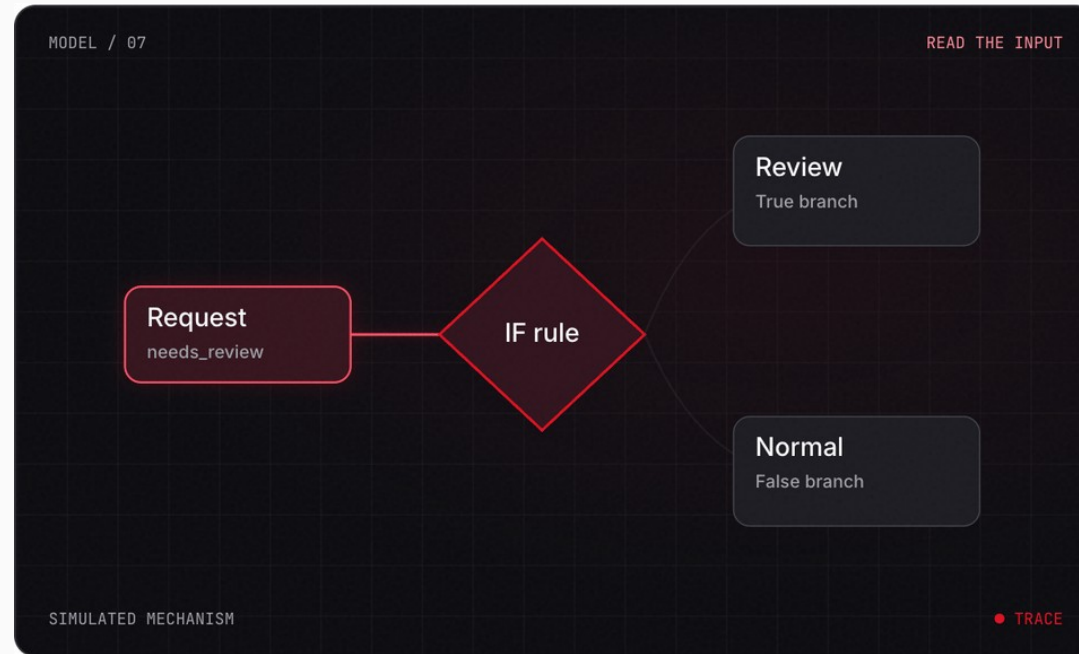
Missing item IDs and strings pretending to be booleans can produce incorrect branches. Check shape and type before making the routing decision.

03

Retries need identities

If a trigger repeats a request, a stable request ID lets you recognize duplication. Otherwise a reliable retry can accidentally create two jobs.

Explore before you build



Write the routing table

1

Open dispatcher-cases.json. For each record predict needs-details, priority or normal. Define what counts as a valid item ID and urgent boolean.

2

Check: Each input has one expected outcome.

3

Hint: Missing details take precedence over urgency.

Build the trigger and fields

- 1 In your n8n test workflow use Manual Trigger and Edit Fields to supply the sample record. Keep requestId, itemId and urgent as typed fields.
- 2 Check: The output matches your input schema.
- 3 Hint: Do not use a real production webhook for this first test.

Branch for missing details

- 1 Add an If node for missing or empty itemId. Route true to a named needs-details output.
- 2 Check: An urgent request with no item ID still asks for details.
- 3 Hint: This branch must happen before the urgency branch.

Branch for urgency

- 1 On valid requests add an If node that compares urgent as a boolean. Set the route field using Edit Fields on each branch.
- 2 Check: Valid urgent and normal requests reach their expected queues.
- 3 Hint: The string "false" is different from boolean false.

Run every fixture

- 1 Execute each of the six cases. Record actual output and execution ID beside the expected route.
- 2 Check: All outputs match the table or have a documented type-validation error.
- 3 Hint: Do not declare success from a single green execution.

Explain reliability

1

Export the workflow with no credentials. Describe how you would reject duplicate request IDs and handle an execution failure before enabling a real trigger.

2

Check: A partner can trace every branch.

3

Hint: Deduplication is an extension until an actual store is connected.

When it goes sideways

01

Missing item goes to priority

Urgency ran before validation. Try: Reorder the workflow so missing details stop the normal routing path.

02

The string false behaves strangely

Input values were not typed consistently. Try: Validate booleans instead of relying on truthiness.

03

Duplicate requests create duplicate work

No stable request identity is checked. Try: Add a request-ID store before acting on real requests.

Review your evidence

1

All three routing outcomes are reachable.

2

Missing data takes precedence.

3

Boolean and string inputs are distinguished.

Review your evidence

- 1 Six fixture outputs are recorded.
- 2 Export contains no credential secrets.
- 3 Duplicate-request handling is explained.

Why use fixed conditions here?

- 1 The routing policy is known and deterministic.
- 2 AI always gives a better result.
- 3 Conditions remove the need for tests.

Which branch runs first?

1

Urgency.

2

Missing-details validation.

3

Random choice.

Build with a clear contract

Design an n8n test workflow with Manual Trigger, Edit Fields and If nodes. Input fields: requestId string, itemId nonempty string, urgent boolean. Missing itemId routes to needs-details; valid urgent routes to priority; valid nonurgent routes to normal. Give a decision table including missing data, true, false, string "false" and repeated requestId. Explain where type validation and deduplication belong. Do not claim a dedup store exists until it is connected.

[Open copyable prompt](#)

Add a quiet-hours policy

- 1 Start/end time boundaries are tested.
- 2 Missing details still take precedence.
- 3 The decision table and workflow agree.

Build independently, then teach it back

Your next challenge

Introduce a time window: valid urgent requests during quiet hours should wait for review. Write boundary tests before editing the workflow.

Explain your decisions

Show your artifact, one test that passed, one failure you repaired and one boundary the system respects.