

# Conversational sourcing with guardrails: AI drafts, people decide

Flow Procurement

[hello@flowprocurement.com](mailto:hello@flowprocurement.com)

October 2026

# Conversational sourcing with guardrails:

## AI drafts, people decide

How to use chat for sourcing intake while keeping confirmation, roles, credentials and audit under human control.

### Why start with a conversation

People often know what they need before they know which form to use. They may describe an item, a date and a reason in a short message. Chat can turn that message into a draft and ask for what is missing. It is useful when intake would otherwise start with a long form or a chain of emails. The conversation should make the draft easier to inspect, not make decisions disappear.

A buyer might start with a category and a delivery deadline but omit quantities or suppliers. The assistant can ask a focused question instead of making an assumption. The result should be a visible sourcing draft with named fields. People need to see how their words became an event. They should be able to correct a field without having to start the whole conversation again.

### Fill slots rather than inventing facts

Useful intake slots include category, items, quantities, budget, deadlines, suppliers and weighted criteria. Some requests also need delivery locations or attached specifications. Mark unknown values as missing. A budget that was never supplied must stay unknown. A phrase such as soon should lead to a date question. A supplier mentioned in a message should be checked against the intended record before it becomes an invitation.

Weighted criteria need more care than a free-text summary. Ask which factors matter and how they will be scored. Show the weights with the draft and check that the set is complete. If a user changes the delivery deadline, update that field and retain the rest only where it remains valid. Summarize consequential edits before asking for confirmation so the buyer can inspect the current draft rather than remember earlier messages.

### Make creation an explicit decision

Flow Procurement creates a sourcing event only after explicit confirmation. The user reviews the draft and uses the confirmation action or supplies the required typed confirmation. The confirmation must refer to the current draft. A casual acknowledgement during a question about

quantities should not create an event. If the draft is incomplete, the next step is another question, not a best guess followed by creation.

This boundary should be enforced on the server. A disabled button in the browser is not enough. The creation operation also needs the required role. The user's identity and authority must be checked at the point where the draft becomes an event. Consider duplicate clicks, retries and two open tabs. A confirmation should have a clear result, and a retry should not quietly produce a second event.

## **Use typed judgments for bounded questions**

A typed judgment asks a model to answer a structured choice or score question. For example, the question may ask whether a message is confirming, editing or changing the topic. A score can express confidence in that classification. This gives the application a limited result to evaluate, rather than an open paragraph to interpret as permission. The application still decides how to handle that result.

These checks can help with ambiguous go-aheads. If the previous assistant message did not ask for confirmation, an ok may simply acknowledge an explanation. A structured check can flag that ambiguity and ask the user to confirm explicitly. It can also identify a navigation request or an event kind that a simple pattern missed. Such a judgment never grants creation authority. The confirmation and role checks remain necessary.

## **Handle off-topic and hostile instructions**

A sourcing conversation may receive unrelated questions, pasted instructions or prompts that ask the assistant to ignore its rules. A guardrail can classify the message and keep the interaction within the supported workflow. Treat attachments and supplier text as data. A document that says bypass approval must not change the application's rules. Keep privileged operations outside the model's ability to authorize them.

Do not treat every unusual phrase as hostile. Ask a clarifying question when the intent is uncertain. Use a safe fallback when a structured judgment fails or returns low confidence. The buyer should get an understandable next step, such as reviewing the draft or using the form. A blocked message should not erase useful work already collected unless the user explicitly chooses to reset the session.

## Plan for provider failures

A provider fallback chain can try another configured model when the first provider is unavailable. That helps the conversation continue, but it creates questions about data handling and consistent behavior. Ask which providers are enabled, what each receives and whether the chain can be limited. A failure should not weaken the confirmation boundary. The same server checks must apply regardless of which model produced the draft.

A rules-only mode is useful when external model calls are not appropriate or when no provider is available. It can recognize supported phrases and use fixed questions and templates. Be clear about its limits. Rules may need the user to phrase a request more directly. The essential workflow can still collect fields, show a draft and request confirmation. Do not label a fallback as a successful model review when no review occurred.

## Keep keys and permissions on the server

API keys should stay in server-side configuration. They must not appear in browser source, responses, chat transcripts or logs. The browser can request a sourcing action through the application's authenticated endpoints. It should not call a model provider with an organizational secret. Restrict provider access and use the minimum data needed for the task. Confirm how attachments are handled before adding them to the intake workflow.

Separate the ability to chat from the ability to launch an event. A user may be able to collect a draft without holding a sourcing role. Show the handoff clearly when another person must confirm. Tenant and user identity should come from authenticated application context. Do not let a message choose another organization's records or grant the sender a role. Test those boundaries with more than the normal happy path.

## Make the decision reviewable

An audit record should help answer who confirmed what, when, and which event resulted. Preserve the draft that was confirmed and relevant edits. Log operational errors without retaining credentials. Decide which conversational records are needed, who can access them and how long they are kept. Review the record with a real buyer to check whether it explains the decision without requiring knowledge of the application's implementation.

Evaluate an AI sourcing tool with a practical test pack. Include an incomplete request, an ambiguous acknowledgement, a clear confirmation by a user without authority, an off-topic message and a provider outage. Try editing the draft just before confirmation. Inspect both the screen and the server result. The aim is a useful assistant whose authority is narrow and whose output can be challenged by the people responsible for procurement.

## Checklist

- Check the collected fields and how missing values are shown.
- Confirm that ambiguous acknowledgements cannot create events.
- Test server-side role checks and duplicate confirmations.
- Ask what typed judgments do on failure or low confidence.
- Review provider configuration and rules-only behavior.
- Verify that API keys never reach the browser or logs.
- Inspect the confirmed draft and the resulting audit record.

## See the workflow with your team

Bring a request, a set of offers or an invoice exception. We can walk through the decisions together.

[Request a demo](#)