SuggestAPI

A GUIDE FOR MERCHANTS & COMMERCE TEAMS

The SuggestAPI Guide to Agentic Commerce

How to make product discovery trustworthy before an AI agent reaches checkout.

SUGGESTAPI · SUGGESTAPI.COM · LAST UPDATED OCTOBER 30, 2026

The core idea: SuggestAPI helps every search experience understand intent and return better results — sitting between a merchant's catalog and the storefronts, apps, copilots, and shopping agents that need to use it, without replacing the search and commerce systems already in place.

CONTENTS

What this guide covers

Product discovery is now part of the purchase path 3

Keep the responsibilities separate 4

Where SuggestAPI fits: intent before transaction 5

ACP and UCP matter — but they do not solve discovery 6

Build for the shopping job, not the keyword 6

A staged way to get ready 7

What SuggestAPI does — and what it does not do 9

Start where trust can be earned 9

THE STARTING POINT

Discovery now begins before the purchase

An agentic shopping experience begins before the purchase. A buyer describes what they need, an AI assistant finds and compares viable options, and — on supported surfaces — can start a merchant-controlled checkout after the buyer authorizes it.

The merchant still decides what is sold, whether an order is accepted, how it is fulfilled, and how the customer is supported.[2][3] That changes the job of product discovery.

> *"I need a carry-on backpack for a three-day work trip, under $200, with room for a 16-inch laptop."*

A shopper who says that has not supplied a tidy keyword. They have supplied a brief: budget, use case, size requirement, and an implicit expectation that the answer reflects what is actually available.

SuggestAPI is built for that part of the journey. It sits between a merchant's catalog and the places that need to use it — storefronts, apps, copilots, and shopping agents — without asking the merchant to replace the search and commerce systems already in place.[4][5]

Product discovery is now part of the purchase path

A conventional storefront can get by with product pages that look good to a person but are inconsistent beneath the surface. An agent cannot reliably guess whether a jacket is available in a particular size, whether it can arrive by Friday, or whether two variants are genuinely different. It needs current facts that can be retrieved and compared against the buyer's constraints.

Stripe's merchant guidance makes the same point from the transaction side. Agents read structured data and technical signals rather than browsing a site's navigation as a person would. Its guide recommends making catalog information accessible, current, and portable instead of relying only on web crawling.[1]

That does not mean turning every product record into a novel. It means including the information needed to answer the next sensible question. For most catalogs, start with five groups of information:

1. **Identity and hierarchy**
   Stable product and variant IDs, clear titles, category, brand, canonical URLs, images, and the relationship between a parent product and its purchasable variants.

2. **Sellability**
   Current price and currency, availability, inventory status, eligible markets, minimum quantities, and any restriction that affects whether the product can actually be bought.

3. **Fit and comparison**
   The attributes people use to choose: dimensions, material, compatibility, size, color, capacity, weight, ingredients, technical specifications, and category-specific trade-offs.

4. **Promise and policy**
   Delivery regions and timing, returns, warranty, subscriptions, personalization rules, and product-specific exclusions. These can decide a recommendation even when two products look similar.

5. **Provenance and freshness**
   The source system, last-updated time, and enough data lineage to establish which record is authoritative when a price, variant, or policy changes.

This is a practical starting point, not a protocol schema. The right fields follow the shopping jobs that matter to the business. A hiking-boot buyer may care about waterproofing, ankle support, and arrival date. A skincare buyer may care about ingredients, skin concerns, and bundle compatibility. A B2B buyer may care about seat limits, integrations, and contract terms.

2 · ARCHITECTURE

Keep the responsibilities separate

Agentic commerce is easier to trust when the agent-facing experience does not invent product truth or take over systems that already have a clear owner. Discovery, checkout, payment authorization, fraud controls, fulfillment, and returns have different failure modes. Calling all of them an “AI layer” makes the whole flow harder to audit and harder to improve.

SHOPPING SURFACES

Storefronts, apps, copilots, and shopping agents ask the questions.

↓

SUGGESTAPI — DISCOVERY LAYER

Turns a shopping brief into structured retrieval and returns catalog information applications and agents can use: search, indexing, ingestion, personalization, recommendations, integrations, and agent discovery.[7]

↓

CATALOG & SEARCH BACKEND

Authoritative for products, variants, price, stock, and policy.

CART & CHECKOUT

The merchant’s commercial path. Payment and risk systems decide whether an order proceeds.

AFTER THE SALE

Order management, fulfillment, returns, and support remain the systems of record.

> The boundary is deliberate. A discovery layer should help an agent find and explain the right options. It should not quietly make up a delivery promise, override a price, or claim to approve a payment.

3 · WHERE SUGGESTAPI FITS

Intent before transaction

Search boxes were designed around a person typing a short phrase. Agents need a clearer contract: what can be searched, which constraints can be applied, which product fields are current, and how options may be compared. SuggestAPI can give consuming surfaces a stable discovery interface while the underlying catalog or search provider stays in place.[4]

The pattern is simple:

1. Find

Turn a natural-language shopping job into retrieval that respects the shopper's non-negotiables — budget, size, compatibility, delivery region, or availability.

2. Compare

Return the attributes that make a real decision possible: a grounded explanation of how two or three viable options differ, not a generic list.

3. Recommend

Keep the shopper's stated preferences distinct from catalog facts, and link each claim back to live product information.

4. Handoff

Pass the selected product, variant, and relevant context into the merchant's cart or checkout route — without making the shopper start again.

The agent, chat interface, or protocol may change. The discovery contract should not have to be rebuilt whenever a new surface appears. That is the point of keeping one layer between the experiences that ask questions and the systems that hold the catalog truth.

> **One layer, many surfaces.** The experiences that ask questions change constantly. The systems that hold catalog truth should not have to change with them.

4 · PROTOCOLS

ACP and UCP matter — but they do not solve discovery on their own

The **Agentic Commerce Protocol (ACP)** is an open standard developed by Stripe and OpenAI for programmatic commerce flows among buyers, agents, businesses, and payment providers. In the documented flow, a buyer authorizes the purchase, the agent presents products and submits a checkout request, and the merchant can accept or decline the transaction as the merchant of record.[2][3]

The **Universal Commerce Protocol (UCP)** is another open standard. Google describes it as a way to support direct buying across its AI surfaces while letting merchants remain the merchant of record and choose the capabilities and communication methods they support.[6]

These protocols can reduce the custom work required to connect agents, commerce platforms, and payment flows. They do not decide what a shopper means by "smart casual," repair missing product fields, or explain which product better matches a narrow use case. They do not make an outdated inventory feed trustworthy.

> Transaction protocols can carry a purchase. They cannot supply the product understanding that makes the purchase worth initiating.

5 · THE SHOPPING JOB

Build for the shopping job, not the keyword

A better starting point than a protocol inventory is a narrow set of customer jobs. Choose requests that are both common and commercially meaningful, then ask whether the catalog can answer them without guesswork.

- **Luggage:** "Find a cabin bag that fits an airline's limits, holds a 15-inch laptop, and arrives before Thursday."
- **Home goods:** "Show me a washable rug for a high-traffic entryway."
- **Software:** "Which plan supports SSO for a 50-person team?"

For each job, test four things: can the system retrieve candidates that meet the hard constraints? Can it explain trade-offs with actual product facts? Can it keep price, stock, policy, and delivery information current while the buyer refines the request? And can it hand the buyer to the correct product, variant, cart, or checkout path?

When the answer is no, the missing work is usually not a new agent integration. It is a gap in product information, retrieval rules, or handoff design.

A staged way to get ready

Start with a small, testable commerce path. Trying to make every product and every channel agent-ready at once usually hides the actual gaps.

1

Pick one job and one surface

Choose a high-intent request with real constraints. Pair it with one discovery surface, such as an on-site assistant, a product-advisory workflow, or a supported external shopping channel. Define what a good answer must include and when it must defer to a person or the merchant checkout.

2

Map the authoritative data

Identify which systems own price, inventory, variants, shipping promises, policies, and product specifications. If several systems can write the same field, decide which value wins and how quickly it must be synchronized. A well-written answer is still a bad experience if it recommends a variant that went out of stock five minutes earlier.

3

Define the discovery contract

Make it possible to search, filter, compare, and retrieve product detail through predictable structured tools or endpoints. SuggestAPI is designed to normalize the interface between catalog and search infrastructure and the people or agents using it.[4][5] This is also where safe defaults belong: an agent should know what it can state, what it needs to retrieve live, and what it must not infer. If a delivery date depends on location and cutoff time, present it as a lookup — not a universal promise.

Design the handoff, manage the risk, measure the answers

4

Design the handoff before adding an agent channel

Decide where discovery ends. In some journeys, an agent may link to a product page or prepare a cart. In others, an approved protocol may support native checkout. The right path depends on the channel, payment setup, product type, and risk policy. The buyer should see the same selected variant, price basis, and relevant constraints in the handoff that informed the recommendation. ACP and UCP both support merchant-controlled transaction patterns, but the merchant still owns the final commercial decision.[2][6]

5

Treat access and risk as product requirements

Legitimate agent traffic can look different from human traffic, while malicious automation can exploit an agent's speed and reach. Stripe advises merchants to plan for new fraud patterns and to use controls that can be adjusted for individual agents or payment tokens.[1] Bring security, payments, legal, customer support, merchandising, and data teams in early. Decide who can change product facts, who can modify agent access, which orders need review, and who handles a bad recommendation or a failed handoff.

6

Measure answer quality before chasing conversion

Track whether the discovery layer is doing its job: constraint-match rate, required-field coverage, catalog freshness, price or availability discrepancies, comparison completeness, handoff completion, customer corrections, and the reasons orders are declined or abandoned. Conversion matters, but it is a lagging signal. A system that sends more shoppers into a broken handoff may briefly look busy while eroding trust. First make recommendations accurate, explainable, and current.

7 · SCOPE

What SuggestAPI does — and what it does not do

SuggestAPI does

- Help merchants make catalog discovery portable across search bars, apps, assistants, and agent-facing tools.
- Sit in front of an existing search stack and normalize requests and responses.
- Provide structured access to live product information without exposing private backend credentials to the consuming experience.[4][5]

SuggestAPI does not replace

- A commerce platform or product information system.
- Cart, checkout, or a payment provider.
- Fraud rules, an order management system, fulfillment, or customer support. Those systems retain their responsibilities.

That boundary makes agentic commerce more practical to adopt. Improve the part of the journey where agents need reliable product understanding while preserving the systems that already execute the transaction and serve the customer after the sale.

8 · NEXT STEP

Start where trust can be earned

The next move is not to collect every new protocol name. Map one real shopping job from question to handoff and see where the catalog stops supporting a confident answer.

Connect the data already trusted by the business. Expose it through a stable discovery layer. Test the questions a real shopper asks after the first result. Then add transaction pathways that match the merchant's risk policy, operations, and customer experience.

> Agentic commerce becomes useful not when an agent can say "buy now," but when it helps a customer reach the right product with facts the merchant can stand behind.

Learn more: What is SuggestAPI — suggestapi.com/what-is-suggestapi · How SuggestAPI works — suggestapi.com/how-it-works · API reference — suggestapi.com/docs/reference

REFERENCES

Sources

[1] Stripe — *How to get ready for agentic commerce in an evolving landscape*. go.stripe.global/rs/072-MDK-283/images/A_Guide_to_agentic_commerce.pdf

[2] Stripe — *Developing an open standard for agentic commerce*. stripe.com/blog/developing-an-open-standard-for-agentic-commerce

[3] OpenAI — *Buy it in ChatGPT: Instant Checkout and the Agentic Commerce Protocol*. openai.com/index/buy-it-in-chatgpt

[4] SuggestAPI — *What is SuggestAPI*. suggestapi.com/what-is-suggestapi

[5] SuggestAPI — *How SuggestAPI works*. suggestapi.com/how-it-works

[6] Google — *Getting started with Universal Commerce Protocol on Google*. developers.google.com/merchant/ucp

[7] SuggestAPI — *API reference*. suggestapi.com/docs/reference

> **SuggestAPI** — the discovery layer in front of your existing search engine, turning natural-language shopping queries into structured intent, filters, and context. Learn more at suggestapi.com.