trebllе

Preparing Your APIs for AI Agents

Five Keys to Making Your APIs Discoverable, Context-Aware, and AI-Ready

Written by:

Mike Amudsem,
API Advisor & Author

treblle.com

Executive Summary

Artificial Intelligence (AI) continues to infiltrate almost every aspect of computing and that includes APIs. As we enter a new phase of the digital evolution, APIs are no longer accessed only by humans. Increasingly, they must also support interactions with machine consumers such as AI agents, autonomous systems, bots, and intelligent orchestration engines. These entities don't rely on visual cues, long printed documents, or trial-and-error learning the way humans do. That means API designers need to identify clear intent, understand machine-readable contexts, recognize predictable API patterns, and have the ability to support finding and operating discoverable affordances like forms and links in API responses.

If your APIs are not already designed to support these capabilities, you are likely making it hard, if not impossible, for machines to safely and successfully take advantage of your APIs. Quite simply, your APIs are not AI-Ready.

This white-paper outlines five critical shifts organizations can make to ensure their API ecosystems are machine-ready. Drawing inspiration from pioneers like Donald Norman, Douglas Engelbart, Roy Fielding, Ted Nelson, and Christopher Alexander, this report offers a strategic and actionable framework to help API teams prepare for the inevitable; machine-driven APIs.

Shift from Interfaces to Intentions

Machine consumers don't need UI—they need meaning.

> "We must design for the way users behave, not for how we would wish them to behave."

Donald Norman
Board member NN/g

Why this matters?

Unlike humans, machines are not intuitive explorers. They don't poke around your interface hoping to find the function they need. Instead, AI bots need explicit affordances; clues about what can be done, when, and why. The classic model of an API is operational and implementation- focused: `GET /users`, `POST /transaction`, etc. The challenge here is that HTTP commands only provide the how, not the why.

Machines benefit form a design that communicates what the system enables; what's possible and why it might be useful. This means going beyond simple HTTP commands and including meaningful labels like `<form name="submitReview" >…</form>`. Make it possible for machines to find meaning in your messages.

What you can do right now?

- **Rename endpoints and operations** to reflect what users or agents can accomplish, not how the system is implemented. A great example of this can be found in the OpenAPI specification's `operationId` property.

- Include **state-transition descriptions** in your documentation (e.g., "Move from draft to submitted using `submitApplication`"). Make it possible for LLMs to derive meaning from metadata in the message.

- Start using **affordance modeling languages** like ALPS, TypeSpec, and Smithy to describe what's available at each state of interaction.

Make Context Machine-Readable

Metadata is the new interface.

"The better we get at getting better, the faster we will get better."

Douglas Engelbart
Human-Computer Interaction Pioneer

Why this matters?

Context is the key to autonomous decision-making. Most of the recent leaps forward for AI engines were powered by the ability to expand the amount of information processed by bots in order to improve their ability to identify context during processing. Machines need to know why something exists, **who owns it, what constraints apply**, and **how it fits into the larger system.** Without this, even well-structured APIs become opaque. That means adding a great deal more metadata in API responses in order to better describe possible actions available to the machine.

What you can do right now?

- Add **metadata endpoints** that declare ownership, versioning policies, capabilities, and terms of use. There are some great examples of this in the APIs.json specification.

- Embed **capability statements** that explain the details of an API in machine-readable formats (e.g., HTTP method, mediatype, input fields, url or action properties, and so forth).

- Declare **domain affiliations** to help agents reason about relevance and relationships (e.g., this API belongs to the banking or accounting or shopping domain).

Standardize Interactions, NotJust Endpoints

Predictability is the foundation of autonomy.

> “A good architecture is not created in a vacuum."

Roy Fielding
Co-Author of HTTP & Creator of REST

Why this matters?

Machine agents depend on predictability. Every inconsistency in the way an API interacts with a machine is a new thing that a machine needs to learn before it is successful. Things like paging, submitting data, completing authentication, error handling, or or anything that requires the machine to deal with common patterns adds cognitive load and risks successful completion of the task. Consistent interaction patterns allow reuse of automation strategies and reduce the need for hard-coded workarounds.

What you can do right now?

- Use **uniform error schemas** with predictable status codes, causes, and remedies. Always report errors using the Problem Details for HTTP APIs format.

- Define **timeout and fallback patterns** clearly, so agents can respond to degradation intelligently. Make sure you make it easy for machines to retry operations. Stick to using HTTP PUT to write data instead of relying on non-idempotent HTTP POST.

- Apply **system-wide conventions** for pagination, retries, and state transitions, etc. Don’t surprise your agents with a novel way to handle a common pattern.

Enable Discovery Through Ecosystem Signals

If they can't find it, they can't use it.

"Everything is deeply intertwingled."

Ted Nelson
Inventor of Hypertext

Why this matters?

Machine consumers can't browse a portal or read a change log. And even if you give an LLM the complete API documentation, there is no assurance it will understand all the details of your prosaic API descriptions. Be sure to provide clear instructions and adopt common approaches to communicating API discovery and find-ability. This means publishing API indexes, posting your description documents to shared registries, providing cross-linked documentation, and employing consistent hypermedia signals (e.g. links to home, collection, item, update, etc.). The more your APIs surface who they are and what they do, the more usable they become for machines as well as humans.

What you can do right now?

- Register APIs in **searchable catalogs** like Postman, SwaggerHub, or your own shared internal directories.

- Use **consistent, semantic tags and domain descriptions** to support machine classification (see Step 2 above)

- Add **hypermedia links** and self-describing endpoints that allow navigation, filtering, and progressive disclosure of capability and do it in a consistent way across your APIs. See RESTful Web API Patterns and Practices Cookbook for lots of helpful examples.

Observe, Adapt, and Iterate

Machine usage is not static; it evolves.

“Each pattern is a solution to a problem in a context.”

Christopher Alexander
Architect & Author of 'A Pattern Language

Why this matters?

Despite what you may think, machines are not passive consumers. This is especially true for LLM- powered machines. In fact, many bots rely on what is called a "greedy algorithm" pattern. They optimize for the current, local context. And that context can easily change over time, even if your API does not. You should pay attention to how machine agents behave when calling your APIs; especially when these agents fail or give up before completing their task. Knowing where machines succeed, where they get stuck, and what assumptions they make will give you valuable information on how to evolve and update your APIs for more successful interactions.

What you can do right now?

- Set up telemetry to **monitor usage by agents**, not just human developers. The world of observability is your friend.

- Watch for friction points and patterns of repeated errors, fallback usage, or documentation gaps. These are cases where machines are giving you a gift; advice on how to improve your APIs.

- Treat each usage pattern as a **signal for adaptation**. Just like living systems, APIs must grow in response to how they’re used. Don’t stop after you first release your API. Constantly monitor and improve your APIs over time.

Final Thoughts: Beyond Interfaces, Toward Intelligence

To prepare your API ecosystem for machine consumers, you must think like a systems architect, not a UI designer. Machines need **semantic labels** to know what is possible. they need **data affordances** to know how to take action, and they need large scale **interaction patterns** to make it easier to interact with your API and get their jobs done. These are not technically challenging, but they represent a mind-shift in designing and implementing APIs. A mind-shift that embraces genAI machines and API clients worthy of your efforts.

By embracing intention-driven design, machine-readable context, standard interaction patterns, discoverable affordances, and iterative learning, you can set your organization on the road to taking advantage of the growing AI-powered machine-to-machine API world.

About Mike Amundsen

Internationally known author and speaker Mike Amundsen consults with organizations around the world on network architecture, Web development, and the intersection of technology & society. He helps companies capitalize on the opportunities provided by APIs, Microservices, and Digital Transformation.

Amundsen has authored numerous books and papers. His most recent book is "RESTful Web API Patterns and Practices Cookbook" (2022). His "Design and Build Great APIs" (2020) is a popular developer-centric book. Amundsen also contributed to the book, "Continuous API Management" (2020, 2018). His "RESTful Web Clients", was published in February 2017 and he co-authored "Microservice Architecture" (June 2016).