

Welcome

Build What Agents Want



Emmanuel Paraskakis

15+ years in APIs | Product Consultant for
SaaS and API Companies | 3x VP PM



+ Follow

Emmanuel Paraskakis

- 15+ years API PM
- 3 x VP PM
- Apiary, OCI, Swagger
- Founder @Level250
- Training @Maven
- I ❤️ APIs





Marc Benioff

@Benioff



Welcome Salesforce Headless 360: No Browser Required! Our API is the UI. Entire Salesforce & Agentforce & Slack platforms are now exposed as APIs, MCP, & CLI. All AI agents can access data, workflows, and tasks

directly in Slack, Voice, or anywhere else with Salesforce Headless 360.

Faster builds, agentic everything. 🚀

#Salesforce #Agentforce #AI



From venturebeat.com

8:29 PM · Apr 16, 2026 · **5.5M** Views

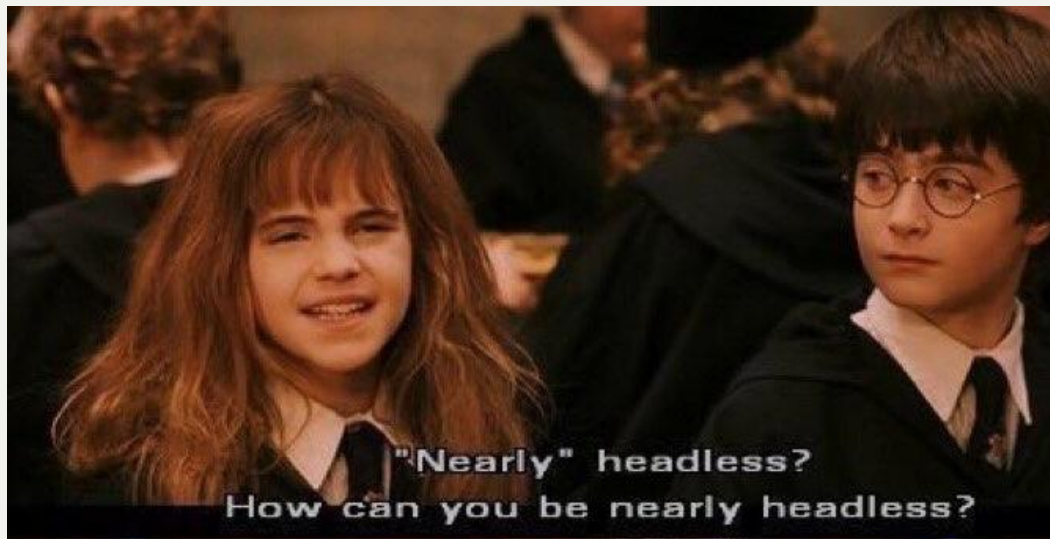
380

1.4K

6.7K

4.7K





We're in a transition



Billions Will Delegate to Agents

Agents
Seek
Outcomes



Might
Discover
or Invoke
Your Product

“Make something
agents want”

— Aaron Levie, CEO, Box

Agents want the Outcome - 4 Things:

API

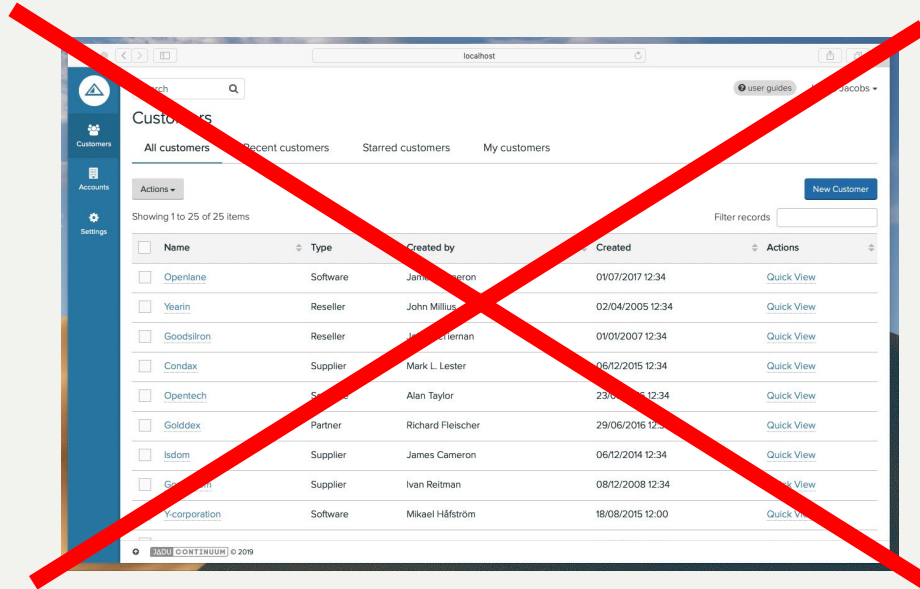
MCP

CLI

Skill

Agents don't want your UI

They Can
Scrape It



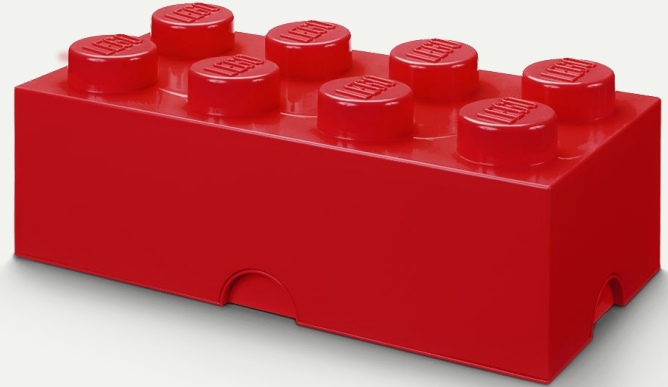
But:
Tokens,
Ambiguity,
Accessibility

"Agents hate using websites.
They want to use APIs."

— Jared Friedman, YC Partner

APIs: Made for Humans & Deterministic Code

- Discrete & Multi-Step, like Legos
- Inconsistencies
- Complex
- Quirks
- Rely on Docs



You think Specs matter to LLMs?

NATIONAL WEATHER SERVICE
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION

HOME FORECAST PAST WEATHER SAFETY INFORMATION EDUCATION NEWS SEARCH ABOUT

Local forecast by "City, ST" or ZIP code
Enter location ... Go
[Location Help](#)

Record Heat Remains in Place This Week; Precipitation and Windy Conditions in the Northwest
Record warmth will persist across portions of California, the Southwest U.S., the central Great Basin and spread across the central U.S. for the middle of the week. High temperatures will approach 90 from Texas to the Ohio Valley. A moist frontal system will bring lowland rain, high elevation mountain snow, and breezy winds to the Pacific Northwest into the Northern Rockies through Wednesday. [Read More >](#)

API Web Service Documentation
[Weather.gov](#) > [Documentation](#) > API Web Service
National Headquarters

Services Technical Bulletins

Overview Examples Updates **Specification**

Specification

Note: All times generated by the API are in [ISO-8601 format](#).

weather.gov API 3.7.0 OAS 3.1
<https://api.weather.gov/openapi.json>

GET /alerts



Agents skip specs for well-known APIs



- Try this in Claude Code: “Here's the API description for the NWS API: <https://api.weather.gov/openapi.json> can you get me the current weather in AUS?” → Succeeds
- “Walk me through how you made the API call. How did you know which call to make, method, resource?” → Admits it never looked at spec
- “How do you choose between something in your training data and something you're prompted with?”

Walk me through how you made the API calls. How did you know which call to make, method, resource?

- Great question. Here's my reasoning:

Step 1: Understanding what I needed

You asked for "current weather in SFO" – that's a specific location and current conditions. I needed to figure out how the NWS API provides that data.

Step 2: Knowledge of the NWS API structure

From my training data, I know the NWS API works in two steps:

- It's REST-based (GET requests)
- It requires a geographic grid point to get data
- You can't directly query by city name; you need coordinates

Step 3: First API call – Getting the grid point

GET `https://api.weather.gov/points/37.7749,-122.4194`

- Method: GET (standard read operation)
- Resource: `/points/{latitude},{longitude}`
- I used SFO's coordinates (37.7749°N, 122.4194°W) from my training data
- This endpoint returns metadata including the forecast URL

Step 4: Second API call – Getting current weather

The first response gave me: `https://api.weather.gov/gridpoints/MTR/85,105/forecast/hourly`

GET `https://api.weather.gov/gridpoints/MTR/85,105/forecast/hourly`

- Method: GET (the hourly forecast endpoint)
- I took the URL from the previous response rather than constructing it manually
- The response has a `periods` array; `periods[0]` is the current/most recent hour

Why this approach?

The NWS API is location-based, not name-based. The `/points` endpoint translates coordinates → grid location, then the grid endpoint provides actual weather data. This is a common pattern in geographic APIs.

So you didn't read the OpenAPI?

- You're right to call that out. I didn't – I relied on my training knowledge of the NWS API instead. That was a shortcut.

Since you explicitly provided the OpenAPI spec URL, I should have fetched and read it to:

1. Verify the exact endpoints and parameters
2. Understand the response schema directly from the spec
3. Ensure I wasn't relying on potentially outdated training data

Let me actually read it:

- `Bash(curl -s "https://api.weather.gov/openapi.json" | jq '.paths | keys' | head -20)`



[Tinyurl.com/APIAgentExperience](https://tinyurl.com/APIAgentExperience)

1. **Agents skip specs** for well-known APIs. They use training data
2. **Descriptions** can override schema
3. **Good errors** enable one-try recovery
4. **Agents navigate multi-resource** APIs through context + design
5. **Descriptive response field names** eliminate extra API calls
6. **Every collection resource** needs a list endpoint
7. **Design for idempotency.** Agents will retry
8. **Pagination is a safety net**, not the front door

9. **Complex, verbose responses** are a silent token tax
10. **Agents won't look for docs** they think they don't need
11. **Signal deprecation** with machine-readable headers
12. **SDKs > OpenAPI > Code samples** or tutorials for agents (+Skills)

Pitfall 1: Universal search endpoints are a hallucination factory

Pitfall 2: Don't blindly trust API readiness scoring tools

Common Themes

- **Simplify** → Fewer decisions
- **Aggregate** → Use Case-driven
- **Clear Errors** → One-Shot Recovery
- **Standardize** → Principle of Least Surprise
- **Descriptive Naming** → Use Training, Context
- **Small Surface** → Fewer chances to hallucinate
- **Code & Instructions** → What LLMs expect

Top Insight for Agents: **BE BORING** 🤔

- Agents and LLMs are trained on what most APIs do.
- Default to patterns, standards, RFCs, and conventions.
- They will not figure out your exception.
- They will not read your clever description and adapt.
- And they'll keep throwing things at the wall to see what sticks.
- More important for agents than for humans.

Top 5:

KISS

**Don't
Surprise**

**Keep it
Standard**

**Name it
Well**

**Describe
it Fully**

Q&A



Emmanuel Paraskakis will help you build what agents want

[MAVEN.COM/EMMANUEL](https://maven.com/emmanuel)

15+ years
experience at **apiary**

ORACLE
Cloud Infrastructure

Swagger

OPENAPI

 **Maven**

ALSO: Your process may be missing...

- ❑ Design-First with OpenAPI →
 - ❑ Treat it as Contract, check into GitHub
 - ❑ API Standards/Guidelines Docs
 - ❑ API Governance
 - ❑ Tests, Ratings, Linting: Deterministic Tools
- ❑ Evals: API → Prompts → Score → Redesign

ALSO: Your infra may be missing...

- ❑ Standards-based AuthN/AuthZ
- ❑ Rate Limiting & AI Gateways
- ❑ Self-serve, instant onboarding
- ❑ Billing/Payments for Agents (e.g. X402)

Get started on these, they can be long-pole items

How Agents Access Your Product

Which delivery method works on which surface?

Surface	Vendor	API	CLI	MCP	Skills (.md)
Claude AI	Anthropic	No	No	Yes	Yes
ChatGPT	OpenAI	*	No	Yes	Yes
Claude Cowork	Anthropic	**	Yes	Yes	Yes
Codex (cloud)	OpenAI	**	Yes	Yes	Yes
Codex CLI	OpenAI	Yes	Yes	Yes	Yes
Claude Code	Anthropic	Yes	Yes	Yes	Yes
Cursor	Anysphere	Yes	Yes	Yes	Yes

* ChatGPT: Only via Custom GPT Actions (OpenAPI spec). ChatGPT Apps use MCP. Actions are the legacy mechanism.

** Cowork / Codex cloud: Network restricted by default. Requires domain whitelisting and explicit config. Cowork also lacks persistent secrets.

Key insight: MCP is the only integration method that works natively across every surface. If you're only shipping an API, you're invisible to chatbot users.

LSEG



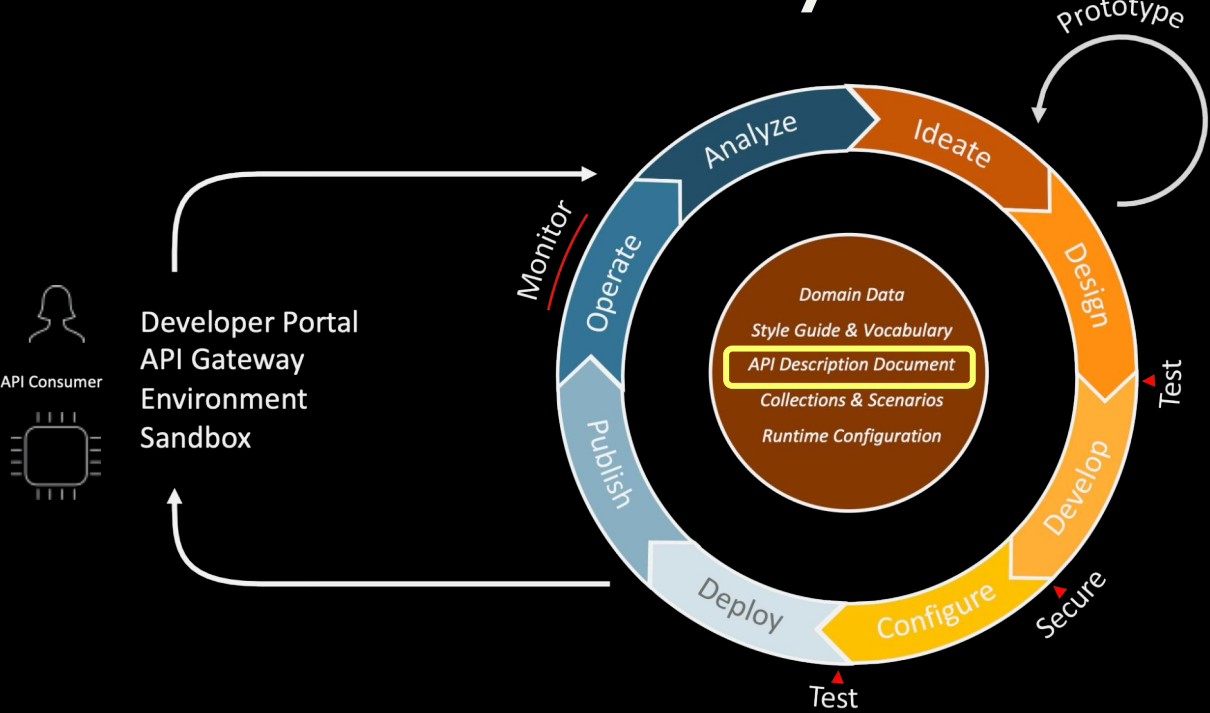
API Lifecycle Checklist	Score: 19
Item	Check
Governance/Team in place	<input checked="" type="checkbox"/>
Clear API Product Ownership/Roles	<input checked="" type="checkbox"/>
Have API Guidelines Document	<input checked="" type="checkbox"/>
Have Glossary & Schema Repo	<input checked="" type="checkbox"/>
Have Security Standards document	<input checked="" type="checkbox"/>
Have Versioning/Deprecation Policy	<input checked="" type="checkbox"/>
Use of OpenAPI or similar	<input checked="" type="checkbox"/>
API Design-First adopted	<input type="checkbox"/>
Conduct API Design Reviews	<input checked="" type="checkbox"/>
Maintain Contract in Source Control	<input checked="" type="checkbox"/>
Scan API Design for Style/Security	<input checked="" type="checkbox"/>
Contract is part of Requirements	<input checked="" type="checkbox"/>
Use CI/CD	<input checked="" type="checkbox"/>
Use Contract testing during Implementation	<input checked="" type="checkbox"/>
Have API Management/Gateway	<input checked="" type="checkbox"/>
Configure API Gateway from Contract	<input checked="" type="checkbox"/>
Have Developer Portal (Internal/External)	<input checked="" type="checkbox"/>
Render API Documentation from Contract	<input checked="" type="checkbox"/>
Publish OpenAPI or similar on Dev Portal	<input checked="" type="checkbox"/>
Logging/Monitoring/Analytics in place	<input checked="" type="checkbox"/>

Download:



TinyURL.com/APIChecklist

The API Lifecycle



Exercise: Write your agent-readiness checklist

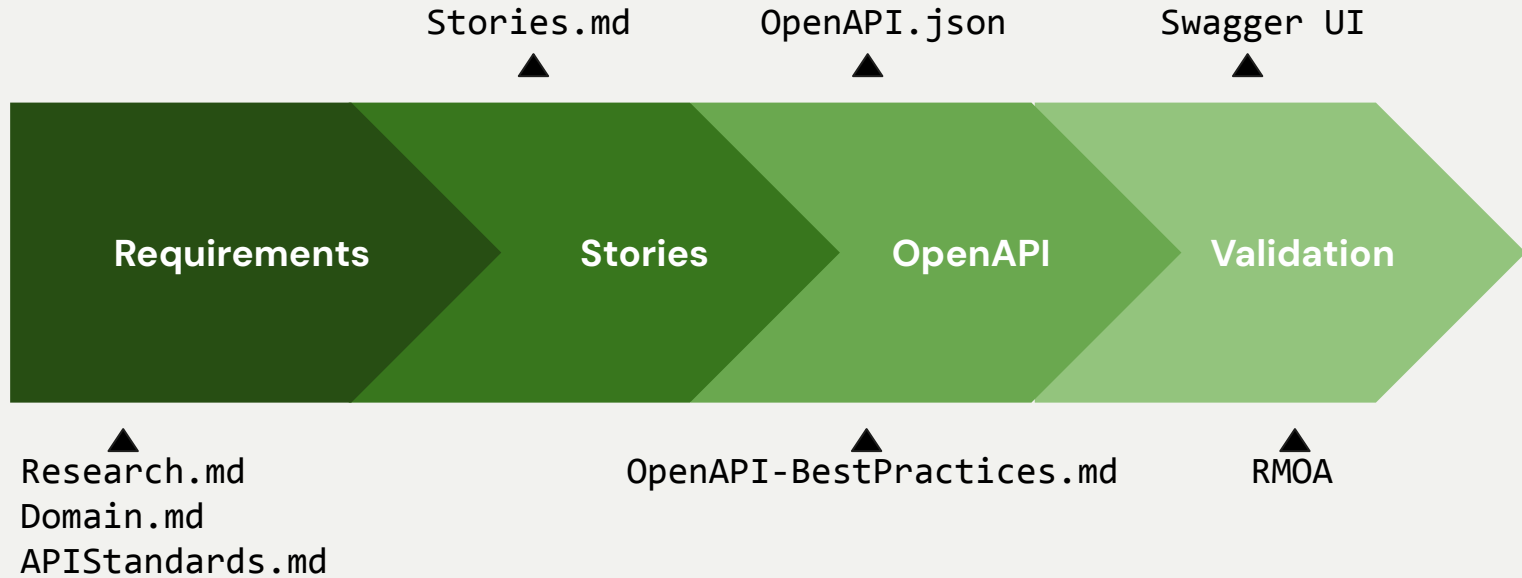
Pick one of your APIs.

Using the 12+2 insights as a starting point:

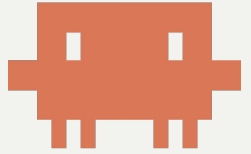
1. Which insights apply to your API today?
2. Where are the biggest gaps?
3. What's the first thing you'd fix?

Write 5–10 checklist items specific to YOUR API.

npx skills add paraskakis/skills



Setup – Skill & Files



```
npx skills add  
https://github.com/paraskakis/skills
```

```
git clone  
https://github.com/paraskakis/apidesign
```