SIGNALS Vol 1 Issue 1

Handy API Extension Patterns

Mike Amundsen

Handy API Extension Patterns

How to Evolve Message Formats Without Breaking Everything

Mike Amundsen

This book is available at https://leanpub.com/api-extension-patterns

This version was published on 2025-05-11

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

The author generated this text in part with GPT-3, OpenAI's large-scale language-generation model. Upon generating draft language, the author reviewed, edited, and revised the language to their own liking and takes ultimate responsibility for the content of this publication.

Contents

Introduction

One of the most persistent challenges in API design is managing change.

It's easy to fall into the trap of thinking an API contract is something you can lock down. Publish the schema, validate the input, ship the docs, and you're done. But anyone who's worked with real systems, at real scale, knows better. APIs evolve. They have to. New features emerge, business requirements shift, integrations deepen, and sometimes what seemed like a clear and stable shape for your data turns out to be incomplete. Change is not just likely; it is inevitable. In software, nothing lasts forever. And every change you make to an API comes with a cost.

Even something as simple as adding a new field to a JSON response can trigger unexpected failures. For example:

- A mobile app, written with a hand-rolled parser, crashes because it didn't expect the new field and treats the message as malformed.
- An automated data pipeline silently discards new properties, leading to inconsistent or incomplete results downstream.
- A form component in a web app binds directly to known fields in the response and starts throwing errors when one is renamed or temporarily omitted.
- A legacy client written in a compiled language has the response shape baked into a generated type. Any change means a full code regeneration, retesting, and redeployment.

These are not theoretical. They are the kinds of stories you hear again and again when you work in APIs long enough. When it comes to chaning API payloads. what started as a small internal tweak can quickly turn into a downstream fire drill.

And that's the challenge with API changes: often, the coupling is invisible until it breaks.

So the question is not *whether* you'll need to change your API. You will. The question is *how* to make those changes without breaking everything around you.

In this whitepaper, I want to focus on just one kind of change: the ability to evolve the *shape* of your response payloads over time. I'll show you a pattern I use regularly to do this safely, with minimal disruption. It's simple, flexible, and works across a wide range of client types—from dynamic JavaScript apps to static type-checked systems.

Let's take a closer look.

But first...

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/api-extension-patterns.

The type is strong with this one

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/api-extension-patterns.

I can haz extensions?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/api-extension-patterns.

Take a walk on the mild side

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/api-extension-patterns.

One ring to access them all

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/api-extension-patterns.

One more thing...

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/api-extension-patterns.

In conclusion ...

Over the course of this whitepaper, we've looked at:

- The **inevitability of change** in API design and why it matters more than we think
- The **spectrum of typing** and how mildly-typed messages give us structure *and* flexibility
- A practical extension format using **name/value pairs** to support safe, schema-compatible evolution
- A unifying **accessor pattern** to flatten the experience and reduce cognitive overhead
- And finally, the **long-term payoff** of adopting this design: safer updates, more confident deployments, and a smoother path forward for everyone involved

This is a small shift in how we model messages. But it leads to big dividends in stability, in resilience, and in the ability to say "yes" to change without fearing what might break.

So go ahead. Build this into your API designs. Start small. Add a `_nvp` array. Use the accessor pattern. And watch how much easier things get in the long run.

The future is always coming. This is how you get ready.

Recommended Reading & Foundations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/api-extension-patterns.

Type Systems & Theory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/api-extension-patterns.

API Design & Evolution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/api-extension-patterns.

JSON Schema & Validation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/api-extension-patterns.

JavaScript Patterns & Extensibility

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/api-extension-patterns.

About the Author

Mike Amundsen is an author, speaker, and advisor on API architecture and distributed systems. He helps organizations design and implement resilient, evolvable systems by focusing on long-term thinking, composability, and interface modeling. His books include *RESTful Web API Patterns & Practices Cookbook* and *Design and Build Great Web APIs*. You can find more of his work at amundsen.com or subscribe to the *Signals Series* on Substack.

About the Signals Series

The Signals Series is a collection of short, forward-looking reports by Mike Amundsen that explore the shifting terrain of software architecture, APIs, and intelligent systems. Each issue focuses on a single concept, design pattern, or emerging challenge—bridging practical implementation with deep systems thinking.

The series is grounded in decades of real-world experience but looks ahead to the evolving needs of composable systems, autonomous agents, and machine-aware interfaces. It's written for engineers, architects, and leaders navigating the uncertainty of modern software.

For more reports, subscribe via Substack.