EcomLabs

FULL CODE AND CHECKLIST

Valid JSON from an LLM, every time

Structured outputs with a schema, zod validation and a retry that sends the error back to the model: the TypeScript function, its tests with a fake client, and the checklist.

FOR	TypeScript projects that ask Claude for structured data
TESTED WITH	Node 22.22.3 · zod 4.6.7 and 3.25.76 · @anthropic-ai/sdk 1.4.0 · TypeScript 5.7.3 · fake client
UPDATED	November 3, 2026
DOCS	platform.claude.com/docs/en/build-with-claude/structured-outputs
ARTICLE	ecomlabs.dev/g/json-en
AUTHOR	Luis Morales · @luismoralesdev

Contents

Before you start	What you will build
Step 01	The structured outputs request
Step 02	extract.ts
Step 03	Test it without calling the API
When you are done	Checklist

BEFORE YOU START

What you will build

One function, `extractJson`, that asks Claude for an object with structured outputs, validates it with zod and, when something fails, retries with the exact error sent back to the model, up to an attempt limit. A refusal stops; a `max_tokens` cutoff is retried with twice the tokens.

- Install in your project: `npm install @anthropic-ai/sdk zod`. Tested with `@anthropic-ai/sdk` 1.4.0 and zod 4.6.7.
- If your project is on zod 3.25.x, change `from "zod"` to `from "zod/v4"`. With `"zod"`, the SDK helper fails.
- Node 22.22.3 runs these `.ts` files with no extra flags.

> **CAUTION**
> The request shape follows Anthropic's docs as of November 2, 2026. We did not run it against the API: check the structured outputs page before you ship it.

STEP 01

The structured outputs request

The schema goes in `output_config.format`, with `type: "json_schema"`. The feature is generally available: no beta header. The old `output_format` parameter is deprecated.

```
{
  "model": "claude-sonnet-5-5",
  "max_tokens": 1024,
  "messages": [{ "role": "user", "content": "Extract the product from this supplier sheet: …" }],
  "output_config": {
    "format": {
      "type": "json_schema",
      "schema": {
        "type": "object",
        "properties": {
          "title": { "type": "string" },
          "price": { "type": "number" },
          "currency": { "type": "string", "enum": ["MXN", "USD"] },
          "sizes": { "type": "array", "items": { "type": "string" } },
          "inStock": { "type": "boolean" }
        },
        "required": ["title", "price", "currency", "sizes", "inStock"],
        "additionalProperties": false
      }
    }
  }
}
```

What structured outputs do not guarantee, per the docs:

CASE	WHAT HAPPENS	WHAT THE CODE DOES
Refusal	`stop_reason: "refusal"`, a 200, billed; the output may not match the schema	Throws, no retry
Cutoff	`stop_reason: "max_tokens"`, incomplete output	Retries with twice the `max_tokens`
`enum` casing	Not guaranteed, no error	Zod rejects it and the code retries
Unsupported constraints	`minimum`, `maxLength`… return a 400; the SDK moves them into the description	Zod enforces them

STEP 02

extract.ts

The whole function. It takes the client as an argument: `new Anthropic()` in production, a fake one in tests.

```
import type Anthropic from "@anthropic-ai/sdk";
import { zodOutputFormat } from "@anthropic-ai/sdk/helpers/zod";
import * as z from "zod";

export const Product = z.object({
  title: z.string(),
  price: z.number().positive(),
  currency: z.enum(["MXN", "USD"]),
  sizes: z.array(z.string()),
  inStock: z.boolean(),
});

type Client = {
  messages: { create(body: Anthropic.MessageCreateParamsNonStreaming): Promise<Anthropic.Message> };
};

export async function extractJson<S extends z.ZodType>(
  client: Client,
  schema: S,
  prompt: string,
  { model = "claude-sonnet-5-5", maxTokens = 1024, maxAttempts = 3 } = {},
): Promise<z.infer<S>> {
  const messages: Anthropic.MessageParam[] = [{ role: "user", content: prompt }];
  let lastError = "";

  for (let attempt = 1; attempt <= maxAttempts; attempt++) {
    const res = await client.messages.create({
      model,
      max_tokens: maxTokens,
      messages,
      output_config: { format: zodOutputFormat(schema) },
```

```
    });

    if (res.stop_reason === "refusal") {
      throw new Error("The model refused the request; retrying will not change that.");
    }
    if (res.stop_reason === "max_tokens") {
      lastError = `output cut off at max_tokens=${maxTokens}`;
      maxTokens *= 2;
      continue;
    }

    const text = res.content.flatMap((block) => (block.type === "text" ? [block.text] : [])).join("");
    try {
      const result = schema.safeParse(JSON.parse(text));
      if (result.success) return result.data;
      lastError = z.prettifyError(result.error);
    } catch (err) {
      lastError = (err as Error).message;
    }

    messages.push(
      { role: "assistant", content: text || "(empty)" },
      {
        role: "user",
        content: `Your reply does not match the JSON schema:\n${lastError}\nReply again with only the
corrected JSON object.`,
      },
    );
  }

  throw new Error(`No valid JSON after ${maxAttempts} attempts. Last error: ${lastError}`);
}
```

Calling it with the SDK (reads the key from `ANTHROPIC_API_KEY`; this file passed the typecheck and was not run):

```
import Anthropic from "@anthropic-ai/sdk";
import { extractJson, Product } from "./extract.ts";

const client = new Anthropic(); // reads ANTHROPIC_API_KEY
const sheet = "Sudadera Volcán, algodón 100 %, tallas S a L, $899 MXN, disponible.";
const product = await extractJson(client, Product, `Extract the product from this supplier sheet:\n${sheet}`);
console.log(product.price, product.inStock);
```

STEP 03

Test it without calling the API

Pass `extractJson` an object whose `messages.create` records each request (`JSON.parse(JSON.stringify(body))`, as it would go over the wire) and returns a message with `content: [{`

`type: "text", text }]` and whatever `stop_reason` you want to test. These are the broken replies we used, in order:

```
export const FAILURES: Reply[] = [
  { text: "```json\n" + VALID + "\n```" },
  { text: "Claro, aquí está el producto: " + VALID },
  { text: '{"title":"Sudadera Volcán","price":"899","currency":"mxn","sizes":["S","M","L"],"inStock":"yes"}'
},
  { text: '{"title":"Sudadera Volcán","price":899,"curr', stop_reason: "max_tokens" },
  { text: VALID },
];
```

With `node:test` we checked four things:

- With `maxAttempts: 5`, it works through all five replies and returns the valid object; each retry carries its error, and the fifth request raises `max_tokens` to 2048 without the truncated reply.
- The request carries `output_config.format` with `type: "json_schema"` and `additionalProperties: false`.
- A refusal ends in an error after a single request.
- Three invalid replies end in `No valid JSON after 3 attempts`.

Excerpt from the output of `node --test --test-reporter=spec extract.test.ts` (4 tests, 4 passed):

```
✓ retries each failure with its error and returns the valid object (9.5142ms)
✓ sends the schema in output_config.format (0.5039ms)
✓ stops on a refusal without retrying (0.6876ms)
✓ gives up after maxAttempts (1.6116ms)
```

WHEN YOU ARE DONE

Checklist

- [ ] The schema goes in `output_config.format` with `type: "json_schema"`, not just in the prompt.
- [ ] Every object has `additionalProperties: false`, and the fields you need are in `required`.
- [ ] Your code validates every reply with zod, structured outputs or not.
- [ ] `refusal` stops the process; `max_tokens` retries with more tokens, without sending the cut-off text back.
- [ ] A `JSON.parse` or zod failure goes back to the model with the exact error, as a `user` turn.
- [ ] There is an attempt limit and a clear error when it is reached.
- [ ] On zod 3.25.x you import from `"zod/v4"`.
- [ ] No protected health information (PHI) in field names, `enum`, `const` or `pattern`.