CLAUDE CODE CENTER OF EXCELLENCE

Force Multiplying Our People

Welcome to the Agentic Age. No question is bad. No topic is off limits. We are here for **you** — to help every engineer ship faster, think bigger, and build with confidence.

OPEN DOOR

Every question welcome, every skill level valued

ACCELERATE

10x your impact with agentic workflows

LEARN TOGETHER

Share patterns, avoid pitfalls, level up as a team

SHIP IT

Real results, real code, real production impact

— WHAT'S NEW — V2.1.217 – V2.1.221

What's New with Claude Code

✨

Features

Claude Opus 5 is now default, 1M context
Background sessions now commit & push work
Emoji shortcode autocomplete in prompts

🔧

MCP & Tooling

Fixed --mcp-config servers not connecting before first turn in print mode
Fixed memory leak in truncated MCP tool outputs
Subagents nest to depth 3 by default

📋

Perf & Fixes

Fixed quadratic slowdown in long-session message normalization
Fixed auto-compact never triggering for Opus 4.8 on Bedrock

DEMO DAY PRESENTATION

TOON Format

Token-Oriented Object Notation — a compact, structured translation layer for passing data to LLMs without wasting tokens.

COMPACT

Fewer tokens than equivalent JSON

STRUCTURED

Declared schema, not free-form text

LLM-READY

Designed for model consumption

VALIDATED

Array lengths catch truncated data

OVERVIEW

What Is TOON?

TOON stands for **Token-Oriented Object Notation** — a lightweight format purpose-built for LLM workflows where every token counts.

📋

JSON-Like Data

TOON represents the same structured data as JSON — objects, arrays, fields, and values — but in a far more compact form.

💰

Token-Efficient

Designed specifically for LLM prompts where token usage drives both cost and context window consumption.

👀

Human-Readable

Despite its compactness, TOON stays readable. Humans can inspect it, models can parse it — no custom decoder needed.

— THE PROBLEM

JSON Is Verbose

JSON repeats field names for every single object

In LLM prompts, repetition = wasted tokens = higher cost + less room for content

🔄

Repeated Field Names

In a list of 100 users, the keys `"id"`, `"name"`, and `"role"` appear 100 times each — adding hundreds of tokens that carry zero new information.

💸

Token Cost Creep

Extra tokens increase API cost directly. On large datasets — catalogs, forecasts, transaction records — JSON overhead compounds fast.

📏

Shrinking Context

Every token spent on syntax is a token stolen from actual content. Verbose payloads leave less room for instructions, examples, and reasoning.

— HOW IT WORKS

Declare Once, Stream Data

JSON — VERBOSE

users.json

```
// Field names repeat on every record [ { "id": 1, "name": "Ada", "role": "admin" }, { "id": 2, "name": "Bob", "role": "user" } ] ~60 tokens
```

TOON — COMPACT

users.toon

```
// Schema declared once, values streamed users[2]{id,name,role}: 1,Ada,admin 2,Bob,user // [2] → record count (truncation guard) // {id,name,role} → schema declared once // rows → values only, no repeated keys ~20 tokens
```

~66% fewer tokens for the same data — savings scale with record count.

— BENEFITS

Why TOON Works for LLMs

TOON is compact *and* structured — a rare combination that makes it uniquely suited to model consumption.

Fewer Tokens, Lower Cost

Eliminating repeated keys directly reduces token count. On large uniform datasets the savings are substantial — less spend per call, more headroom for actual content.

📖

Easier for Models to Follow

Uniform row-based data is easier for a model to scan and reason over than deeply nested JSON trees. A consistent pattern reduces ambiguity.

Built-In Truncation Guard

The declared record count — `[2]` in `users[2]` — lets a model (or your code) detect missing or truncated data before processing begins.

✅

Easier to Parse & Validate

The field list at the top acts as a schema header. Downstream code and the model both know exactly what columns to expect — no guessing from context.

— BEST USE CASES

Where TOON Shines

TOON works best with **uniform arrays of objects** — repeated records that share the same fields.

👥

User Lists

Passing user records, roles, or permission sets to a model for analysis, filtering, or summarization.

🛍️

Product Catalogs

SKU lists, pricing tables, inventory snapshots — any catalog with consistent fields across many items.

📈

Forecasts & Metrics

Time-series data, KPI snapshots, and trend tables that share a consistent schema across every row.

💳

Transaction Records

Financial transactions, event logs, audit trails — high-volume records where JSON overhead compounds quickly.

🔍

Search Results

Ranked result sets returned from a search API — consistent structure, many records, ideal for TOON encoding.

🧩

Any Repeated Structure

If your data is an array of objects where every object has the same fields, TOON is almost certainly a better fit than JSON.

— LIMITATIONS

When to Skip TOON

TOON is a specialized tool — not a universal JSON replacement. Reach for it only when it adds clear value.

🌳

Deeply Nested or Irregular Data

TOON's row-based format assumes flat, uniform records. Deeply nested objects or records with variable structure don't map cleanly to TOON's schema-then-rows model.

🔀

Records with Different Fields

If objects in your array don't share the same keys, TOON's single-schema header breaks down. Use JSON when field sets vary per record.

📋

Already Flat — Use CSV

If your data is a simple flat table with no nesting and no metadata, plain CSV is even more compact than TOON and universally understood.

⚖

Minimal Token Savings

For small payloads (a single object, a handful of records) the overhead of switching formats outweighs the token savings. Don't over-engineer small data.

Rule of thumb: If JSON is already working well and token savings are minimal, keep JSON. TOON earns its place on high-volume, uniform, repeated data.

— KEY TAKEAWAY

Declare Structure Once. Stream Data Compactly.

~66%
FEWER TOKENS VS JSON

1x
SCHEMA DECLARATION

∞
RECORDS, NO KEY REPETITION

💻

In Your App

Keep using JSON. TOON is a translation layer — your application logic stays the same.

🤖

Into the LLM

Convert to TOON before passing structured data to a model. Fewer tokens, same meaning.

🎯

Best Fit

Repeated records with consistent fields — user lists, catalogs, forecasts, transactions.

TOON is a compact, structured translation layer for JSON-like data in LLM prompts.

Use JSON in your app — use TOON when talking to the model.