— CIVILYTICS CONSULTING

U.S. Census of Governments Finance API

A public web API for exploring government finances, FY1967–FY2024

AUTHORS
Jared Knowles, Civilytics

PUBLISHED
2026-08-06

This document specifies the public, read-only web API over the Civilytics U.S. Census of Governments (COG) finance corpus, as built and deployed at `https://cog-api.civilytics.org`*. The API exposes government finance data spanning FY1967–FY2024 (no source data exists for FY1968 or FY1969) as parameterized HTTP endpoints returning JSON, alongside a bulk parquet download surface for full extracts. It is designed for direct consumption by business intelligence tools such as Tableau, and by analysts working in R, Python, or SQL. Every value can be returned in nominal or inflation-adjusted dollars and on a per-capita basis; spending and revenue are each available under more than one accounting concept (see "Which total do you mean?" below); and every response carries a provenance record for auditability. The endpoint, parameter, category, and scope tables below are generated at render time directly from the live service, so this document describes what the API does, not what it was meant to do.*

CONTENTS

Overview . . . 3
The primary use case . . . 3
Conventions . . . 4
Inflation and population adjustment . . . 6
Endpoints . . . 7
  Accepted parameters, by endpoint . . . 8
  Example: resolve a government . . . 10
  Example: the profile endpoint . . . 10
Which total do you mean? . . . 12
Reading a sample year . . . 13
Metric catalog . . . 13
Bulk download . . . 16

Connecting from Tableau 17
Scope, provenance, and versioning 18

Overview

The Census of Governments Finance API turns Civilytics' curated COG finance corpus into a live web service. Instead of downloading and cleaning raw Census files, a client points a tool at a stable URL and receives analysis-ready finance data — harmonized across a span of source-format changes, with inflation and population adjustment built in.

What the data covers

Dimension	Coverage
Years	FY1967 – FY2024 (no source data for FY1968 or FY1969)
Government types	State (0), County (1), Municipality (2), Township (3)
Governments in the registry	50 states, 3,061 counties, 20,106 municipalities, 17,119 townships (40,336 total)
Measures	Spending and revenue by functional category, plus cash and security holdings
Adjustments	Nominal or real (CPI-adjusted) dollars; total or per-capita

Special districts (type 4) and school districts (type 5) are not in the corpus and are out of scope for this API.

Two ways to consume it

1. **Interactive REST** — parameterized JSON endpoints for dashboards, lookups, and targeted queries. Ideal for Tableau, Power BI, and scripted analysis.
2. **Bulk download** — the full corpus as partitioned parquet files, for users who want to pull everything into a local database or warehouse.

The primary use case

The API is built first and foremost to answer one question cleanly:

> "Show me the complete financial history of this government — what it spends and where its money comes from — adjusted for inflation and population, so I can compare fairly across years."

A single call to the government **profile** endpoint returns that whole story: a government's spending and revenue, by category, for every year it appears in the corpus, in real per-capita dollars by default. That series drops directly into a Tableau time-series or small-multiples view.

Conventions

Base URL

```
https://cog-api.civilytics.org/api/v1/
```

`GET /openapi.json` and the browsable API docs (/__docs__/) are the two exceptions: they hang off the bare domain rather than `/api/v1/`. (Underscores in that path are backslash-escaped above so Markdown doesn't read them as emphasis markers.)

Access. Public, open, read-only. No API key is required for v1. All endpoints are HTTP `GET` and return JSON.

Response envelope. Every response uses a consistent structure, so a client parses all endpoints the same way. Here is a live call, trimmed to one row:


```
{
  "status": "success",
  "data": [
    {
      "year": 2022,
      "layer": "state",
      "canonical_govid": "120000226351",
      "gov_name": "FLORIDA",
      "spend_subtype": "capital",
      "category": "Police",
      "amt_nominal": 18240000,
      "codes_included": "F62",
      "aggregate_fallback": false,
      "scope_note": "state total; not limited to geography served by listed city/
county",
      "notes": "",
      "fips_state": "12"
    }
  ],
  "error": null,
  "meta": {
    "total": 2,
    "layer": "state",
    "subtype": null,
```



```
    "limit": 1000,
    "page": 0,
    "coverage": "all",
    "n_units_expected": 1,
    "n_units_reporting": 1,
    "is_census_year": true,
    "coverage_by_year": [
      {
        "year": 2022,
        "n_units_reporting": 1,
        "n_units_expected": 1,
        "is_census_year": true
      }
    ],
    "expenditure_concept": "primary",
    "version": "v1"
  },
  "provenance": {
    "verb": "cog_geographic_rollup",
    "years": 2022,
    "category": "Police",
    "basis": "harmonized",
    "expenditure_concept": "primary",
    "revenue_concept": "general",
    "scope": {
      "gov_types_included": [0, 1, 2, 3],
      "gov_types_excluded": [4, 5],
      "scope_note": "v0.1 covers state, county, city/municipality, and township
governments. Special districts (type 4) and school districts (type 5) are
excluded pending validation in a future cycle.",
      "govids_found": "120000226351",
      "govids_missing": []
    },
    "manifest": {
      "schema_version": 7,
      "pipeline_commit": "afb3a44",
      "built_at": "2026-08-04T15:10:22Z"
    }
  },
  "suggestions": []
}
```


Six keys appear in every response: `status`, `data`, `error`, `meta`, `provenance`, and `suggestions`. `provenance` records exactly how each number was produced — down to the corpus release (`manifest`), the accounting concepts in force, the harmonization and unit/inflation transformations applied, and the coverage of the request (see “Reading a sample year” below) — which makes every figure auditable and citable. The example above shows a curated subset (`verb`, `basis`, the concepts, `scope`, `manifest`); a live call also returns the raw SQL, the exact function call, and the full harmonization and transformation detail, omitted here only because they run several hundred words apiece. `suggestions` carries machine-readable hints (for example, a nearby recipe when a filter combination returns nothing); it is empty on most calls.

Inflation and population adjustment

Three query parameters are available on every monetary endpoint and are the heart of the "fair comparison across years" design.

Parameter	Values	Default	Meaning
`adjust`	`nominal`, `real`	`nominal`	Return dollars as reported, or inflation-adjusted.
`base_year`	e.g. `2023`	the corpus's latest published year	The dollar-year for `adjust=real`.
`per_capita`	`true`, `false`	`false`	Divide by the government's population that year.

- **Inflation adjustment** uses the FRED Consumer Price Index (CPI-U, BLS series `CPIAUCSL`, annual average). With `adjust=real&base_year=2023`, all amounts are expressed in 2023 dollars.
- **The default base year is dynamic, not pinned in the API's code.** Omit `base_year` and `adjust=real` resolves to whatever fiscal year is most recently published in the corpus — a live, unqualified `adjust=real` call made while rendering this document resolved to base_year: 2024, the corpus's newest year, with no API change required to make that so. The day a future release adds a newer fiscal year, the same unqualified call starts returning that year's dollars automatically.
- **This is already "as close to today's dollars as possible."** There is no more-current figure to adjust to: CPI-U is itself an annual average, published only once a year has closed, and the corpus's own spending and revenue data lag by roughly the same cycle. "Latest published year" and "today's dollars" are the same request.
- **For a report that must stay fixed** — so a chart doesn't quietly re-base itself between two runs made months apart, once a newer corpus release ships — pass an explicit `base_year=YYYY` rather than relying on the default. The default is the right choice for an always-current dashboard; an explicit year is the right choice for a point-in-time publication.
- **Per-capita** uses the population recorded for that government *in that fiscal year*, not a single modern estimate — so a per-capita trend line reflects real population change over time. On the `/balances` endpoint, per-capita divides a *stock* (reserves on hand) rather than a *flow*, and is not comparable to per-capita spending or revenue — see "Endpoints" below.
- The **profile** endpoint defaults to `adjust=real` and `per_capita=true`, because inflation- and population-adjusted is the natural way to read a multi-decade history.

Beyond these three, several endpoints accept additional parameters — `expenditure_concept`, `revenue_concept`, `coverage`, `subtype`, `basis`, `recipe`, and `govids` among them. Rather than list every parameter for every route by hand (the previous draft of this document did, and

went stale), the next section generates the accepted-parameter list for each endpoint directly from the live API.

Endpoints

All 18 endpoints, generated from `GET /openapi.json` at render time, one table per group (a single table with a `Group` column repeated down the left overflowed its column width; splitting avoids that):

Service metadata

Path	Purpose
/api/v1/	API metadata root
/api/v1/health	Liveness

Self-documentation

Path	Purpose
/api/v1/data-dictionary	Human-readable data dictionary (markdown)
/api/v1/llms.txt	Agent-facing plain-text guide
/openapi.json	Machine-readable OpenAPI description (this table's source)

Government lookup

Path	Purpose
/api/v1/governments	Search / resolve a government
/api/v1/governments/{govid}	Government identity + latest snapshot

Government-scoped

Path	Purpose
/api/v1/governments/{govid}/balances	Cash and security holdings for one government (filterable)
/api/v1/governments/{govid}/peer-comparison	Target vs. its peer cohort on one category (per-capita)
/api/v1/governments/{govid}/peers	Peer cohort (same type, similar population)
/api/v1/governments/{govid}/profile	Full spending + revenue history for one government (real, per-capita by default)
/api/v1/governments/{govid}/revenue	Revenue time series for one government (filterable)

Path	Purpose
/api/v1/governments/{govid}/spending	Spending time series for one government (filterable)

Fleet / nationwide

Path	Purpose
/api/v1/revenue	Flat nationwide revenue slice
/api/v1/rollups	Geographic aggregates by layer (state/county/city)
/api/v1/spending	Flat nationwide spending slice

Catalogs

Path	Purpose
/api/v1/categories	Metric dictionary (drives BI field lists)
/api/v1/recipes	Harmonization recipe catalog (multi-code, cross-vintage series) – use a

A few of these are easy to miss in a table and worth calling out directly:

- `/balances` (`/governments/{govid}/balances`) returns cash and security holdings — Fund Balances, Insurance Trust Balances, and Retirement System Holdings — for one government. These are **stocks**, not flows: a balance is a point-in-time reserve, not an annual spending or revenue figure, and it should never be plotted on the same per-capita axis as spending or revenue.
- `/recipes` is the catalog of harmonization recipes: named, multi-code series that bridge item-code changes across Census reporting-format vintages (for example, the pre-2012 wide-era codes and their modern equivalents). Pass `recipe=` instead of `category` / `subtype` on `/profile` , `/spending` , `/revenue` , or `/balances` to pull a long, break-free series.
- `/llms.txt` is a plain-text, agent-facing guide to the API — written for an LLM or automated client to self-orient without parsing the full OpenAPI document.
- `/data-dictionary` is the human-readable counterpart: field and code documentation in Markdown.

Accepted parameters, by endpoint

Path	Accepted parameters
/api/v1/	(none declared)
/api/v1/health	(none declared)
/api/v1/data-dictionary	(none declared)

Path	Accepted parameters
/api/v1/llms.txt	(none declared)
/api/v1/governments	limit, page, q, state, type
/api/v1/governments/{govid}	(none declared)
/api/v1/governments/{govid}/balances	adjust, base_year, basis, category, limit, page, per_capita, recipe, subtype, years
/api/v1/governments/{govid}/peer-comparison	adjust, base_year, category, coverage, expenditure_concept, limit, page, years
/api/v1/governments/{govid}/peers	coverage, n, year
/api/v1/governments/{govid}/profile	adjust, base_year, basis, category, expenditure_concept, limit, page, per_capita, recipe, revenue_concept, subtype, years
/api/v1/governments/{govid}/revenue	adjust, base_year, basis, category, expenditure_concept, limit, page, per_capita, recipe, revenue_concept, subtype, years
/api/v1/governments/{govid}/spending	adjust, base_year, basis, category, expenditure_concept, limit, page, per_capita, recipe, revenue_concept, subtype, years
/api/v1/revenue	adjust, base_year, basis, category, expenditure_concept, govid, limit, page, per_capita, recipe, revenue_concept, state, subtype, type, years
/api/v1/rollups	adjust, base_year, category, coverage, expenditure_concept, govids, layer, limit, page, per_capita, state, subtype, years
/api/v1/spending	adjust, base_year, basis, category, expenditure_concept, govid, limit, page, per_capita, recipe, revenue_concept, state, subtype, type, years
/api/v1/categories	pattern, type
/api/v1/recipes	pattern

This table is generated by asking each endpoint a question it can't answer: every call above appends a nonsense query parameter (`__cog_spec_probe__=1`) and reads the resulting `400` . The API's own rejection message enumerates exactly what that route accepts — for example, `GET /api/v1/rollups?bogus=1` currently answers *"Unknown query parameter: bogus. accepted here: adjust, base_year, category, coverage, expenditure_concept, govids, layer, limit,*

page, per_capita, state, subtype, years (/api/v1/rollups). Parameters are not silently ignored." This document asks the server rather than asserting an answer, so the table above cannot drift out of sync with what is actually deployed.

Example: resolve a government


```
{
  "status": "success",
  "data": [
    {
      "canonical_govid": "121011212191",
      "gov_name": "BROWARD COUNTY",
      "govs_type": 1,
      "type_label": "county",
      "fips_state": "12",
      "fips_county": "011",
      "fips_place": null,
      "legacy_govs_id": "101006006",
      "first_year": 1967,
      "last_year": 2024,
      "census_geoid": "12011",
      "population_acs": 1940907,
      "pop_confidence": "exact",
      "id_source": "census_pid"
    }
  ],
  "error": null,
  "meta": {
    "total": 1,
    "limit": 100,
    "page": 0,
    "version": "v1"
  },
  "provenance": null,
  "suggestions": []
}
```


Example: the profile endpoint


```
{
  "status": "success",
  "data": [
    {
      "year": 2022,
```



```
      "canonical_govid": "121011212191",
      "gov_name": "BROWARD COUNTY",
      "subtype": "capital",
      "category": "Police",
      "amt_nominal": 23590000,
      "codes_included": "F62",
      "aggregate_fallback": false,
      "amt_per_capita_nominal": 12.0474,
      "pop_source": "census_f33",
      "amt_real": 25288697.5953,
      "amt_per_capita_real": 12.9149,
      "notes": "",
      "flow": "expenditure"
    }
  ],
  "error": null,
  "meta": {
    "total": 2,
    "limit": 1000,
    "page": 0,
    "govid": "121011212191",
    "subtype": null,
    "basis": "harmonized",
    "version": "v1"
  },
  "provenance": {
    "verb": "cog_spending",
    "years": 2022,
    "category": "Police",
    "basis": "harmonized",
    "expenditure_concept": "primary",
    "revenue_concept": "general",
    "scope": {
      "gov_types_included": [0, 1, 2, 3],
      "gov_types_excluded": [4, 5],
      "scope_note": "v0.1 covers state, county, city/municipality, and township
governments. Special districts (type 4) and school districts (type 5) are
excluded pending validation in a future cycle.",
      "govids_found": "121011212191",
      "govids_missing": []
    },
    "manifest": {
      "schema_version": 7,
      "pipeline_commit": "afb3a44",
      "built_at": "2026-08-04T15:10:22Z"
    }
  },
  "suggestions": []
}
```


Each row is one government × year × category × subtype. A call without `years=` or `category=` on `/profile` returns the government's *entire* history, ready to chart.

Which total do you mean?

Two concept models govern every dollar figure this API returns, and neither existed in the previous draft of this document.

Expenditure: `primary ⊂ direct ⊂ total`.

- `primary` (the default) is spending on services, excluding debt service.
- `direct` adds interest on long-term debt — this is the Census Bureau's published **Direct Expenditure** figure.
- `total` adds payments to other governments on top of `direct`. It is available on **exactly one endpoint** — `GET /governments/{govid}/spending` — because that is the only route guaranteed to describe a single government. Every fleet or aggregate route (`/spending`, `/rollups`, `/profile`) rejects `expenditure_concept=total` with a 400: summing "total" spending across several governments double-counts every dollar transferred between them.

Revenue: `general ⊂ total`.

- `general` (the default) is the Census Bureau's published **General Revenue** — own-source revenue plus federal, state, and local intergovernmental aid. For **local governments**, a `general`-concept response broken out by category is materially incomplete on its own: intergovernmental aid to local governments sits on aggregate `*89` item codes, which the harmonized, category-broken-out `general` response excludes by design (for Atlanta, GA, FY2022, the excluded aid comes to 19.4% of its true general revenue). That money is not missing from the API — it is recoverable via the `ig_*_?89_wide` recipes that endpoint's `suggestions` payload names for the affected category and government — but a client reading only the category-broken-out `general` rows at face value will understate local-government revenue by a nontrivial share, not a rounding error.
- `total` adds utility revenue, liquor-store revenue, and insurance-trust revenue: `Total = General + Utility + Liquor Store + Insurance Trust`, the Census Bureau's own identity. Unlike expenditure `total`, revenue `total` **is available on every revenue endpoint, including the fleet routes** — because it only adds a government's *own* revenue, never a transfer it received, so summing it across many governments double-counts nothing.

A live example. Broward County (govid 121011212191), FY2022, summed across every category returned by its `/spending` and `/revenue` endpoints:

Concept	Value
Expenditure — `primary`	$2,903,208,000
Expenditure — `direct`	$3,042,136,000
Expenditure — `total`	$3,042,913,000
Revenue — `general`	$2,718,871,000
Revenue — `total`	$2,888,390,000

The gap between `primary` and `direct` is interest on long-term debt; the gap between `direct` and `total` is payments to other governments; the gap between `general` and `total` revenue is utility, liquor-store, and insurance-trust revenue. None of these are rounding — they are different, individually valid answers to "how much did this government spend/collect," and citing a figure without naming its concept invites a reader to compare it to a number that answers a different question.

Reading a sample year

The Census of Governments is a **complete census only in years ending in 2 or 7** (2012, 2017, 2022, ...); every other year is a **sample**. Large governments sit in a **certainty stratum** and are collected every year regardless — small governments do not, and drop in and out of the corpus between census years.

A concrete, live-verified example: Georgia cities reporting Police spending.

Year	Census/sample	Cities reporting	Cities expected
2022	census year	393	567
2024	sample year	94	567

In the sample year, collection outside the certainty stratum thins sharply — most of Georgia's smaller cities simply are not in a FY2024 sweep, alongside a smaller and separate effect where a city genuinely has no Police line item (for example, because a county sheriff provides the service). The API reports the raw reporting count in `provenance.coverage`; it does not attempt to separate "not sampled this year" from "reports zero in this category," so this figure should be read as a coverage disclosure, not a response rate.

The rule this implies, applied throughout this API's client-facing material: a figure whose denominator is "every government of type X" needs a census year, or `coverage=census` on the endpoints that support it (`/rollups` , `/peer-comparison`). A comparison of *named or population-matched* large governments is safe in the newest year, because large governments are exactly the ones the certainty stratum guarantees.

Metric catalog

Categories come in three kinds — `expenditure` , `revenue` , and `balance` (a stock, served only by `/balances`) — 26 expenditure categories, 11 revenue categories, and 3 balance categories in the live catalog. The groupings below are a hand-curated reading aid; the full, generated catalog follows and is the authoritative list.

Expenditure — by domain

Domain	Categories
Public safety	Police, Fire, Corrections, Judicial & Legal, Protective Inspection
Transportation & utilities	Highways, Other Transportation, Transit Utilities, Water Utilities, Electric Utilities, Gas Utilities, Sewerage & Solid Waste
Health & human services	Public Welfare, Health & Hospitals, Insurance Trust Benefits, Veterans Services
Education	Education K-12, Higher Education, Other Education
Culture, housing & environment	Parks & Recreation, Libraries, Housing & Community, Natural Resources
Government & finance	General Government, Interest on Debt, Liquor Stores

Revenue — by source

Group	Categories
Own-source taxes	Property Tax, General Sales Tax, Individual Income Tax, Corporate Income Tax, Other Taxes
Own-source charges	Current Charges, Miscellaneous Revenue
Intergovernmental	IG Federal, IG State, IG Local
Insurance trust (in `total` only)	Insurance Trust Revenue

Balances — cash and security holdings

Category	What it is
Fund Balances	General-fund cash and security holdings
Insurance Trust Balances	Reserves held for employee retirement, unemployment, workers' comp, and other insurance-trust funds
Retirement System Holdings	Pension and retirement system assets

Full generated catalog, one table per `category_type` (a combined table with a repeated `Type` column overflowed its column width; splitting avoids that and reads better besides):

Expenditure

Category	Subtypes	Item codes
Corrections	capital, intergovernmental, operations	10

Category	Subtypes	Item codes
Education K-12	capital, intergovernmental, operations	7
Electric Utilities	capital, interest, intergovernmental, operations	6
Fire	capital, intergovernmental, operations	5
Gas Utilities	capital, interest, intergovernmental, operations	6
General Government	capital, intergovernmental, operations	22
Health & Hospitals	capital, intergovernmental, operations	15
Higher Education	capital, intergovernmental, operations	9
Highways	capital, intergovernmental, operations	8
Housing & Community	capital, intergovernmental, operations	5
Insurance Trust Benefits	insurance_benefits	6
Interest on Debt	interest	1
Judicial & Legal	capital, intergovernmental, operations	5
Libraries	capital, intergovernmental, operations	5
Liquor Stores	capital, operations	3
Natural Resources	capital, intergovernmental, operations	5
Other Education	assistance, capital, intergovernmental, operations	6
Other Transportation	capital, intergovernmental, operations	13
Parks & Recreation	capital, intergovernmental, operations	10
Police	capital, intergovernmental, operations	5
Protective Inspection	capital, intergovernmental, operations	5
Public Welfare	assistance, capital, intergovernmental, operations	15
Sewerage & Solid Waste	capital, intergovernmental, operations	10
Transit Utilities	capital, interest, intergovernmental, operations	9
Veterans Services	assistance, capital, operations	4
Water Utilities	capital, interest, intergovernmental, operations	6

Revenue

Category	Subtypes	Item codes
Corporate Income Tax	own_source	1
Current Charges	liquor_store, own_source, utility	24
General Sales Tax	own_source	1

Category	Subtypes	Item codes
IG Federal	federal	17
IG Local	local_aid	14
IG State	state	14
Individual Income Tax	own_source	1
Insurance Trust Revenue	insurance_trust	11
Miscellaneous Revenue	own_source	10
Other Taxes	own_source	22
Property Tax	own_source	1

Balance

Category	Subtypes	Item codes
Fund Balances	general	3
Insurance Trust Balances	other_insurance_trust, unemployment_trust, workers_comp_trust	4
Retirement System Holdings	employee_retirement	7

Every expenditure and revenue row carries several forms of the same figure — nominal and inflation-adjusted dollars, total and per-capita, and (where applicable) a share of the requested total — selected with the `adjust`, `base_year`, and `per_capita` parameters described above rather than as separate fields to look up.

Subtype rollups. Expenditure rows carry a `subtype` of `operations`, `capital`, `intergovernmental`, `assistance`, `interest`, or `insurance_benefits` — this is exactly the vocabulary behind the `primary` / `direct` / `total` concepts above. Revenue subtypes are `own_source`, `federal`, `state`, `local_aid`, `utility`, `liquor_store`, or `insurance_trust` — the vocabulary behind `general` / `total`. Filter to one subtype with `subtype=`, or leave it unset to get every subtype for the requested category.

Bulk download

For users who want the entire corpus rather than targeted queries, the full dataset is published as Hive-partitioned parquet and can be queried in place with DuckDB — no download step required:

```
INSTALL httpfs; LOAD httpfs;
SELECT *
FROM read_parquet(
  'hf://datasets/civilytics/us-cog-finance/year=2022/*.parquet'
)
WHERE category = 'Police';
```

The bulk files carry the same category names and definitions as the API, so results are directly comparable. See `GET /data-dictionary` for the exact column layout.

Connecting from Tableau

The client builds their own Tableau workbooks; this section covers only what is specific to this API.

1. **Web Data Connector / JSON connection** — point Tableau at an endpoint URL, for example `https://cog-api.civilytics.org/api/v1/governments/121011212191/profile`.
2. `.data` **is the extract root.** Tableau reads rows from the `data` array; the other five envelope keys (`status`, `error`, `meta`, `provenance`, `suggestions`) are not rows and are silently dropped by a `.data`-only extract. That includes `provenance.coverage` — the block that discloses whether a year is a census or a sample and how many governments are expected versus reporting (see “Reading a sample year”). A workbook built this way inherits the coverage caveat above without any way to see it in Tableau; note it in the workbook if the extract crosses a sample year. The one partial exception is `/rollups`: it mirrors its coverage block into `meta` — the same `meta$coverage` field the “Reading a sample year” example above reads — so an extract built to include `meta` alongside `.data`, not `.data` alone, keeps that disclosure for `/rollups` specifically. `/spending`, `/revenue`, and the per-government routes have no such mirror, so noting the caveat in the workbook remains the only option there.
3. `/peer-comparison`**’s summary rows are not additional peers.** That endpoint appends `summary_p25` / `summary_p50` / `summary_p75` rows to the `data` array — per-category cohort quantiles, distinguishable from actual peer rows only by the `role` field. Summed or averaged blindly in a Tableau extract, they silently corrupt any aggregate built from that sheet. Filter on `role` before aggregating.
4. **Large or whole-nation extracts** should use the bulk parquet source above rather than paginating the REST API — `limit` is capped at 1,000 rows per request.

Scope, provenance, and versioning

- **Scope (v1), generated from `GET /api/v1/`:** *gov types 0-3, FY1967-2024 (no source data FY1968, FY1969)*. Government types 0–3 only (state, county, municipality, township).
- **Provenance, a live example.** Every response records how each number was produced. A live `/rollups` call's `provenance` carries 27 top-level fields in total; the list below is a curated subset (the rest — the raw SQL, the exact function call, and per-request detail like which government IDs matched — is real but not legible here). A short field next to a long JSON value reads better as a list than as a two-column table, so that's how this one is shown:
- `verb`: cog_geographic_rollup
- `years`: 2022
- `category`: All Categories
- `basis`: harmonized
- `expenditure_concept`: primary
- `revenue_concept`: general
- `scope`: {"gov_types_included":[0,1,2,3],"gov_types_excluded":[4,5],"scope_note":"v0.1 covers state, county, city/municipality, and township governments. Special districts (type 4) and school districts (type 5) are excluded pending validation in a future cycle.","govids_found":"120000226351","govids_missing":[]}
- `manifest`: {"schema_version":7,"pipeline_commit":"afb3a44","built_at":"2026-08-04T15:10:22Z"}
- **Versioning.** The API is served under `/api/v1/`. Data releases are versioned independently and surfaced in every response's `provenance`, so a dashboard can pin to a known release.

This document describes the API as deployed at `https://cog-api.civilytics.org/api/v1/`. Its endpoint, parameter, category, and scope tables are generated at render time from the live service; a static copy of this PDF reflects the service's state as of the date above.