AI Inference Census: how your data is handled

Venturi TS AI Factory Sizer v4.0.1. One page for your security, privacy and compliance reviewers (FERPA, HIPAA, GRAMA). It describes what the software does, and the test suite checks it.

In one paragraph

Usage files are read **inside a web browser**, either on your own computer (customer-run census) or on the Venturi TS consultant's laptop. In the browser they are reduced to **counts**: tokens per day, requests per minute, the hourly curve, peak rates, and numbers of people and seats. Only those counts leave the browser. Prompts, responses and documents are never read. Email addresses and user IDs are turned into salted hashes in the browser, and the salt is discarded when the page closes. The sizer server does not store or log the census.

Step by step

Step	Where	What happens	What is kept
1. Export	Your systems	You export usage with your own admin tools or our script (below). You can **strip** content columns and **hash** identities before anything leaves your systems.	Your files, on your systems
2. Read	Browser	Files are streamed and parsed row by row. Only the mapped fields are read: time, model, token counts, durations, user, app, cost and status.	Nothing is uploaded
3. Reduce	Browser	Rows become totals per minute, day and month, plus token-size histograms. Each user is normalized (lowercase; any domain aliases you choose) and hashed with a random salt created when the page loads.	Hashes stay in browser memory only
	Browser	The **census profile** goes to the sizer to become	Not stored or

4. Size	→ sizer server	workloads (see *What leaves the browser*).	logged on the server
5. Scenario	Browser storage	The workloads and a short census summary are saved in your browser's local storage, as for any scenario.	Counts only (see below)
6. Report	Sizer server	A PDF or HTML report is generated on request.	See *Reports*

What leaves the browser (the census profile)

For each **stream** (one model, app, team, deployment or seat product):

- working-day totals of input, cached and output tokens, requests and spend
- tokens by hour of day
- token-size percentiles
- the measured peak: requests per second at the 99th-percentile minute, requests in flight, and the throttled share
- the number of people (a count only)
- monthly seasonality multipliers
- for seat products, licensed, active and unused seat counts and total activity

For each **source**:

- the product name and your note on how it connects
- the period covered and time detail
- rows read and people counted (counts)
- whether people are matchable across products
- the throttled count
- the names (not contents) of any content columns found
- your overlap decisions

Never included: raw rows, prompts, responses, request bodies, emails, user IDs, hashes, API keys or IP addresses.

People and the hash salt

- Identifiers are normalized and then hashed in the browser with a **salt generated**

randomly when the census page loads. The salt is held in memory only. It is never sent, saved or written to any file, and it is gone when the page is closed or reloaded.

- Hashes are used only to count unique people across products, and never leave the browser. A later census in a new session cannot be linked to an earlier one.
- If you prefer, pseudonymize identities yourself before export with `venturi_usage_export.py hash` and a salt that only you hold. Use the same salt on every file so people still match across products.

Content (prompts and responses)

- The census reads only the mapped fields. Some exports also **contain** content: Bedrock request and response bodies, LiteLLM `messages` / `response`, and Open WebUI chats. When such a column is present, the census page flags it. It never reads or keeps it; while a row is parsed, the row is held only briefly in browser memory.
- Recommended: run `venturi_usage_export.py strip …` first, so content never leaves your systems. The Bedrock strip keeps the response time (`latencyMs`) and removes the bodies.

Customer-run census (raw files never leave your premises)

1. Your team signs in to the sizer and runs the census in your own browser.
2. **Save census summary** writes `census-summary-YYYY-MM-DD.json` : the census profile above, as readable JSON (format `venturi-census-summary/1`). Open it and review it before sharing.
3. You send only that file. The consultant loads it into the sizer.

Overlap and people decisions are made in your browser and are fixed in the summary.

What is stored, where

- **Your browser** (local storage): the scenario and its workloads, plus the census summary. The summary holds product names, periods, row and people counts, matchability, throttled counts, overlap decisions and notes. Clearing the browser's site data removes it at any time.
- **Sizer server:** nothing from the census. The `/api/import/usage` request is processed and discarded; it is not logged or stored. The server keeps sign-in records and a count of workloads modeled per person, as for any use of the sizer.
- **Reports:**

- A PDF report generated by a signed-in person is filed, with a short note, to that person's contact record in Venturi TS's CRM (Zoho Bigin).
- The report's census section holds the counts above. It **never** holds a customer's negotiated seat price unless a Venturi TS **admin** generated it.
- Ask your Venturi TS contact to delete CRM records.

The export script

- `venturi_usage_export.py`, version 4.0.1, uses Python 3.8+ and the standard library only. It runs on your computer with your keys; keys are never sent to Venturi TS.
- Its header lists every API endpoint it calls and every field it writes. The `strip` and `hash` commands make no network calls.
- The census page and the download response (`X-Script-SHA256`) show its **SHA-256**. Check that the file you review is the file you run: `shasum -a 256 venturi_usage_export.py` (macOS / Linux) or `Get-FileHash venturi_usage_export.py` (Windows).