Numinor SAM Amplifier Construct Data v1.0

English: Numinor SAM Amplifier Construct Data **中文:** Numinor SAM Amplifier 构建数据

---

1. Product Identity

Product: SAM Amplifier 构建数据 (Construct Data) **Version:** v1.0 **Methodology Reference:** *Numinor SAM Amplifier Whitepaper v1.6* (Jan 2026) **Reference Implementation:** github.com/Numinor-Systems/sam-amplifier-construct-reference — MIT License

What this product is

A pre-engineered **stock-level peer / upstream / downstream relationship graph**, derived from ChinaScope's SAM product taxonomy and supply-chain disclosures. Each trading day, for each A-share listco, Numinor publishes the set of peer / upstream / downstream counterparties with revenue-share-weighted edge weights.

The buyer applies this graph as an **aggregation operator** against their own factor model. The output is the buyer's existing factors, "amplified" through the SAM-defined network — a known-good methodology for adding cross-sectional information to multi-factor portfolios.

What this product is NOT

- **Not an alpha signal.** This is reshaped data; the buyer's alpha depends on their factor model.
- **Not raw ChinaScope data.** That is sold separately by ChinaScope. This product *uses* that data as input and reshapes it.
- **Not a pre-amplified factor file.** We do not ship "amplified versions of specific factors" — the buyer applies the graph to their own factor file.
- **Not residualized against Numinor's illustrative factors.** The graph is factor-model agnostic.

Who it's for

Quant equity buyers operating in Chinese A-shares who already use a multi-factor model (Barra, MSCI, FactSet, CIQ, or proprietary) and want to add a network-aggregation channel to their existing factor exposures.

---

2. Schema

Primary schema (parquet and CSV — identical columns)

Column	Type	Description
`trade_date`	`date32`	The trading date this graph snapshot applies to (Asia/Shanghai timezone).
`focal_ts`	`string`	Focal stock ticker in format `NNNNNN.XX` (e.g., `000001.SZ`, `600000.SH`).
`relation_type`	`string`	One of: `"peer"`, `"upstream"`, `"downstream"`.
`counterparty_ts`	`string`	Related stock ticker (same format as `focal_ts`).
`weight`	`float64`	Revenue-share-derived edge weight in `[0.0, 1.0]`.
`source_basis`	`string`	Which ChinaScope source this edge derives from: `"sam_product_overlap"` (peer) or `"sam_supply_chain_v4_corein"` (upstream/downstream).
`source_rpt_date`	`date32`	Source data effective date — the SAM rpt date used to compute this edge (PIT-correct).
`source_publish_date`	`date32`	When the governing filing became public (max over the edge's two parties — the edge exists only once BOTH inputs are public). *(schema v1.1)*
`eff_date`	`date32`	The date this edge was admitted into the dataset: `source_publish_date + lag_days`. **Strict audit column** — `eff_date ≤ trade_date` holds on every served row. *(schema v1.1)*

Example rows

trade_date	focal_ts	relation_type	counterparty_ts	weight	source_basis	source_rpt_date
2026-05-30	000001.SZ	peer	600036.SH	0.034	sam_product_overlap	2025-12-31
2026-05-30	000001.SZ	peer	601398.SH	0.018	sam_product_overlap	2025-12-31
2026-05-30	000001.SZ	upstream	002475.SZ	0.022	sam_supply_chain_v4_corein	2025-12-31
2026-05-30	000001.SZ	downstream	600519.SH	0.015	sam_supply_chain_v4_corein	2025-12-31
2026-05-30	000001.SZ	peer	000063.SZ	0.011	sam_product_overlap	2025-12-31

Type / value rules

- `trade_date`: ISO 8601 calendar date (`YYYY-MM-DD`). Trading-day-aligned (Shanghai/Shenzhen Stock Exchange calendar). Non-trading days are NOT published.
- `focal_ts`, `counterparty_ts`: A-share listco tickers only. Format `NNNNNN.SZ` for Shenzhen, `NNNNNN.SH` for Shanghai. Non-listco entities are absent — SAM's source data operates at the listco level by construction.
- `relation_type`: Lowercase only. Exactly one of three values.
- `weight`: 6 decimal places of precision. Always positive (no negative weights). For peer edges, weights within a `focal_ts` sum to ~1.0 (revenue-share allocation). For upstream/downstream, weights sum to the focal's economic exposure share (may be < 1.0 if some counterparties are non-listed and absent from this product).
- `source_basis`: String literal. New values may be added in future versions (e.g., `"sam_supply_chain_v4_bizsup"`) but existing values are stable.
- `source_rpt_date`: The SAM source data effective date. Always ≤ `trade_date - publish_lag_days` (see §4 PIT discipline).
- `source_publish_date`: the filing's publication date. `source_rpt_date ≤ source_publish_date` always.
- `eff_date`: `source_publish_date + 30 calendar days` — when the edge became admissible. `eff_date ≤ trade_date` on every row (the strict no-look-ahead invariant, verifiable directly from the data).

What's NOT in the schema (intentional)

- Self-edges (`focal_ts == counterparty_ts`) are excluded by construction.
- Non-listco focal or counterparty stocks are excluded.
- Zero-weight edges are excluded.
- No "raw" company identifiers (CSF-ids, etc.) — only stock tickers. The buyer should not need to handle non-listco entities.

3. File Layout & Delivery

S3 prefix structure (two tiers, same schema)

API and Sandbox tiers consume the **same data shape** from two disjoint S3 prefixes — the only difference is recency. Sandbox tier sees the data 30 calendar days behind the API tier (T−30d). This is enforced at the storage layer: a Sandbox API key cannot read API-tier prefixes, and vice versa.

Parquet (primary):

```
s3://numinor-construct-data/sam-amplifier/api/parquet/
├── historical/sam_amplifier_2016_<latest_year>.parquet     (single concatenated file)
└── year=YYYY/month=MM/day=DD/data.parquet                  (Hive-partitioned daily, T+1)

s3://numinor-construct-data/sam-amplifier/sandbox/parquet/
├── historical/sam_amplifier_2016_<latest_year_minus_1mo>.parquet
└── year=YYYY/month=MM/day=DD/data.parquet                  (Hive-partitioned daily, T+1 of T-30d data)
```

CSV (secondary, ChinaScope-pipeline-compatible):

```
s3://numinor-construct-data/sam-amplifier/api/csv/      (same shape, files as .csv.zip)
s3://numinor-construct-data/sam-amplifier/sandbox/csv/  (same shape, files as .csv.zip)
```

Both formats contain identical data. Parquet is snappy-compressed; CSV is ZIP-compressed (ChinaScope convention). Hive partitioning (`year=YYYY/month=MM/day=DD/`) is buyer-friendly for `pyarrow.dataset`, Spark, DuckDB, and Athena range queries.

Historical bulk file vs daily partitions

- **Historical bulk file** (`historical/sam_amplifier_2016_<latest_year>.parquet`) is a single concatenated parquet of every trading day from 2016-01-04 through the cutoff date. Provided for buyer convenience at onboarding — one download, ~12 GB, no concatenation work needed.
- **Daily partitions** (`year=YYYY/month=MM/day=DD/data.parquet`) are the ongoing delivery channel. Daily refresh writes one new partition per trading day.

The historical bulk is rebuilt once per quarter; between rebuilds, buyers consume partitions for the gap between bulk cutoff and current date.

File sizes (approximate)

File	Parquet	CSV.zip
Historical bulk (2016-2025, ~375M rows)	~12 GB	~30 GB
Typical daily partition (~150K rows)	~3-5 MB	~8-12 MB
Rebal-day partition (after semi-annual SAM update, ~500K rows)	~15-25 MB	~40-70 MB

v1.0 launch data vintage

> ***At v1.0 launch, the dataset is complete and current through 2026-05-15*** *(10.4 years, 2016-01-04 → 2026-05-15). New subscribers receive the full historical depth immediately. The* ***daily T+1 partition-advancement cadence described below is the steady-state design****; it activates once Numinor's upstream SAM/market ingestion is productionized on a daily schedule (a fast-follow shortly after launch). Because the underlying SAM classification data updates* ***semi-annually*** *with company filings, the day-to-day informational delta is small — the data's analytical value refreshes on the SAM-update cycle, not daily. Subscribers are notified when daily auto-advancement goes live; the historical depth and methodology are fully usable from day one.*

Update cadence & SLA (steady-state)

- **API tier daily refresh:** new `year=YYYY/month=MM/day=DD/data.parquet` published by **07:00 Asia/Shanghai** for trading day T, written on T+1. (ChinaScope upstream delivery typically completes ~06:30 SGT; Numinor pipeline gates on the upstream `.ready` marker before proceeding — see `daily_refresh_spec.md`.)
- **Sandbox tier daily refresh:** copies API tier file dated `today − 30 days` into the sandbox prefix nightly. Sandbox tier therefore lags API by exactly 30 calendar days.
- **No publication** on Chinese A-share market holidays.
- **Historical bulk:** rebuilt quarterly. Buyers can opt into a fresh historical bulk pull or stay on partitions between rebuilds.
- **Methodology stability:** Numinor commits not to change the methodology mid-version. Methodology updates trigger a version bump (e.g., v1.1) with 60-day advance notice.

---

4. PIT Discipline

The product is **point-in-time correct**: a graph snapshot dated `trade_date = D` uses only ChinaScope source data that was already publicly available on day `D`.

Publish lag rule

For every source used in v1.0:

```
source_publish_date + 30 calendar days ≤ trade_date
```

This buffer models realistic vendor delivery (ChinaScope T+1) + institutional-buyer ingestion / recompute / deployment lag (typically 3-4 weeks combined) + a modest conservatism cushion.

> ***Measured availability floor (2026-06):*** *across 5,675 filings spanning the FY2025 annual + Q1 reporting season, 99% of SAM records were delivered within* ***4 days*** *of publish_date (99.9% within 30). The 30-day buffer is therefore ~4 days of measured availability + ~26 days of buyer-workflow allowance — conservative by construction. It matches the convention used in Numinor's SAM PM construct, keeping the lag rule consistent across our catalog.*

This is mechanically enforced by the pipeline — no row appears in the snapshot for `trade_date = D` whose source data was not publicly available on `D − 30 calendar days`.

What this means for the buyer

- The buyer never receives an edge that "looked into the future" relative to its `trade_date`.
- Backtests using this data inherit the PIT discipline automatically — no extra work on the buyer's side.
- For audit / transparency, the buyer can verify the **strict** no-look-ahead rule directly from the served data: `eff_date ≤ trade_date` for every row, where `eff_date = source_publish_date + 30` (all three dates ship in the schema as of v1.1). The weaker report-date proxy (`source_rpt_date + 30 ≤ trade_date`) also holds, but the publish-date-anchored check is the airtight version — it proves the edge used only filings that were public in time.

Relationship to the WP

The published **SAM Amplifier Whitepaper v1.6** used a more conservative **60-day lag** for its empirical analysis. The v1.0 construct data ships with **30-day lag**, providing fresher data for production use. A buyer running the WP's methodology end-to-end on this construct data will get **structurally similar but slightly different numbers** than the WP — the signal is identical, but ~30 days more recent SAM data is included in the construct's PIT window.

Can the buyer change the lag?

- **Tighter lag (<30 days)**: not available through this product. Requires licensing raw ChinaScope SAM data and running your own pipeline.
- **Looser lag (>30 days, more conservative)**: easily applied buyer-side — simply consume the graph at `trade_date + extra_days` in your pipeline. Costs nothing, adds conservatism.

The 30-day choice is intentionally on the permissive side so buyers can tighten their own pipeline if they prefer; they cannot loosen ours below 30 without raw data.

Sandbox tier recency (separate from PIT)

The PIT discipline above (`source_rpt_date + 30 days ≤ trade_date`) is identical across API and Sandbox tiers. What differs is **recency of delivery**:

Tier	Latest `trade_date` available
API	T (i.e., yesterday's trading day, written at T+1 by 06:00 Asia/Shanghai)
Sandbox	T − 30 calendar days

Both tiers contain the **same row content for any given `trade_date`** — the only difference is that Sandbox cannot see rows newer than 30 calendar days back. Sandbox is intended for research, prototyping, and product evaluation; API is intended for production use. Buyer types are distinct (retail/students vs institutional), and the tiers do not bundle.

---

5. Graph Construction — Numinor's Engineering Value-Add

ChinaScope ships raw SAM data: a per-company × per-product revenue mix table, and a product-level supply chain taxonomy (v4). For stock-level quantitative analysis, the buyer needs this raw data **assembled into a stock-to-stock**

graph with appropriate weights, time-varying snapshots, and PIT discipline. That assembly is the engineering work Numinor performs.

Peer edges (relation_type = "peer")

For each focal stock `i` at `source_rpt_date`, the focal's reported revenue is distributed across SAM product nodes (its product mix). For each product node `p`, all other listcos with revenue exposure to `p` are candidate peers, weighted by the product of revenue shares:

```
peer_weight[i → j on product p] = revenue_share[i, p] × revenue_share[j, p]
```

Summing over all shared products gives the peer edge weight:

```
peer_weight[i → j] = (Σ_p revenue_share[i, p] × revenue_share[j, p]) / Σ_p revenue_share[i, p]
```

Self-exclusion is enforced ($i \neq j$). Weights within a `focal_ts` are renormalized to sum to ~1.0.

Upstream / downstream edges (relation_type = "upstream" or "downstream")

For each focal stock `i` at `source_rpt_date`, the focal's product mix is mapped to SAM supply chain v4's `COREIN` (core-input) edges. Each focal product's COREIN parents become upstream candidates; each focal product's COREIN children become downstream candidates. Weights are derived from the focal's revenue exposure to the source product × all listcos' revenue exposure to the counterparty product:

```
upstream_weight[i → j on edge p → q]   = revenue_share[i, q] × revenue_share[j, p]
downstream_weight[i → j on edge p → q] = revenue_share[i, p] × revenue_share[j, q]
```

(p = upstream product, q = downstream product on the SAM supply chain edge.)

Summed across all relevant supply-chain edges, normalized per `focal_ts`.

Universe filtering

After computing raw edges: 1. Drop self-edges (`focal_ts == counterparty_ts`) 2. Drop edges where either side is not an A-share SH/SZ/KC/CYB listco 3. Drop edges with weight below a numerical floor (1e-6) — irrelevant noise 4. Drop edges where source-data PIT condition is violated (`source_rpt_date + 30 days > trade_date`)

Daily refresh

At each new `trade_date`: 1. Inspect ChinaScope's daily delta files for any SAM source data that newly crossed `publish_date + 30 days` 2. Recompute affected focals' peer / upstream / downstream edges 3. Emit a delta file containing only changed rows since the previous trading day

Most days, the delta is small (no new SAM filings have just crossed the buffer). Post-filing-season days (typically May/June and October/November) have larger deltas as new SAM data crystallizes into the graph.

What the buyer pays for vs. does themselves

Step	Done by Numinor	Buyer would need to do
Read ChinaScope SAM raw tables	✓	Schema knowledge, multi-table joins
Apply PIT discipline	✓	Track each filing's `publish_date` and apply buffer
Compute peer weights (revenue-share math)	✓	Implement the aggregation correctly per date
Map SAM products to supply chain v4 edges	✓	Cross-reference two ChinaScope tables
Filter to A-share listco universe	✓	Maintain listco lists, handle delistings/IPOs
Daily refresh pipeline	✓	Run own pipeline daily, monitor for delivery failures
Maintain historical snapshots	✓	Build own historical store

Doing this end-to-end from raw ChinaScope data is approximately **2-3 weeks of focused data engineering work** for an experienced team, plus ongoing maintenance. Numinor delivers it pre-built, refreshed daily, with documented methodology.

---

6. Universe Rules

Inclusion

- **All A-share listcos** with SAM coverage at `trade_date`, traded on Shanghai (SH) or Shenzhen (SZ) exchanges.
- Stocks under suspension on `trade_date` are **included** in the graph (a suspension is temporary; their network position remains valid).
- Stocks delisted before `trade_date` are **excluded** from that date forward.
- Stocks IPO'd after `trade_date` are **excluded** prior to listing.

Exclusion (by design)

- Beijing Stock Exchange (BJ): excluded. The Amplifier methodology in our v1.6 WP was validated on SH+SZ only.
- STAR Board (KC) and ChiNext (CYB): **included** in the universe (despite being SH/SZ-listed, these are growth boards; they had been considered for exclusion but were included after the v1.6 WP review).

Coverage start

- The dataset covers `2016-01-04 → present`. SAM data prior to 2016 has insufficient depth for reliable peer/upstream/downstream relationship inference.

---

7. Methodology Overview (Companion Doc)

The data dictionary defines **what** is delivered. The companion methodology document (`methodology.md`, shipped with the reference implementation) describes **how** to apply it. Key sections:

How peer weights are computed

For each focal stock `i` at `source_rpt_date`, the focal's revenue is distributed across SAM product nodes (the focal's product mix). For each product node `p`, all other listcos with revenue exposure to `p` become peers, weighted by:

```
peer_weight[i → j on product p] = revenue_share[i, p] × revenue_share[j, p]
```

Summing over all shared products gives the peer edge weight:

```
peer_weight[i → j] = Σ_p (revenue_share[i, p] × revenue_share[j, p]) / Σ_p revenue_share[i, p]
```

Self-exclusion is enforced (i ≠ j). Renormalized within `focal_ts` so weights sum to ~1.0.

How upstream/downstream weights are computed

For each focal `i` at `source_rpt_date`, the focal's disclosed supplier-customer relationships (with affiliate rollup) become upstream / downstream edges, weighted by the focal's economic dependence on the counterparty (trade amount / balance amount / revenue share, depending on the disclosure detail level).

How the buyer applies the graph

Companion reference code (Python) provides:

```python
import numinor_sam_amp as ns

# Load buyer's factor file (their property)
factors = load_my_factors()  # DataFrame[trade_date, ts_code, factor_name, value]

# Load Numinor graph
graph = ns.load_construct_data("2026-05-30", path="s3://...")

# Apply peer aggregation: for each (focal_ts, factor_name), compute weighted
# average of factor values across peers
amplified = ns.aggregate(factors, graph, relation_type="peer")

# Now `amplified` has the same schema as `factors`, but each value is
# the peer-aggregated version. Buyer uses this alongside original factors.
```

This is a 20-line operation, fully vectorizable. The reference code includes a complete worked example.

---

8. API Specification

All API access is via signed URL minting. The buyer authenticates once with their API key; the API returns a time-limited S3 URL the buyer downloads from directly.

Tier access policy (read this first)

Tier	REST endpoints below (`GET /historical` , `/day/` , `/range` , `POST /query`)	In-Sandbox MCP tools (`query_construct_data` , etc.)
API tier	✅ Yes, signed URLs minted against `s3://numinor-construct-data/sam-amplifier/api/...`	✅ Yes (if subscribed to Sandbox separately)
Sandbox tier	❌ **No** — returns `HTTP 403 {"error": "rest_requires_api_tier"}` . Sandbox tier accesses data only through metered, egress-firewalled in-Sandbox MCP tools	✅ Yes, reads against `sandbox/` prefix (T−30d lagged)

Why the REST endpoints are API-tier only: a Sandbox-tier subscriber paying $20/mo who could `GET /historical` and receive a signed URL to the full T−30d bulk file (~531 million rows) would effectively extract the whole dataset for a flat fee, defeating the egress firewall + output gateway model documented in `sandbox_security_guardrails_v1.0.md` . Sandbox tier therefore accesses Construct Data **only** through the metered, per-row-capped, egress-firewalled MCP tools running inside the Sandbox kernel — never via raw REST signed URLs. The API-tier subscription is what unlocks raw signed-URL access.

Base URL

```
https://api.numinor.io/v1/constructs/sam-amplifier
```

Authentication

```
Authorization: Bearer <numinor_api_key>
```

- **Key model:** one API key per user, with the SKU-access allowlist derived from the user's active subscriptions. The same key works across all SKUs the user has paid for; revocation is a single action.
- **Default expiration:** none. API keys do not expire automatically.
- **Rotation:** client-controlled via subscriber dashboard. Subscribers may rotate at any cadence; we recommend 90 days as security best practice.
- **Revocation:** immediate. Compromised keys can be invalidated instantly via the dashboard.
- **Multiple keys per subscriber:** supported, useful for separating dev / staging / production access. All keys belong to the same user and share the same SKU allowlist.

Endpoints

`GET /historical`

Returns a signed URL to the historical bulk file (single concatenated parquet covering 2016 → latest cutoff).

Query parameters: - `format` : `parquet` (default) or `csv`

Response (200):

```
{
  "url": "https://numinor-construct-data.s3.amazonaws.com/sam-
        amplifier/api/parquet/historical/sam_amplifier_2016_2026.parquet?X-Amz-Signature=...",
  "expires_at": "2026-05-30T16:00:00Z",
  "file_size_bytes": 12884901888,
  "format": "parquet",
  "schema_version": "v1.0",
  "row_count": 375000000,
  "cutoff_date": "2026-05-27"
}
```

Tier note: Sandbox-tier keys receive `HTTP 403 {"error": "rest_requires_api_tier"}` from this endpoint. Sandbox-tier subscribers should use the `query_construct_data` MCP tool inside their Sandbox session for data access (see `sandbox_ai_agent_v1.0.md`).

GET `/day/{YYYYMMDD}`

Returns a signed URL to a specific trading day's partition file.

Path: - `YYYYMMDD`: trading date in compact format (e.g., `20260530`)

Query parameters: - `format`: `parquet` (default) or `csv`

Response (200):

```
{
  "url": "https://numinor-construct-data.s3.amazonaws.com/sam-
        amplifier/api/parquet/year=2026/month=05/day=30/data.parquet?X-Amz-Signature=...",
  "expires_at": "2026-05-30T16:00:00Z",
  "file_size_bytes": 4194304,
  "format": "parquet",
  "schema_version": "v1.0",
  "trade_date": "2026-05-30",
  "row_count": 152840
}
```

404: if `YYYYMMDD` is not a published trading date.

GET `/range`

Returns signed URLs for a date range (one URL per trading day).

Query parameters: - `start`: `YYYY-MM-DD` - `end`: `YYYY-MM-DD` - `format`: `parquet` or `csv`

Response: list of signed URLs, one per trading day in the requested range.

POST `/query` (small filtered queries)

For ad-hoc queries (small responses returned inline as JSON, not signed URLs).

Body:

```
{
  "trade_date": "2026-05-30",
  "focal_ts": "000001.SZ",
  "relation_type": "peer"   // optional, defaults to all three
}
```

Response (200):

```
{
  "rows": [
    {"trade_date": "2026-05-30", "focal_ts": "000001.SZ", "relation_type": "peer", "counterparty_ts":
        "600036.SH", "weight": 0.034, ...},
    ...
  ]
}
```

GET `/manifest`

Returns a single JSON document summarizing the current state of the data feed for the bearer's tier. Cached server-side ~60 sec; clients can poll without rate-limit concern.

Response (200):

```
{
  "sku": "sam-amplifier-construct-v1",
  "schema_version": "1.0",
  "tier": "api",
  "status": "green",
  "latest_trade_date": "2026-05-28",
  "latest_published_at_utc": "2026-05-28T23:07:42Z",
  "historical_coverage": {
    "start": "2016-01-04",
    "end": "2026-05-28",
    "trading_days": 2517,
    "row_count": 532086359
  },
  "historical_bulk": {
    "path": "sam-amplifier/api/parquet/historical/sam_amplifier_2016_2026.parquet",
    "size_bytes": 2789015432,
    "sha256": "...",
    "last_rebuilt_utc": "2026-04-01T07:00:00Z",
    "next_rebuild_utc": "2026-07-01T07:00:00Z"
  },
  "delivery": {
    "format_primary": "parquet",
    "formats_available": ["parquet", "csv.zip"],
    "partition_layout": "hive year=YYYY/month=MM/day=DD",
    "s3_region": "ap-northeast-2"
  },
  "pit": { "lag_days": 30, "rule": "source_rpt_date + 30 calendar days <= trade_date" },
  "upstream": {
    "vendor": "ChinaScope",
    "last_freshness_check_utc": "2026-05-28T22:32:00Z",
    "lag_minutes_from_typical": 0,
    "consecutive_red_days": 0
  },
  "eta_iso": null,
  "links": {
    "data_dictionary": "https://numinor.io/lab/data/sam-amplifier/dictionary",
    "methodology": "https://numinor.io/lab/data/sam-amplifier/methodology",
    "status_page": "https://numinor.io/status",
    "support_email": "support@numinor.io"
  },
  "request_endpoints": { "...": "..." }
}
```

For Sandbox-tier keys: `tier: "sandbox"`, `latest_trade_date` is lagged 30 calendar days, `historical_bulk.path` points to `sandbox/` prefix, and the response includes a `tier_notice` field encouraging upgrade.

`status` field values:

Value	Meaning
`green`	All upstream markers fresh; latest partition published on schedule
`delayed`	ChinaScope upstream is late but within 09:00 SGT recovery window; auto-retry pending
`red`	Past 09:00 SGT recovery deadline; today's partition will not publish today; eta_iso may be present
`warming_up`	Cron 0 has not yet run for the first time (pre-launch / fresh deployment)

Sample JSON files in the canonical repo at `data_product/samples/`: `_freshness.sample.{green,red,warming_up}.json`, `_heartbeat.sample.json`, `manifest.sample.{api,sandbox}.json`.

Rate limits

Endpoint	Limit	Per-response cap
`POST /query` (inline filtered)	100 req/min per API key, burst 20 in 10 sec	10,000 rows per response. Exceed → HTTP 413, response includes a suggested `/range` URL
`GET /historical`	unlimited	n/a (returns signed S3 URL)
`GET /day/{YYYYMMDD}`	unlimited	n/a (returns signed S3 URL)
`GET /range`	unlimited	n/a (returns list of signed S3 URLs)
`GET /manifest`	1000 req/min (server-cached 60 sec)	small JSON, see shape above

Signed URL validity: 4 hours from issuance. Within the validity window, downloads from the signed URL itself are unlimited (handled by S3, not the Numinor API). If the URL expires before completion, request a fresh one — no penalty.

What counts as a "request": one HTTP call to a Numinor API endpoint. Pulling a 12 GB historical dump from a signed S3 URL counts as exactly one request to `/historical`, regardless of how many S3 byte-range fetches happen during the download.

Burst handling: if a subscriber needs to exceed sustained rate limits temporarily (e.g., onboarding rush), contact `support@numinor.io` — temporary lifts are routine.

Coming in v1.1: MCP

We will publish an MCP server exposing the same API as native LLM tools (`get_sam_amplifier_graph(date, focal_ts)`) once MCP infrastructure matures across major model providers. This is forward-looking — subscriber feedback shapes priorities.

---

9. Quickstart: From Subscription to First Value in 5 Minutes

1. Get your API key

Issued via email at onboarding. Store as `NUMINOR_API_KEY` env var:

```
export NUMINOR_API_KEY="nm_live_..."
```

2. Pull today's partition

Python (pandas):

```python
import os, requests, pandas as pd

key = os.environ["NUMINOR_API_KEY"]
date = "20260530"
resp = requests.get(
    f"https://api.numinor.io/v1/constructs/sam-amplifier/day/{date}",
    headers={"Authorization": f"Bearer {key}"}
).json()
df = pd.read_parquet(resp["url"])
print(df.head())
```

Python (polars):

```python
import polars as pl
df = pl.read_parquet(resp["url"])
```

Python (pyarrow dataset — for Hive-partitioned range reads):

```python
import pyarrow.dataset as ds
# Pull a range of URLs from /range, then read them as a Hive-partitioned dataset:
dataset = ds.dataset(signed_urls, partitioning="hive", format="parquet")
df = dataset.to_table().to_pandas()
```

R:

```r
library(arrow)
library(httr)
resp <- httr::GET("https://api.numinor.io/v1/constructs/sam-amplifier/day/20260530",
                  httr::add_headers(Authorization=paste("Bearer", Sys.getenv("NUMINOR_API_KEY"))))
url <- jsonlite::fromJSON(httr::content(resp, "text"))$url
df <- arrow::read_parquet(url)
```

DuckDB (no Python needed):

```
INSTALL httpfs;
LOAD httpfs;

-- After getting the signed URL via API
SELECT * FROM read_parquet('<signed_url>') WHERE focal_ts = '000001.SZ' LIMIT 100;
```

3. Pull the historical dump (one-time)

```
resp = requests.get(
    "https://api.numinor.io/v1/constructs/sam-amplifier/historical",
    headers={"Authorization": f"Bearer {key}"}
).json()
df_history = pd.read_parquet(resp["url"])
```

4. Apply to your factors

The reference code (MIT-licensed, on GitHub) does this in a few lines:

```
import numinor_sam_amp as ns
factors_df  = pd.read_parquet("my_barra_factors.parquet")    # your factor data
graph_df    = ns.load_construct_data(date="2026-05-30", api_key=key)
amplified   = ns.aggregate(factors_df, graph_df, relation_type="peer")
amplified.to_parquet("my_peer_amplified_factors.parquet")
```

That's it. The reference repo has full worked examples in `examples/` covering: - Loading & validating the data dictionary - Applying to a buyer's factor file - Running the v1.6 WP's full robustness battery on the buyer's data - Multi-offset evaluation - ΔSharpe / ΔICIR analysis vs the buyer's base portfolio

5. Ask Gandalf for help

Stuck on integration? Open Gandalf (your onsite AI assistant), ask:

> *"How do I load `sam_amplifier_delta_20260530.parquet` in R?" "Why does focal `300750.SZ` have 80 peers? Is that normal?" "How do I apply the peer aggregation to my Barra-Long file?"*

Gandalf has context on the data dictionary, the reference code, and methodology — and replies in seconds.

---

10. Onboarding Checklist

When a new subscriber comes online:

Step	Owner	Time
1. Subscription contract signed	Sales	—
2. API key generated, emailed to subscriber's technical lead	Numinor ops	< 1 hour
3. Subscriber tests `/manifest` endpoint to confirm access	Subscriber	5 min
4. Subscriber downloads historical dump	Subscriber	30-60 min (12 GB)
5. Subscriber validates schema against this data dictionary	Subscriber	30 min
6. Subscriber runs reference code on their factor file	Subscriber	30-60 min
7. Subscriber's first amplified factor file produced	Subscriber	end of day 1
8. Optional: live integration call with Numinor team	Joint	1 hour

Total time from contract to producing usable amplified factors: **< 1 business day**.

---

11. Versioning & Changelog

Version	Date	Notes
v1.0	2026-05-28	Initial release. Mirrors *SAM Amplifier WP v1.6* (Jan 2026) methodology.

Roadmap

- **v1.1 (planned Q3 2026):** MCP server for LLM-native data access.
- **v1.2 (planned Q4 2026):** Optional channel decomposition file (separates peer / upstream / downstream into distinct files for buyers who want them).
- **v2.0 (planned 2027):** SAM Supply Chain v5 incorporation (when ChinaScope ships v5).

Subscribers receive 60-day advance notice of any breaking changes. Methodology stability within a version is guaranteed.

---

12. FAQ

Q: How is this different from buying ChinaScope's raw SAM data directly?

A: ChinaScope sells the raw `sam_product_calc`, `supply_chain_trade`, etc. tables. To use them for stock-level quant analysis, you'd need to (a) build the peer/upstream/downstream graph topology, (b) do affiliate rollup, (c)

implement PIT discipline, (d) maintain ongoing daily refreshes. Numinor does all this for you, with a documented methodology and a reference implementation. You pay for the engineering + ongoing maintenance, not the data.

Q: What's the licensing relationship with ChinaScope?

A: This product is a **derivative work** of ChinaScope's SAM data under a separate Numinor-ChinaScope agreement. Subscribers do not need a separate ChinaScope license to use the construct data; the license is bundled into the Numinor subscription. If the subscriber wants the raw SAM tables, they can license those separately from ChinaScope.

Q: Can I use the graph for purposes other than factor amplification?

A: Yes. The graph is reusable across any application that benefits from peer/upstream/downstream stock relationships — e.g., risk modeling, sector rotation, event-driven trading, counterparty exposure. The reference code shows the factor-amplification use case (matching the v1.6 WP), but the data itself is general-purpose.

Q: Does the methodology actually deliver the alpha shown in the WP?

A: The WP shows that *applied to Numinor's illustrative 22-factor Tushare-derived base*, SAM Amplifier delivered +0.149 ICIR and +0.54–+0.72 Sharpe lift in OOS 2022-2026. The buyer's actual lift depends on the buyer's factor model — typically higher with cleaner, more orthogonal base factors (Barra, MSCI, FactSet). The construct data is sold as the *engineering input*, not as a guaranteed alpha output. The reference code lets the buyer measure their own lift in their own factor stack.

Q: How often does ChinaScope update SAM data?

A: SAM data updates semi-annually with company financial filings (typically March/April for FY data, August/September for H1 data). Supply chain disclosures follow the same cadence. Numinor's pipeline incorporates updates via daily deltas — most days the delta is small (only metadata changes), but post-filing-season days can have larger deltas as new SAM data crystallizes.

Q: What if a focal stock has no SAM coverage at a given date?

A: It's simply absent from the graph for that date. The buyer's pipeline should treat missing focal_ts as "no amplification available" — fall back to the buyer's raw factor values. The reference code demonstrates this fallback.

Q: Can I run my own backtest to verify the WP's claims before purchasing?

A: Yes — two paths:

1. **Sandbox tier** (see numinor.io/lab/data/sam-amplifier for current pricing): runs the same Construct Data through `sandbox.numinor.io`, **real-time (same recency as the API)**. Sandbox is the primary verification path before committing to the API tier. The full historical depth (2016 → today) is available; you can replicate the WP's full robustness battery using the reference code on your own factor model.
2. **Custom historical pull:** for institutional buyers who need to validate against API-fresh data before subscribing, contact `support@numinor.io`. Custom historical pulls are arranged case-by-case.

There is no money-back guarantee on the API tier — Sandbox is the verification step. Buyers prove value in Sandbox before subscribing to API.

Q: What's the difference between Sandbox and API tiers?

A: Both serve the **same Construct Data** (identical schema, identical row content for any given `trade_date`). The difference is **recency**:

- **API tier:** sees today's data (T) by T+1 07:00 Asia/Shanghai. Production-grade. Direct S3 + signed-URL access for inclusion in the buyer's own pipeline.
- **Sandbox tier:** sees the same real-time data as the API (no delay). Hosted in `sandbox.numinor.io` notebook environment. Token-metered for research, prototyping, and product evaluation. **Cannot be substituted for API in**

production — Sandbox is in-Realm and egress-controlled (you get results, never the bulk feed); production pipelines need the API.

Current pricing for both tiers lives on numinor.io/lab/data/sam-amplifier. The two tiers do not bundle — institutional buyers needing both pay for both.

Q: What data is available *inside* the Sandbox environment beyond the Construct Data itself?

A: Sandbox subscribers can use the following data inside `sandbox.numinor.io` without leaving the environment:

- **Tushare A-share market data** (OHLCV, daily returns, volume) — for factor computation, backtests, and visualization. Refreshed weekly; in-Sandbox use only.
- **Numinor's 22 illustrative factors** (`numinor_22.parquet` + `numinor_22_F2_extras.parquet`) — the same factors used in the WP v1.6 replication. These are **demo factors, not the product**; the product is the Construct Data graph. Use these to reproduce the WP results, then replace with your own factor model.
- **SAM source data** (product master, supply chain v4 COREIN edges, sam_product_calc) — read-only inputs to the Construct Data pipeline, available for inspection.
- **Your uploaded factor file** (BYOFactor) — buyer-uploaded panel for the buyer's own factor model evaluation. Per-user isolated storage.

All data in the Sandbox can be *used* by buyer code (read, transformed, joined, backtested), but raw files cannot be exfiltrated. Outputs (PDF, Excel, PNG) are allowed via the output gateway. See *Sandbox Data Layout v1.0* for the full layout specification.

Q: I want raw Tushare data for my own production environment outside Sandbox. Where do I get it?

A: Subscribe to Tushare directly. Numinor does not redistribute Tushare data — what lives in our Sandbox stays in our Sandbox. Tushare offers tier-based subscriptions for A-share market data; buyers handle their own Tushare relationship for production use.

Q: Can I drive the Sandbox using my own AI agent (Claude Code, OpenAI Assistants, etc.)?

A: Yes — Sandbox is **MCP-native by design**. Connect your own agent to Numinor's MCP server at `wss://sandbox.numinor.io/mcp/v1` using your Numinor API key. Your agent's LLM calls go to your own LLM provider (you pay them directly); Numinor charges only for Sandbox compute (kernel time, data access, output generation) against your Sandbox token balance. The same MCP server powers our default in-product agent ("Gandalf") and your BYOA setup — same tool surface, same data access. See *Sandbox AI Agent Architecture v1.0* for the full MCP tool inventory.

---

13. Contact & Support

Channel	Use case
Gandalf (in-product)	First-line technical questions, code examples, methodology clarifications
`support@numinor.io`	Everything else — production issues, subscriptions, data quality, methodology

Appendix A: Schema Reference Card (Printable)

```
Numinor SAM Amplifier 构建数据 v1.0
Format: parquet (.parquet) or CSV/ZIP (.csv.zip)

  trade_date         date32     YYYY-MM-DD, Shanghai trading days
  focal_ts           string     NNNNNN.SZ or NNNNNN.SH
  relation_type      string     "peer" | "upstream" | "downstream"
  counterparty_ts    string     NNNNNN.SZ or NNNNNN.SH
  weight             float64    [0.0, 1.0], 6-decimal precision
  source_basis       string     "sam_product_overlap" | "sam_supply_chain_v4_corein"
  source_rpt_date    date32     YYYY-MM-DD, PIT (source_rpt_date + 30 ≤ trade_date)

Universe:        A-shares (SH + SZ + KC + CYB); not BJ
Listco-only:     non-listcos absent by SAM source construction
Coverage:        2016-01-04 → present, daily on trading days
PIT discipline:  source_rpt_date + 30 days ≤ trade_date
Tier recency:    API → trade_date up to T (delivered T+1)
                 Sandbox → trade_date up to (T − 30 calendar days)
S3 layout:       s3://numinor-construct-data/sam-amplifier/{api,sandbox}/{parquet,csv}/
                   historical/sam_amplifier_2016_YYYY.parquet        (bulk file)
                   year=YYYY/month=MM/day=DD/data.parquet            (Hive partitions)
Delivery:        Signed-URL minting via https://api.numinor.io/v1/constructs/sam-amplifier/
                   GET /historical · GET /day/{YYYYMMDD} · GET /range · POST /query
```

End of SAM Amplifier 构建数据 v1.0 specification. Methodology reference: Numinor SAM Amplifier Whitepaper v1.6. Reference implementation: github.com/Numinor-Systems/sam-amplifier-construct-reference (MIT License).