GraphRAG Programmatic API Reference

> **Audience:** Developers integrating GraphRAG into back-end services, agents, batch pipelines, or other server-to-server workloads. **Version:** v2 (APIM gateway)

All programmatic traffic is served through the **Azure API Management (APIM) gateway**. The gateway is the single, supported public entry point for the GraphRAG service: it terminates TLS, authenticates the caller via a subscription key, applies per-subscription throttling, and forwards requests to the backend over a private channel using a managed identity.

> The legacy direct-backend paths (customer-issued `gro_*` API keys and customer Service Principal Bearer tokens) are no longer publicly supported. Existing tenants using those flows will be migrated to APIM subscription keys.

Table of Contents

---

1. Quick Start

You need two things:

Item	Where to get it
Gateway URL	`https://graphrag-apim-wg3temevssbja.azure-api.net/graphrag`
Subscription key	Emailed after checkout and visible in the dashboard under **API Access → APIM subscription key**

Send your first query:

```
curl -X POST "https://graphrag-apim-wg3temevssbja.azure-api.net/graphrag/hybrid/query" \
  -H "Content-Type: application/json" \
  -H "Ocp-Apim-Subscription-Key: <YOUR_SUBSCRIPTION_KEY>" \
  -H "X-API-Version: v2" \
  -d '{"query": "What are the payment terms?"}'
```

Successful response:

```
{
  "response": "The payment terms are net 30 days from invoice date...",
  "route_used": "hipporag2_community",
  "citations": [
    { "document_id": "contract-001", "chunk_id": "s-0042", "document_title": "Acme MSA", "score
  ],
  "evidence_path": ["..."],
  "metadata": { "duration_ms": 1843, "model": "gpt-4.1-mini" }
}
```

---

2. Authentication

A single auth method is supported for programmatic access: **APIM subscription key**.

2.1 Required Headers

Every request must include:

Header	Value	Purpose
`Ocp-Apim-Subscription-Key`	Your subscription key	Identifies and authenticates the caller
`X-API-Version`	`v2`	Selects the v2 API version

The gateway validates the subscription key, then transparently attaches a backend-trust header and a managed-identity Bearer token before forwarding the request. **Callers never need to acquire or send Azure AD tokens themselves.**

2.2 Obtaining a Subscription Key

Each API customer gets their **own** APIM subscription, scoped to their group and tied to their purchased API add-on tier — subscriptions are isolated per tenant, not shared.

After a successful API add-on checkout, GraphRAG automatically:

1. Activates the purchased API tier for the account.
2. Creates or updates the customer's APIM subscription for that tier.
3. Fetches the APIM subscription key.
4. Emails the key to the account address and makes the current key available in the dashboard.

The dashboard **API Access** panel shows the API tier, APIM provisioning/email-delivery status, gateway URL, and required headers. Use **Reveal key** to fetch the current APIM key from Azure APIM on demand; the key is not stored by GraphRAG. Use **Copy cURL** for a ready-to-run request example, or **Emergency rotate** if the key is leaked. Emergency rotation invalidates the previous key immediately.

The customer can start using the API as soon as the key is available, though newly-created APIM keys can take a few seconds to propagate through the gateway. If the key email cannot be delivered immediately, GraphRAG retries delivery in the background, and the dashboard can still reveal the current key once APIM provisioning is complete. Contact `support@hulkdesign.com` if no key appears in the dashboard, or if a key must be revoked or reissued.

2.3 Key Hygiene

- Treat subscription keys like passwords. Store them in a secrets manager (Azure Key Vault, AWS Secrets Manager, environment variable from a secret store).
- Never commit keys to source control.
- Never expose keys in browser or mobile client code — the gateway is for **server-to-server** traffic only.

---

3. Conventions

3.1 Base URL

All endpoints in this document are relative to:

`https://graphrag-apim-wg3temevssbja.azure-api.net/graphrag`

3.2 Content Type

All `POST` request bodies are `application/json`. All responses are `application/json` unless noted.

3.3 Idempotency

`GET` endpoints are idempotent. `POST /hybrid/query` is read-only — it never mutates server state — but is computationally expensive and should not be retried aggressively. `POST /hybrid/index/documents` mutates the caller's indexed corpus by starting an asynchronous indexing job. `POST /hybrid/cache/prewarm` only warms the process-local retrieval cache and is safe to call before large-corpus queries.

---

4. Query Endpoint

`POST /hybrid/query`

The primary endpoint. Submits a natural-language query against the indexed knowledge base.

4.1 Request Body

Field	Type	Required	Default	Description
`query`	string	Yes	—	The natural-language question.
`response_type`	string	No	`detailed_report`	Output style. See §4.3.
`force_route`	string	No	*auto*	Override route selection. See §4.4.

Field	Type	Required	Default	Description
`language`	string	No	*auto*	Reply language as ISO 639-1 (e.g. `en`, `ja`, `de`). Defaults to the query's detected language.
`folder_id`	string	No	*all*	Restrict retrieval to a single folder/collection within your group.
`include_context`	boolean	No	`false`	If `true`, returns the full retrieved context (assembled evidence) in the response metadata under `llm_context`.
`relevance_budget`	number	No	*route default*	Retrieval thoroughness budget between `0.0` (fast) and `1.0` (most thorough).
`synthesis_model`	string	No	*server default*	Override the synthesis LLM deployment name (e.g. `gpt-4.1`, `gpt-4.1-mini`).

Note: the field is `query`, **not** `question`.

4.2 Example Request

```
curl -X POST "https://graphrag-apim-wg3temevssbja.azure-api.net/graphrag/hybrid/query" \
  -H "Content-Type: application/json" \
  -H "Ocp-Apim-Subscription-Key: <YOUR_KEY>" \
  -H "X-API-Version: v2" \
  -d '{
    "query": "Compare liability caps across all vendor contracts",
    "response_type": "comprehensive",
    "force_route": "hipporag2_coverage",
    "language": "en"
  }'
```

4.3 `response_type` Values

Value	Description
`detailed_report`	Long-form structured answer with section headings (default).
`summary`	Short, executive-summary style answer.
`audit_trail`	Includes per-claim source attribution suitable for audits.
`comprehensive`	Maximally thorough; uses the full relevance budget.
`comprehensive_sentence`	Comprehensive coverage with sentence-level citation grouping.

4.4 `force_route` Values

Value	Route	When to use
`hipporag2_community`	R8 — community-bipartite	Single-document or topic-coherent questions (best general default).
`hipporag2_experimental`	R9 — monopartite α-tuned	Multi-hop reasoning where evidence is scattered across documents.
`hipporag2_coverage`	R10 — wide-coverage retrieval	Comparative or aggregative questions across many documents.

If omitted, the server picks a route automatically based on query characteristics.

4.5 Response Body


```
{
  "response": "string — the synthesized answer",
  "route_used": "hipporag2_community",
  "citations": [
    {
      "index": 1,
      "chunk_id": "s-0042",
      "document_id": "contract-001",
      "document_title": "Contract — Acme MSA",
      "document_url": "https://.../acme-msa.pdf",
      "score": 0.87,
      "text_preview": "Liability is capped at fees paid in the prior 12 months...",
      "page_number": 4,
      "section_path": "9. Limitation of Liability"
    }
  ],
  "evidence_path": ["entity-or-passage-id", "..."],
  "metadata": {
    "duration_ms": 1843,
    "model": "gpt-4.1-mini",
    "tokens_in": 4123,
    "tokens_out": 612,
    "k1": 35,
```

```
    "k2": 14
  }
}
```

Citation objects always include `index`, `chunk_id`, `document_id`, `document_title`, `document_url`, `score`, and `text_preview`. Location fields (`page_number`, `section_path`, `start_offset`, `end_offset`, `page_dimensions`) are included only when available for the source document.

If `include_context: true` was set, the full assembled evidence context is returned in `metadata.llm_context` (useful for debugging retrieval vs. synthesis).

---

5. Index Documents Endpoint

```
POST /hybrid/index/documents
GET /hybrid/index/status/{job_id}
```

Use this endpoint to submit documents to the caller-scoped corpus through the same APIM subscription key used for queries. Indexing is asynchronous: submit the documents, poll the returned `job_id` until `status` is `completed`, then query with `POST /hybrid/query`.

5.1 Request Body

Field	Type	Required	Default	Description
`documents`	array	Yes	—	Documents to index. Each item can be a text string, a URL string, or an object with `title`, `text`/`content`, `source`/`url`, and optional `metadata`.
`ingestion`	string	No	`document-intelligence`	Use `none` for direct text content, or `document-intelligence` for supported document URLs/files.
`reindex`	boolean	No	`false`	If `true`, clears existing indexed data for the caller's corpus before indexing the submitted documents.

Field	Type	Required	Default	Description
`run_community_detection`	boolean	No	`false`	Optional graph community precomputation; normally not needed for customer API ingestion.
`run_raptor`	boolean	No	`false`	Legacy RAPTOR summarization; normally leave disabled.
`folder_id`	string	No	*all*	Optional folder/collection partition to index into.

5.2 Example: Submit Text, Poll, Query

```
curl -X POST "https://graphrag-apim-wg3temevssbja.azure-api.net/graphrag/hybrid/index/documents
  -H "Content-Type: application/json" \
  -H "Ocp-Apim-Subscription-Key: <YOUR_KEY>" \
  -H "X-API-Version: v2" \
  -d '{
    "documents": [
      {
        "title": "Acme MSA",
        "text": "Payment is due net 30 days from invoice date. Liability is capped at fees paid
        "metadata": {"source_type": "api_text"}
      }
    ],
    "ingestion": "none",
    "reindex": true
  }'
```

Response:

```
{
  "status": "accepted",
  "group_id": "<your-group-id>",
  "job_id": "<job-id>",
  "documents_received": 1,
  "message": "Indexing job <job-id> started..."
}
```

Poll status:

```
curl "https://graphrag-apim-wg3temevssbja.azure-api.net/graphrag/hybrid/index/status/<job-id>"
  -H "Ocp-Apim-Subscription-Key: <YOUR_KEY>" \
  -H "X-API-Version: v2"
```

When the status is `completed`, run `POST /hybrid/query`.

5.3 Customer-Style Acceptance Test

Use the reusable E2E script to validate the full customer workflow through APIM: create a temporary API subscription, submit a fresh document, poll indexing, query the document on R8/R9/R10, assert citations/evidence, and clean up the temporary tenant data.

```
APIM_GATEWAY_URL="https://graphrag-apim-wg3temevssbja.azure-api.net/graphrag" \
AZURE_SUBSCRIPTION_ID="<azure-subscription-id>" \
AZURE_RESOURCE_GROUP="rg-graphrag-feature" \
APIM_SERVICE_NAME="graphrag-apim-wg3temevssbja" \
COSMOS_DB_ENDPOINT="<cosmos-endpoint>" \
COSMOS_DB_DATABASE_NAME="graphrag" \
python scripts/e2e_customer_api_flow.py --create-temp-subscription
```

Pass criteria:

1. `POST /hybrid/index/documents` returns `status=accepted` and a `job_id`.
2. `GET /hybrid/index/status/{job_id}` reaches `status=completed`.
3. `POST /hybrid/query` returns HTTP 200 for R8 (`hipporag2_community`), R9 (`hipporag2_experimental`), and R10 (`hipporag2_coverage`).
4. Each route returns a non-empty `response`, at least one citation, and a non-empty `evidence_path`.
5. Each route answer contains the known facts from the submitted test document.

---

6. Cache Prewarm Endpoint

```
POST /hybrid/cache/prewarm
GET /hybrid/cache/prewarm
```

Large corpora may need a one-time cold load of the graph and embeddings before the first Route 8/9 query. Start this load with `POST /hybrid/cache/prewarm`; the endpoint returns immediately while the backend warms the process-local cache. Poll `GET /hybrid/cache/prewarm` until `status` is `ready`, then call `POST /hybrid/query`.

```
curl -X POST "https://graphrag-apim-wg3temevssbja.azure-api.net/graphrag/hybrid/cache/prewarm"
  -H "Content-Type: application/json" \
  -H "Ocp-Apim-Subscription-Key: <YOUR_KEY>" \
  -H "X-API-Version: v2" \
  -d '{"force_route": "hipporag2_experimental"}'
```

Poll status:

```
curl "https://graphrag-apim-wg3temevssbja.azure-api.net/graphrag/hybrid/cache/prewarm?force_rou
  -H "Ocp-Apim-Subscription-Key: <YOUR_KEY>" \
  -H "X-API-Version: v2"
```

Response:

```
{
  "status": "running",
  "group_id": "<your-group-id>",
  "partition_group_id": "<resolved-folder-or-group-id>",
  "route": "hipporag2_experimental",
  "cache_key": "<partition>:general_enterprise",
  "started_at": 1780900000.0,
  "completed_at": null,
```

```
  "error": null
}
```

`status` is one of `started`, `running`, `ready`, or `failed`. The cache is process-local and may be rebuilt after deployment, restart, scale-out, or cache flush. For normal small/medium corpora, prewarm is optional; use it for large Route 8/9 corpora to avoid APIM gateway timeouts on the first query.

---

7. Health Endpoint

`GET /hybrid/health`

Lightweight liveness/readiness probe. Returns service status and basic backend reachability.

```
curl "https://graphrag-apim-wg3temevssbja.azure-api.net/graphrag/hybrid/health" \
  -H "Ocp-Apim-Subscription-Key: <YOUR_KEY>" \
  -H "X-API-Version: v2"
```

Response:

```
{
  "status": "healthy",
  "components": {
    "router": "ok",
    "synthesizer": "ok",
    "profile": "general_enterprise"
  },
  "profile": "general_enterprise",
  "group_id": "<your-group-id>"
}
```

> `status` is `healthy` when all components report `ok`/`not_configured`/`fallback_mode`/`no_llm`, `degraded` otherwise, and `unhealthy` if the readiness check itself fails.
>
> An anonymous `GET /graphrag/health` (gateway-only liveness) is also exposed for monitoring without a subscription key.

---

8. Profiles Endpoint

`GET /hybrid/profiles`

Returns the retrieval routes available to `POST /hybrid/query`, plus the server-side defaults. Each `id` is a valid value for the `force_route` request field. When `force_route` is omitted, the server auto-selects a route based on the query.

```
curl "https://graphrag-apim-wg3temevssbja.azure-api.net/graphrag/hybrid/profiles" \
  -H "Ocp-Apim-Subscription-Key: <YOUR_KEY>" \
  -H "X-API-Version: v2"
```

Response shape:

```
{
  "routes": [
    {
      "id": "hipporag2_community",
      "route": "R8",
```

```
      "name": "Community (bipartite)",
      "description": "Entity-focused retrieval with Personalized PageRank plus community contex
      "best_for": ["Single-document questions", "Topic-coherent questions", "Known-entity looku
    },
    {
      "id": "hipporag2_experimental",
      "route": "R9",
      "name": "Experimental (monopartite, α-tuned)",
      "description": "Multi-hop PPR graph walk for reasoning where evidence is scattered across
      "best_for": ["Multi-hop reasoning", "Cross-document evidence", "Relationship mapping"]
    },
    {
      "id": "hipporag2_coverage",
      "route": "R10",
      "name": "Coverage (wide retrieval)",
      "description": "Exhaustive map-reduce retrieval for comparative or aggregative questions
      "best_for": ["Comparative questions", "Aggregation across documents", "Coverage and negat
    }
  ],
  "default_route": "auto",
  "default_synthesis_model": "gpt-4.1-mini",
  "response_types": ["detailed_report", "summary", "audit_trail", "comprehensive", "comprehensi
}
```


---

9. Rate Limits & Quotas

Two layers of throttling apply:

9.1 Gateway-Level Throttling (APIM)

The APIM Consumption tier enforces per-subscription throttling. When exceeded, the gateway returns `429 Too Many Requests` with a `Retry-After` header before the request reaches the backend.

> Each customer authenticates with their own per-tenant APIM subscription, scoped to the pay-as-you-go product (`graphrag-api_payg`). The backend resolves entitlement from the subscription and draws down a **prepaid credit wallet** per request (see §9.2).

9.2 Prepaid Credit Wallet (usage-based billing)

API access is **prepaid and usage-based**: the customer buys credit packs up front and every metered call draws the balance down based on **real** usage. There is no monthly subscription, no fixed query/page allowance, and credits do not expire.

One billing rule. Every metered operation deducts `ceil(real_COGS_credits × 2.5)` from the wallet — a flat 2.5× markup on the underlying cost of goods (~60% gross margin) applied automatically on every route, so simple lookups cost little and multi-hop queries cost proportionally more.

- **1 credit = $0.001** of face value (so **$1 = 1,000 credits**).
- **Query cost** = the orchestrator's measured token usage (`usage.credits_used`) × 2.5. Auto-scales with route complexity (R8/R9/R10) — you are not billed a flat per-query rate.

- **Index cost** = `pages × 24 credits` (raw COGS) × 2.5, where the API ingests text/URLs (no PDF pages), so **1 page = 3,000 characters** of ingested text (≈ 800 tokens). Charged once per indexing job at completion, metered from the **actual indexed content** — so a document supplied as a URL is billed on the text fetched during indexing, not on the (empty) request body.

Credit packs (one-time Stripe Checkout, `mode="payment"`):

- $1 → 1,000 credits (starter / entry) — ≈ 26 queries or 16 pages
- $10 → 10,000 credits — ≈ 263 queries or 166 pages
- $25 → 25,000 credits — ≈ 657 queries or 416 pages
- $50 → 50,000 credits — ≈ 1,315 queries or 833 pages
- $100 → 110,000 credits (+10% bonus) — ≈ 2,894 queries or 1,833 pages

What credits buy: a typical query bills ~38 credits (real usage × 2.5, so simple queries cost less) and one indexed page bills 60 credits (exact). The `GET /billing/api/credits/packs` response carries `credits_per_query`, `credits_per_page`, and per-pack `approx_queries` / `approx_pages` so the dashboard can show these. Query counts are estimates; page counts are exact.

Entry is **paid-only** — the $1 starter pack is the entry point; there are no free credits. Buy packs from the dashboard **API Access** panel (`POST /billing/api/credits/checkout`, catalog at `GET /billing/api/credits/packs`); the grant is applied on the Stripe webhook and is idempotent on the Checkout session id. Buying any pack provisions the customer's `graphrag-api_payg` APIM subscription if they don't already have one.

A user must have a provisioned APIM subscription (i.e. have purchased at least one credit pack) to authenticate via the API. App-tier subscriptions alone do **not** grant API access.

Service principals inherit the API entitlement of the admin who created them: an SP is rejected at the gateway if the creating admin has no API wallet.

Empty balance → `402 Payment Required`. When the wallet is exhausted the request is blocked before work begins:

```
{
  "error": "insufficient_credits",
  "message": "Your prepaid API credit balance is exhausted. Purchase more credits to contir
  "credits_remaining": 0,
  "purchase_url": "/dashboard#api"
}
```

Successful responses carry the live balance in the `X-Credits-Remaining` header; query responses also include `metadata.credits_charged` and `metadata.credits_remaining`.

Legacy note: subscription-style API tiers (Trial / Pro / Pro Plus, products `graphrag-api_trial` … `graphrag-api_pro_plus`) are retired for new customers. Existing records that still carry one of those `api_tier` values keep their monthly pools for back-compat, but all new purchases are pay-as-you-go credit packs.

APIM subscription-key rotation: `POST /billing/api-subscription/rotate-key` (web-session auth only) regenerates the caller's APIM **primary** key and emails the new key to the account address. This is an **emergency rotate** — the previous key stops working **immediately** (use it if a key is leaked). Returns `200` with `{"status":"delivered"}` once emailed, or `202` with `{"status":"pending_retry"}` if the key was regenerated but the email could not be sent yet (the reconciliation sweep retries delivery of the current key). A short server-side cooldown (`APIM_KEY_ROTATION_COOLDOWN_SECONDS`, default 60s) throttles repeated rotations.

Cancelling or downgrading off the API add-on **deletes** the APIM subscription, invalidating all of its keys. The two-key (primary/secondary) model leaves room for a future zero-downtime rotation flow.

Legacy `gro_*` key rotation: `POST /api-keys/{key_id}/rotate` issues a new key and marks the old one for expiry after a **5-minute grace window** (`ROTATION_GRACE_SECONDS`). Both keys validate during the grace window so deployments can roll forward without an outage. This applies only to the legacy direct-backend `gro_*` keys, which are no longer publicly supported (see §1) — it is independent of the APIM subscription-key flow above.

Backend quota responses include these headers on success:

Header	Description
`X-RateLimit-Limit-Daily`	Daily query limit
`X-RateLimit-Remaining-Daily`	Remaining daily queries
`X-RateLimit-Limit-Monthly`	Monthly query limit
`X-RateLimit-Remaining-Monthly`	Remaining monthly queries

These `X-RateLimit-*` headers and the `429` body below apply to **app-tier** traffic and legacy subscription API tiers. **Pay-as-you-go API callers** are not rate-limited by daily/monthly pools — they receive `X-Credits-Remaining` on success and `402 insufficient_credits` (see §9.2) when the wallet is empty.

Backend quota-exceeded body (`429`):

```
{
  "error": "quota_exceeded",
  "message": "Daily query limit exceeded",
  "plan": "pro_plus",
  "daily_used": 10000,
  "daily_limit": 10000,
  "monthly_used": 8542,
  "monthly_limit": 20000,
  "retry_after_seconds": 3600
}
```

---

10. Error Reference

10.1 Standard Error Shape

The gateway and backend both return JSON errors. Backend errors use `detail`; gateway errors use `message`/`statusCode`:

```
{ "detail": "Human-readable error description" }
```

```
{ "statusCode": 401, "message": "Access denied due to missing subscription key." }
```

10.2 HTTP Status Codes

Status	Meaning	Common cause
`200`	Success	—

Status	Meaning	Common cause
400	Bad Request	Malformed JSON, missing required field (query), invalid force_route value
401	Unauthorized	Missing or invalid Ocp-Apim-Subscription-Key; missing X-API-Version
403	Forbidden	Subscription disabled, or group lacks access to the requested folder
404	Not Found	Unknown operation path (typo in endpoint)
402	Payment Required	Prepaid credit wallet exhausted — buy more credits (see §9.2)
429	Too Many Requests	Gateway throttle or backend quota exceeded (see §9)
500	Internal Server Error	Backend exception during synthesis
503	Service Unavailable	Upstream dependency (vector store, LLM provider) failed

10.3 Common Auth Errors

Missing subscription key:

```
HTTP/1.1 401 Unauthorized
{ "statusCode": 401, "message": "Access denied due to missing subscription key. Make sure to include
```

Invalid subscription key:

```
HTTP/1.1 401 Unauthorized
{ "statusCode": 401, "message": "Access denied due to invalid subscription key." }
```

Missing version header:

```
HTTP/1.1 404 Not Found
{ "statusCode": 404, "message": "Resource not found" }
```

> Always include `X-API-Version: v2`. Without it, APIM cannot resolve the operation.

Appendix A — Python SDK Example

```python
import os
import requests

GATEWAY = "https://graphrag-apim-wg3temevssbja.azure-api.net/graphrag"
SUB_KEY = os.environ["GRAPHRAG_SUBSCRIPTION_KEY"]

HEADERS = {
    "Content-Type": "application/json",
    "Ocp-Apim-Subscription-Key": SUB_KEY,
    "X-API-Version": "v2",
```

```
}


def query(question: str, *, route: str | None = None, language: str | None = None) -> dict:
    """Query the GraphRAG knowledge base via the APIM gateway."""
    body: dict = {"query": question}
    if route:
        body["force_route"] = route
    if language:
        body["language"] = language

    resp = requests.post(f"{GATEWAY}/hybrid/query", json=body, headers=HEADERS, timeout=120)
    resp.raise_for_status()
    return resp.json()


# Simple query
result = query("What is the contract termination notice period?")
print(result["response"])

# Force a specific route
result = query("Compare all liability caps across vendors", route="hipporag2_coverage")

# Reply in Japanese
result = query("What are the payment terms?", language="ja")
```

Appendix B — cURL Cheat Sheet

```
GATEWAY="https://graphrag-apim-wg3temevssbja.azure-api.net/graphrag"
KEY="<YOUR_SUBSCRIPTION_KEY>"

# Query
curl -X POST "$GATEWAY/hybrid/query" \
  -H "Content-Type: application/json" \
  -H "Ocp-Apim-Subscription-Key: $KEY" \
  -H "X-API-Version: v2" \
  -d '{"query": "What are the payment terms?"}'

# Health
curl "$GATEWAY/hybrid/health" \
  -H "Ocp-Apim-Subscription-Key: $KEY" \
  -H "X-API-Version: v2"

# Profiles
curl "$GATEWAY/hybrid/profiles" \
  -H "Ocp-Apim-Subscription-Key: $KEY" \
  -H "X-API-Version: v2"
```