

AIGW · MULTI-TENANT AI + API GATEWAY

Govern the content. Trace the caller. Account for every call.



AIGW multi-tenant AI + API gateway: one control plane and two data planes, governing both AI calls and general HTTP APIs.

AIGW Multi-tenant AI + API Gateway

Every AI call and every API request can answer three questions: who is calling, why it was allowed, and whose account the usage belongs to. The general API data plane is disabled by default and enabled as needed; when it stays off, behavior is identical to v1.9.x, with zero changes to the AI path.

Chongqing HongChuangDa Technology Co., Ltd.

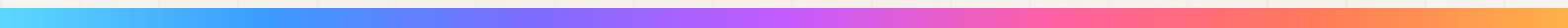
Verified attribution

Content moderation

AI + API governance

Call-level reconciliation

Private deployment



When these show up, your calls need a governed entry layer

When several units and applications share AI, or existing HTTP interfaces must be opened to others, content, identity, accounting and rules become problems at the same time.

01 · Content risk needs a checkpoint External: every answer from a service bot, mini-program or public Q&A speaks for your organization, so risk must be stopped before it leaves the gateway. Internal: inappropriate remarks and off-purpose use must be observable, traceable and evidenced.

02 · Many departments share one Key The model side sees a single exit: no one can tell who is using it, let alone trace a person.

03 · Own compute pool, multiple upstreams Capacity planning, quotas and contribution need one usage basis; upstreams with different capacity and rate limits need one entry point and continuity mechanisms.

04 · Monthly reconciliation and audit evidence Usage must be reviewable, lockable and auditable, and past figures must not be rewritten after reorganizations.

05 · HTTP interfaces to open up Give sister organizations, subsidiaries or partners one entry point and approvable subscriptions, instead of every system issuing its own Keys and rate limits.

Chapter guide	Pages
Opening · Mixed, blind, disputed; the approach and the difference	P03–P07
Ch. 1 · Attribution & metering: three ledgers, six dimensions, frozen snapshots	P08–P12
Ch. 2 · Trusted identity: eight identity questions, L1/L2/L3 contract	P13–P16
Ch. 3 · Content safety & compliance: risk surfaces, pipeline and modes, evidence	P17–P19
Ch. 4 · Routing & continuity: multiple upstreams, circuit breaking, queueing	P20–P23
Ch. 5 · Governance & operations: audit and locking, three-level policy, deployment	P24–P26
Ch. 6 · General API gateway: dual data planes, immutable versions, subscriptions	P27–P30
Scope and contact	P31–P32

Product screenshots show the Chinese-language interface; all data in them is demonstration data.

Govern the content

Trace the caller

Account for every call

Keep business running

One governance, two kinds of traffic

One Key hides every caller behind it

Several departments and many employees share one AI application, yet the model side sees only one backend exit and one API Key.

People are mixed together

The model service does not know who initiated a call; it sees the application identity, never the employee.

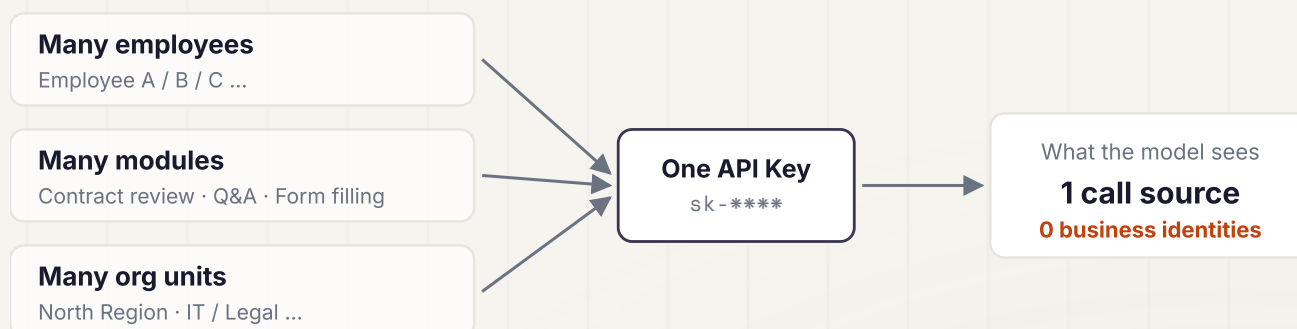
Modules are mixed together

Contract review, knowledge Q&A and smart form-filling share one exit, so usage can only be summed, not traced back to a business action.

Organizations are mixed together

One platform serves several regions and departments; once calls reach the model side they lose their organizational owner, and accounts lack granularity from the source.

MANY CALLERS · ONE KEY



People, modules and org units are flattened at the exit: the records carry no business identity.

The question is not “are there call records?” but “do the records carry business identity?”

Without identity, problematic content cannot be traced to a person, and growing usage cannot be explained.

One Key

Shared by many

Mixed modules

Missing ownership

You see the GPUs are busy, not whom they are busy for

“Scheduling” solves different problems at different layers. However complete resource-layer metrics are, they do not answer who the usage belongs to.

FIVE LAYERS · WHO ANSWERS WHAT

▲ Enterprise governance	Tenants, departments, apps, Keys, accounts	AIGW
Inference requests	Which model replica serves each request	Inference router
Model serving	Model instances, replicas and scaling	Model serving platform
Job admission	Training jobs, team GPU quotas	Job queue
GPU devices	GPUs, memory, partitioning, allocation	Container orchestrator

The higher the layer, the closer to the business: only the top layer answers whose account the usage belongs to.

Each layer handles one thing

The GPU device layer decides how memory is partitioned, the job admission layer whether a job gets scheduled, the model serving layer how instances scale, and the inference request layer which replica serves a request. None of them answers “whose account does this consumption go to?”, and none of them looks at content.

The enterprise governance question

Which region and department is using it; which employee and application initiated it; which module and model produced the usage; and whether this request and its answer are compliant.

The boundary must be clear

The gateway keeps application-layer accounts and governs application-layer content; the scheduling platform manages resource-layer GPUs. Each has its own job; together they give a complete view from business call to compute resource.

Resource view

Business view

Application-layer accounts

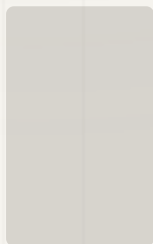
Resource-layer GPUs

Shared compute raises three questions: expansion, quotas, contribution

Co-building partners care less about charging than about usage that can be explained. When accounts lack granularity, management decisions lack a shared basis.

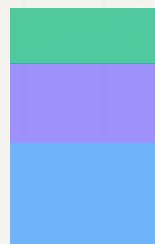
SAME TOTAL · DIFFERENT LEDGERS

Total only

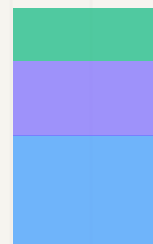


Monthly total
up 38%

The same total, split by different dimensions



By region



By app & module

Expansion, quotas and contribution all need the split ledgers on the right, not just the total.

Who drives expansion

When resources stay tight, managers need to know which region, department, application and module the growth comes from, not just that the total is rising.

Who gets quota

When several applications compete for one compute pool, quotas should rest on verifiable historical usage and business priority.

How contribution is counted

Co-building investment and actual use need one metering basis: converted by model differences, summarized by organization and application.

Compute can be shared; the accounts must be clear.

Without accounts there is no fairness. First give every call an owner, then agree on the co-building rules.

Shared-compute allocation

Expansion basis

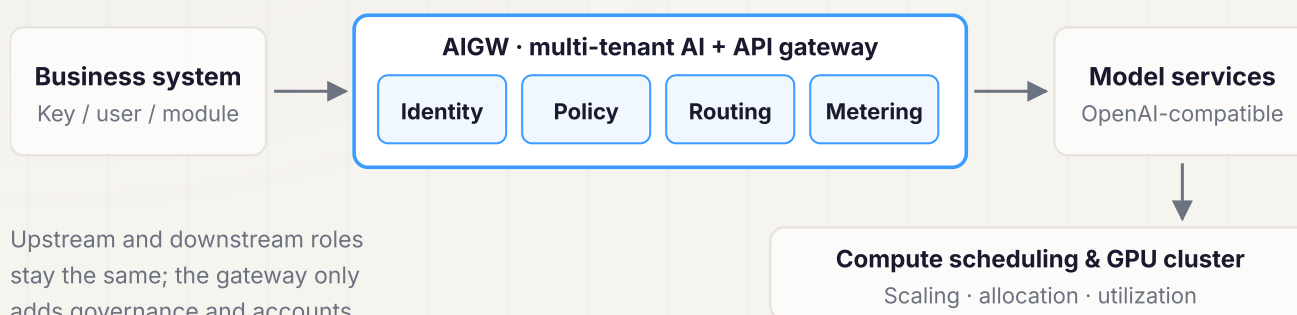
Quota basis

Contribution accounting

One control plane, two data planes, governance in a single layer

The AI data plane carries model calls and the general API data plane carries ordinary HTTP services; both share identity, RBAC, auditing and cluster operations.

ONE GATEWAY · FOUR JOBS



One call becomes one chain of accountability



Two data planes, one kind of traffic each

On the AI data plane, identity, policy, content, routing and Token metering form a chain of accountability; on the general API data plane, the service catalog, subscription admission, effective policies and byte-and-latency access records form a governance chain. The two ledgers live in separate tables with separate bases.

Upgrade boundaries, stated up front

The general API data plane is disabled by default and enabled as needed; when off, behavior is identical to v1.9.x. The AI data plane was unchanged across four upgrade batches, each of which re-ran the /v1 golden regression cases. The major version moved to 2.0 because the new general HTTP proxy entry changes the deployment topology, not because of a rewrite.

Upstream and downstream roles stay the same

Model services accept requests through compatible endpoints; compute scheduling platforms keep handling instance scaling and utilization.

The result: every AI call and API request can be traced from “who is using it” to “why it was allowed” to “how it was recorded”.

One control plane

Two data planes

Identity chain

Content moderation

Access records

Identity, content, usage and API governance on one chain

The point is not one more feature: AI calls and ordinary APIs share identity, subscriptions, policies and auditing, instead of each keeping its own governance chain.

FROM METRIC TO LEDGER

Usage as a metric

- Trends and totals
- Basis shifts with current config
- Rewritten after reorganizations
- For monitoring and budgeting



Usage as accounts

Attributable

Org, app, module, user, model in one record

Frozen

Org snapshot fixed at recording time

Reconcilable

Each line re-checkable; groups sum to total

Lockable

Locked periods never recomputed; audited

Why not just “a better dashboard”

Metrics can be recomputed at any time; accounts must be reviewable and lockable. Filtering only answers “should this sentence go out?”; accountability must also answer “who said it, in which module of which application?”. The data requirements differ fundamentally: accounts and evidence both need ownership fixed at the moment of recording, never rewritten by later configuration changes.

What the combination means

The enterprise org tree, call-level multi-dimensional attribution, frozen organizational snapshots, monthly ledger locking, content hit records, capacity awareness and queueing, plus API asset versions, plan subscriptions, five-level policies and access records, together form one chain of accountability from person to content and from admission to accounts.

Enabling it does not change the existing AI path

The general API data plane is disabled by default and enabled as needed; when off, behavior is identical to v1.9.x, with zero changes to the AI path.

Attributable

Accountable

Reconcilable

One governance, two kinds of traffic

Turn mixed-up calls into accounts you can explain

Metering is more than counting calls. AIGW puts identity, organization, application, module, model and tokens into the same usage record.

This chapter answers four questions

- How system, user and department ledgers are built up layer by layer
- How six business dimensions combine in queries
- How different models convert to a common metering unit
- How historical accounts stay intact after people and organizations change

From totals to ownership, from ownership to verification.



The dashboard shows the situation, not the accounts: read totals and trends here, then break them down by dimension in reports.

① Compute units weighted by model tier ② Distinct end users ③ Calls and TOKENS over the last 7 days, dual axis (demo data)

Three ledgers

Six-dimension reports

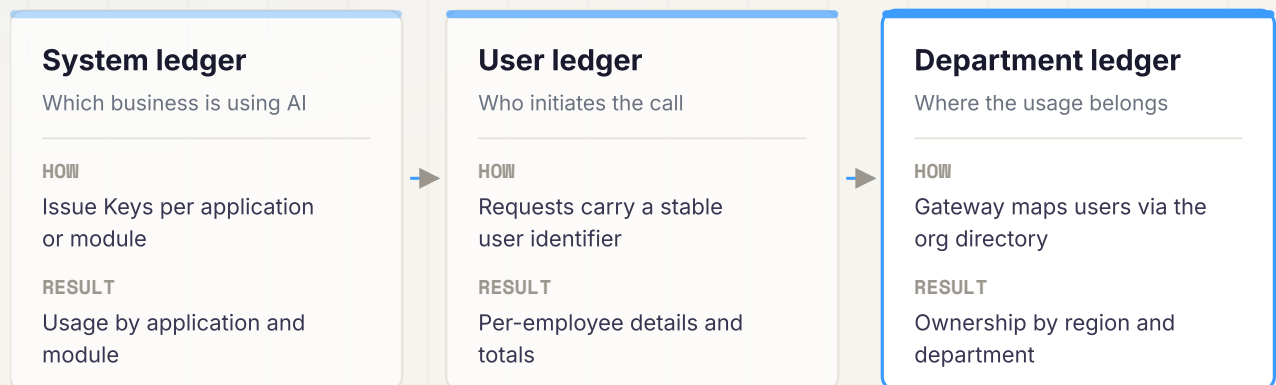
Compute units

Frozen snapshots

System ledger first, then user and department ledgers

Different integration conditions give different attribution granularity. AIGW makes each step pay off on its own and lets the ledgers keep getting finer.

THREE LEDGERS · STEP BY STEP



Each step pays off on its own: step one already yields application and module accounts.

The first ledger needs no code changes

Issue Keys per application or module and you have application and module accounts the same day; not one line of business calling code changes.

The second ledger needs the right identifier

The user identifier must be a stable employee ID or an immutable user ID. Do not send names or department names: names collide and departments change, so neither can anchor an account.

The third ledger rests on a trusted org mapping

Department accounts hold up only once the org mapping matches HR records. The mapping is maintained in one place, the gateway, so systems do not each keep their own copy of the organization.

One set of usage, regrouped around management questions

Region, department, application (business system), module, model and time period combine freely, with drill-down from totals to call details.

Six dimensions

- Region: how much each region used
- Department: how usage is spread across the organization
- Application: which business system generates the calls
- Module: usage by specific business scenario
- Model: requested vs. executed models
- Time period: usage by hour, day and accounting period

Typical management questions

- Department × model: which departments use premium-tier models, and does that fit their role?
- User × module: who calls heavily, in what scenario, and does it match their job?

分组维度

区域

部门

系统

模块

模型

时段

日期

时间范围

近 7 日

应用

全部应用

导出 CSV

查询

区域	系统	部门	模块	模型	时段	调用数	失败	TOKEN	算力
华南区域	智能文档审核系统	华南区域	evidence-audit	qwen3-32b	2026-08-16T18:00:00Z	8	0	20,144	2
华东区域	智能会议纪要系统	运营指挥中心	incident-summary	qwen3-32b	2026-08-16T18:00:00Z	8	0	19,920	2
华南区域	智能会议纪要系统	华南区域	transcription	qwen3-32b	2026-08-16T18:00:00Z	8	0	19,696	2
华东区域	智能文档审核系统	运营指挥中心	evidence-audit	qwen3-32b	2026-08-16T18:00:00Z	8	0	19,472	2
华南区域	智能会议纪要系统	华南区域	incident-summary	qwen3-32b	2026-08-16T18:00:00Z	8	0	18,848	2
华东区域	智能会议纪要系统	运营指挥中心	transcription	qwen3-32b	2026-08-16T18:00:00Z	8	0	18,224	2

Tick different dimensions and the same usage becomes a different account; every group drills down to call details.

① Tick any of the six grouping dimensions ② Ticked dimensions become group columns ③ Calls, failures and TOKENS per group (the table continues right with compute units and actions)

Free combination

Drill-down

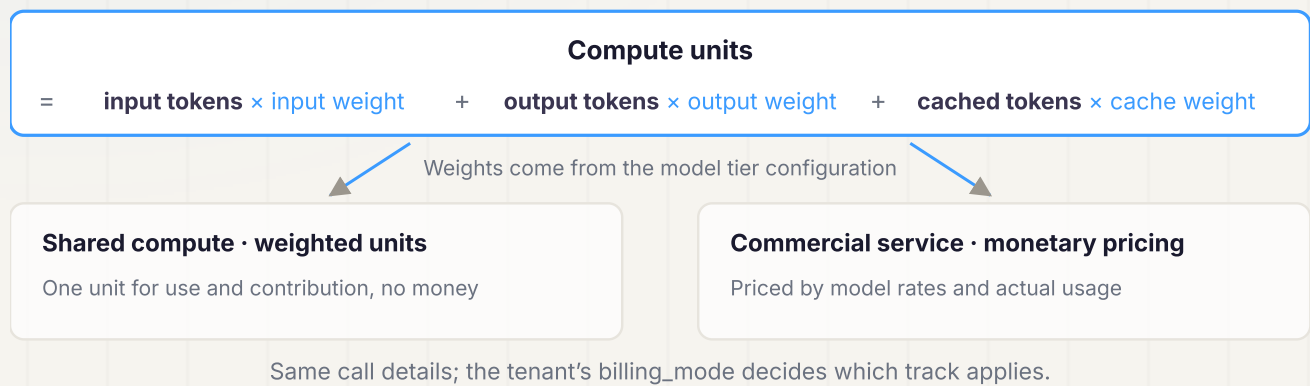
Hourly rollups

Monthly locking

Not just counting calls: converting by tokens and model tier

A long generation and a short Q&A should not count as the same usage, and the resource cost of different models needs a common basis.

WEIGHTED UNIT · ONE FORMULA



Every conversion can be checked line by line

Weights come from the model tier configuration, set by the participants under common rules. Each detail record keeps input, output and cached tokens plus the requested and executed model, so anyone can recompute any line. At monthly locking the weight version is kept too, so later reviews use the weights of that time, not today's.

Why shared-compute scenarios avoid money

Co-building partners need to settle "who used how much, and who should bear how much". Converting to money means agreeing on prices first, which makes the accounts harder to agree on. Weighted compute units answer only the usage question; where pricing is genuinely needed, the tenant switches `billing_mode` to monetary pricing without re-collecting call details.

People move, units restructure, and past accounts stay put

Reports must not rewrite yesterday's ownership with today's org chart. When a call is recorded, AIGW freezes the organizational path of that moment.

At recording time

The user identifier is resolved against the org directory; region, department, org path, application name and usage are written together into the detail snapshot.

After a reorganization

Transfers, department renames or hierarchy changes affect only later calls; existing periods still read the snapshot taken at recording.

Management outcome

- Last month's confirmed department accounts do not follow later changes
- New calls are attributed under the new structure
- After monthly locking, backfills and recalculations do not change locked periods

An organizational snapshot is not a display field; it is what makes accounts and evidence credible. A hit record from six months ago still shows the department of that time.

部门 ?	调用数 ?	失败 ?	TOKEN ?	算力单元 ?	均延迟 ?	操作
华南区域	24	0	58,688	79,568	1,088 ms	明细
运营指挥中心	24	0	57,616	77,896	1,071 ms	明细
财务与资产部 ①	110	5	2,033	3,639 ②	1,544 ms	明细
法务部	40	2	760	1,428	26 ms	明细
技术部	38	0	760	1,428	27 ms	明细
表格分页展示，合计口径以上方汇总卡为准（全量统计）						
计量口径：算力单元 · 权重版本 49 · 数据截止 2026/8/17 12:35:01 ③ 上一页 下一页						

After a department rename, its figures for the same period match those before the rename: reports read the org snapshot taken at recording.

① New department name after the rename ② Compute units for the same period unchanged ③ Metering basis, weight version and data cut-off shown with the report

Frozen at recording

History not rewritten

Old and new kept apart

Auditable locking

Not self-declared, but signed and verified

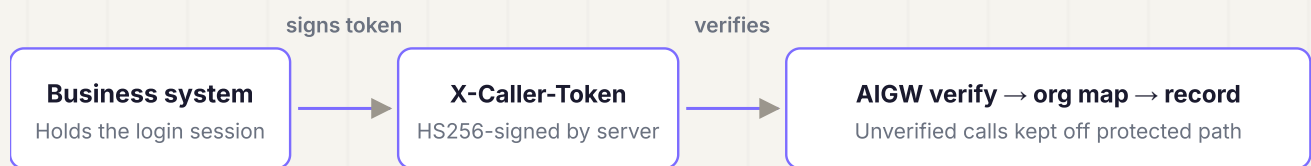
The gateway sees each request but cannot know on its own whose behalf a call is made on. Trusted identity must be brought in by the business system that holds the login session, and verified by the gateway.

Principles of the identity chain

- The business system knows the logged-in employee and module
- The caller puts a stable identifier into the AI request
- Where trusted attribution is needed, the server issues a caller identity token (X-Caller-Token, the L3 token)
- The gateway verifies the signature, then applies org mapping, policy checks, content moderation and recording

Trusted identity is not one more field; it makes “who is calling” a verifiable input.

TRUSTED IDENTITY FIRST



With identity and organization established, the other four links have a basis



L1 / L2 are unsigned inputs for statistical attribution; trusted attribution relies on L3 tokens.
The gateway does not replace an enterprise IdP; it only verifies on whose behalf a call is made.



One call, eight questions, all answered on the same chain

From user and organization to result and trace, AIGW unifies identity, authorization, resources and metering in the call context.

Identity-chain question	What you can manage
WHO · Who is calling	The employee ID or stable ID that initiated the call
ORG · Where they belong	Region and department; the org path is frozen at recording
AGENT · Through what	The application, Agent and module
WHY · Why it was allowed	Policy-based admission, with the authorization result recorded
WHAT · What was used	Requested model, executed model and upstream service
HOW MUCH · How much	Input, output and cached tokens, as compute units or amounts
RESULT · Outcome	Success or failure status and error code
TRACE · How to trace it	A request ID links the call, metering, hit records and audit trail

Together the eight answer: who, in which organization, through which application, under which rules, used which resources, and how the result can be verified. Content hit records hang on the same chain, so “who said this” has an answer too.

时间 ?	1 请求 ID ?	应用 ?	用户标识 ?	模块 ? 3
2026/8/17 02:41:33	b6dbb2d7f357b94f3984d d238f8f96bf	智能会议纪要系统	zhangsan 2	transcript
2026/8/17	f9399af387cc7933545d2			

The eight questions are not eight reports; they are eight groups of fields in one call record.

① Request ID links call, metering and audit (TRACE) ② A specific employee identifier (WHO) ③ Module identifier (AGENT); further right the same record holds executed model, status code and usage

WHO

ORG

AGENT

RESULT

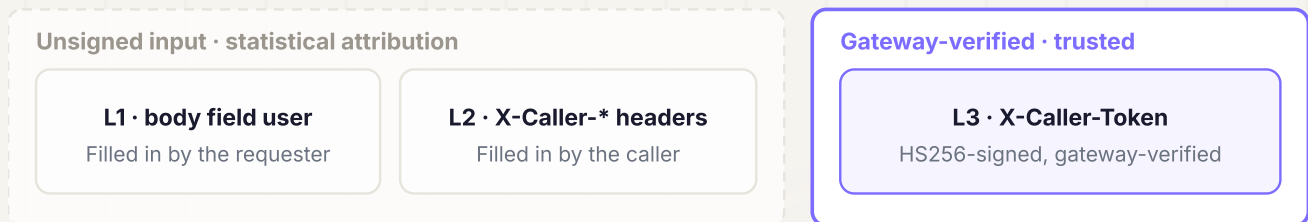
TRACE

Choose a level by integration capability, set the boundary by trust needs

Identity can be onboarded step by step, but statistical attribution and trusted attribution must be kept apart. The more capable the platform, the finer the chain and the stronger the verification.

Level	Identity mechanism	Attribution	Trust boundary
L1	OpenAI-standard request body user field	User level	Set by the requester; not signature-verified
L2	X-Caller-User / X-Caller-Module headers	User + Agent / module	Set by the caller; not signature-verified
L3	X-Caller-Token, HS256-signed JWT	User + org + Agent / module + metering	Gateway-verified; fit for trusted attribution

PRECEDENCE & TRUST BOUNDARY



User identifier precedence: L3 token → header → request body

When several identities are present

The user identifier is taken from the L3 token, then the headers, then the request body; the Agent identifier prefers a verified L3 token.

When trusted attribution is required

Administrators can require a valid L3 identity by policy; requests that fail verification do not enter the protected model call path. For accountability-sensitive cases, such as public-facing apps, requiring L3 outright is recommended.

Capability boundary Unified identity onboarding, authorization, call attribution and auditing, connected to enterprise directories (LDAP / SCIM); it does not replace an enterprise identity provider.

- L1 compatible
- L2 extended
- L3 verified
- Tiered onboarding

End-user identity does not reach the model gateway by itself

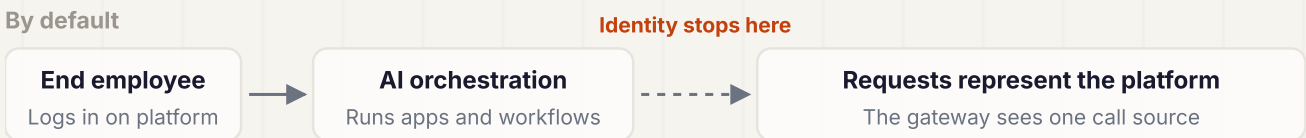
AI orchestration platforms run workflows and applications, but their downstream model requests usually represent the platform itself, not the individual employee.

Tested platform	Tested version	Technical fact
Dify	1.16	Does not pass through end-user identity
FastGPT	v4.15.7 + aiproxy v0.6.5	Does not pass through end-user identity
Coze	Open-source edition (tested 2026-08)	Does not pass through end-user identity

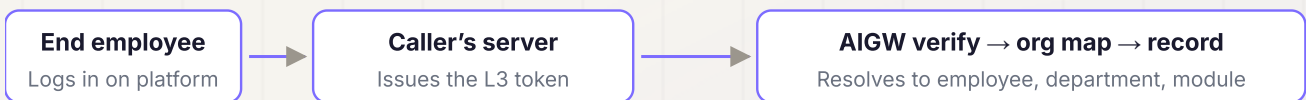
All three findings were tested in 2026-08 on Dify 1.16, FastGPT v4.15.7 + aiproxy v0.6.5 and the Coze open-source edition. They state only the current state of identity pass-through and do not evaluate the products.

WHERE IDENTITY STOPS

By default



With identity carried in



What this means

To build trusted usage accounts by employee and department, or to trace a content hit to the person responsible, end-user identity must be brought in explicitly by the caller that holds the login session, then verified, mapped and recorded by the AI gateway.

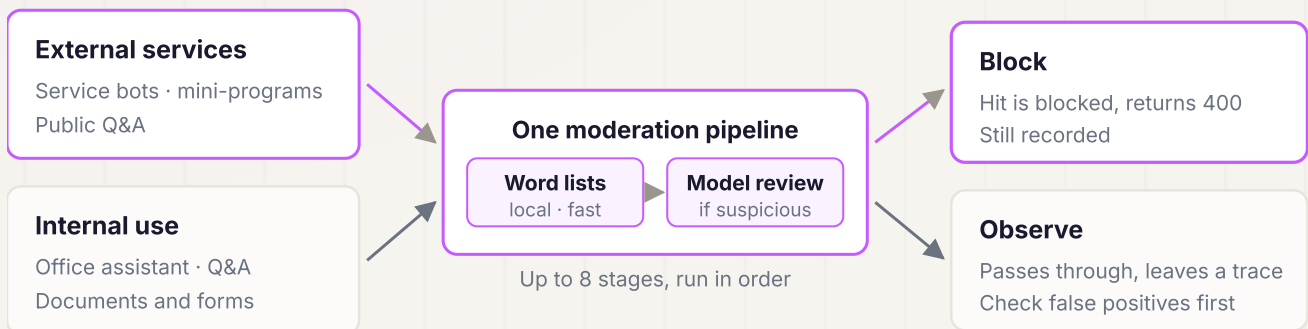
Integration does not change platform roles.

The orchestration platform keeps managing applications and workflows; AIGW handles identity attribution, content moderation, policy admission, model routing and usage auditing at the downstream exit.

If you don't know who is speaking, you can't govern what is said

The compute is private, but the people served may be outside. Content risk has two surfaces, and they are handled differently.

TWO SURFACES · ONE PIPELINE



In one gateway, external apps can use Block and internal apps Observe, issued per application.

External services: every output speaks for you

For service bots, WeChat mini-programs and public Q&A, every answer goes out in the organization's name. Inappropriate content, once sent, becomes a public-opinion and compliance incident, so it must be stopped before it leaves the gateway, not deleted afterwards.

Internal use: accountability without prying

Inappropriate remarks, disparaging content and off-purpose AI use must be traceable to a person, leave evidence and support follow-up. AIGW's trade-off: hit records carry full attribution, but **under the current retention policy only a masked summary and a content-digest hash are kept, not the original text**; even viewing hit details is audited.

This chapter answers three questions

- How the pipeline is arranged, and when to use Off, Observe or Block
- How word lists are self-maintained and how hits are graded
- What is kept after a hit, and where accountability stops

Word lists first, then model review; observe first, then block

Moderation must stop what it should without slowing business or misfiring on day one, so it is split into a multi-stage pipeline and three operating modes.

Three operating modes

One master switch: **Off** (no moderation), **Observe** (asynchronous review that records hits without affecting requests), **Block** (runs pipeline stages and hit actions). Run new rules in Observe first, confirm the false-positive rate, then switch to Block.

How the pipeline is arranged

Up to 8 stages run top to bottom, combining keyword and model stages: cheap local word-list screening first, then suspicious content goes to model review. Each stage sets its own hit action (observe, pass to the next stage for review, block, or block and short-circuit); keyword stages can set a minimum hit score so one weak match does not block.

Self-maintained word lists

Terms are kept in separate lists by category, with search, import/export and bulk paste (delimiter and conflict strategy configurable); one list can hold tens of thousands of entries, so existing lists import directly. Settings are issued per application: Block for external apps, Observe for internal ones, side by side in one gateway.

The screenshot displays the configuration interface for content moderation. At the top, there are three radio buttons for operating modes: 1. 关闭 (Off): 不做任何审核，所有请求直接进入后续处理。 2. 观察 (Observe): 异步审核并记录命中，不影响业务请求。 3. 拦截 (Block): 按流水线阶段和处置规则执行审核。 Below this is the '审核流水线' (Review Pipeline) section, which includes a title, a description '按从上到下的顺序执行，最多可配置 8 个阶段', and buttons to add '关键词阶段' (Keyword Stage) or '模型阶段' (Model Stage). The first stage is '阶段 1 关键词审核' (Stage 1 Keyword Review), which is enabled. It shows a list of participating word lists: '全局 · 暴恐 (暴恐)' and '全局 · 色情 (色情)'. There is also a field for '最低命中分' (Minimum Hit Score).

Three modes plus a multi-stage pipeline: run new rules in Observe, then switch to Block once the false-positive rate is confirmed.

① Off / Observe / Block, issued per application ② Stages run in order, up to 8, keyword or model stages ③ Keyword stage: participating word lists and minimum hit score

Off / Observe / Block

Multi-stage pipeline

Local word lists

Model review

Hits must trace to a person, and stop where they should

A hit that can't be tied to a person and module gives no basis for action; keeping the original text creates a new source of risk.

How far the evidence goes

Hit records carry full attribution and link to the exact call by request ID; L3-verified calls resolve to a specific employee and module. Under the **current retention policy, hit details keep only a masked summary and a content-digest hash, not the original text**. The hash checks “is this the same content?”; it cannot restore it. Viewing hit details is itself audited, so what administrators looked at is recorded too.

Blocked calls are still recorded; failures don't stop business

In Block mode a hit request returns 400, while usage and hit records are still stored per the rules, so blocked calls don't vanish from the accounts. If the moderation service fails, the pipeline lets requests through after a timeout rather than blocking model traffic; its status is exposed by monitoring metrics and a “moderation surge” alert. Results support judgment and do not replace human decisions.

请求 ID 3f5c1c1eb35ee6227be9667f7f2f6922	1	时间 2026/8/17 03:41:32
租户 算力共建演示租户 #1		应用 智能会议纪要系统 #1
模块 transcription		终端用户 lisi
API Key 会议转写演示 Key #7		审核阶段 <input type="button" value="关键词"/>
处置结果 ■ 已拦截		
内容摘要哈希 3c9c1022bea0a19af3d706d2fc20e849c748fead535a9873f05fc62b200f9ad3	2	

命中原文

当前留存策略只保留脱敏摘要，未留存原文 3

One hit record, traced to a person and module; under the current retention policy the original text is not kept.

① Full attribution: request ID, tenant, application, module, end user, API Key, moderation stage and action ② Content-digest hash (only for checking whether content is identical) ③ Retention policy in this environment: masked summary only, original text not retained

Full attribution

Masked summary

Viewing is logged

Fail-open on timeout

Keep upstream capacity changes from becoming business outages

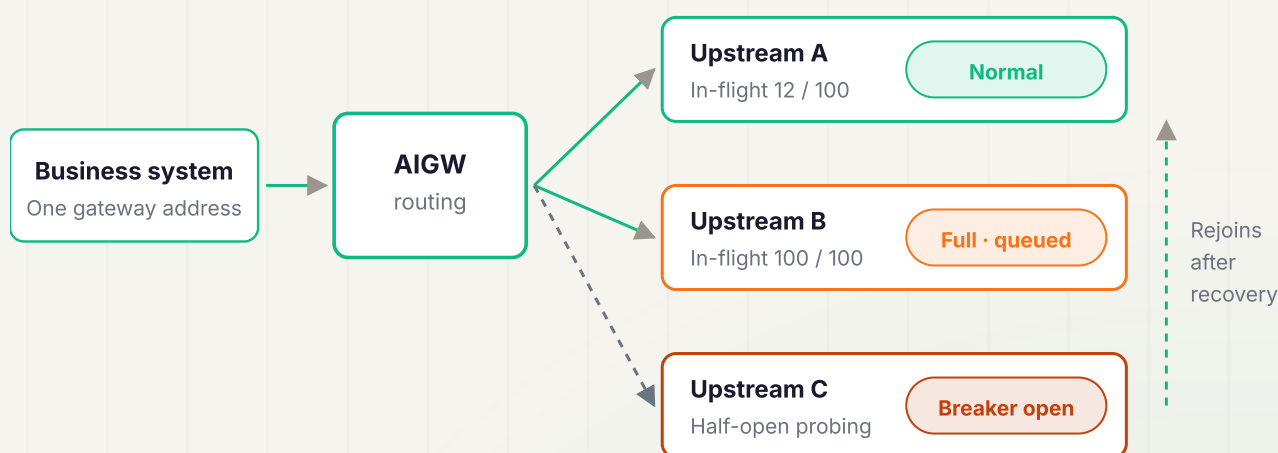
Multiple upstreams are not simple round-robin. AIGW routes by capacity, concurrency, rate, weight and health, and provides transition mechanisms for saturation and change.

This chapter covers three kinds of change

- Upstream failure and recovery: circuit breaking, half-open probing, automatic return
- Compute saturation: queueing and timeout backpressure
- Routine changes: rotating Keys, switching models, upgrading nodes

Business systems see a single gateway address; changes are detected, routed around and recorded at the gateway layer.

ONE ENTRY · THREE UPSTREAM STATES



Not just “can it connect”, but “can it take the load right now”

Upstreams differ in capacity and their load keeps changing. AIGW factors both static configuration and runtime state into routing.

Capacity-aware routing

- Each upstream has a weight, maximum in-flight concurrency, and RPM and TPM capacity
- least-in-flight strategy: prefer the available upstream with the fewest in-flight requests
- If same-tier candidates are full, other candidates are tried; if none is available, routing tiers apply

Fault isolation and recovery

- Consecutive errors trip the circuit breaker and temporarily remove the upstream
- Business traffic shifts to upstreams that are still available
- A recovered upstream enters half-open probing and returns automatically once it passes

Managers get not a static routing table, but routing that follows capacity and health.

■ 区域共建算力池 OPENAI				■ 总部备用算力池 OPENAI			
1 当前状态 熔断中				当前状态 接流正常			
http://127.0.0.1:9310				http://127.0.0.1:9311			
权重 ① 100	在途/上限 ② 0 / 未限制	RPM 用量 ② 未限制	TPM 用量 ② 未限制	权重 ① 100	在途/上限 ② 0 / 未限制	RPM 用量 ② 未限制	TPM 用量 ② 未限制
连续失败 5 次熔断, 900 秒后半开, 探测 30 次 ③				连续失败 5 次熔断, 30 秒后半开, 探测 2 次 ③			
🔑 密钥池 · 0				🔑 密钥池 · 0			
编辑 删除				编辑 删除			

The upstream on the left has been removed by the circuit breaker; traffic has shifted to the available upstream on the right.

① Circuit open (right: serving normally) ② In-flight / limit, RPM and TPM visible live ③ Breaker thresholds and half-open probe settings per upstream

Multiple upstreams

least-in-flight

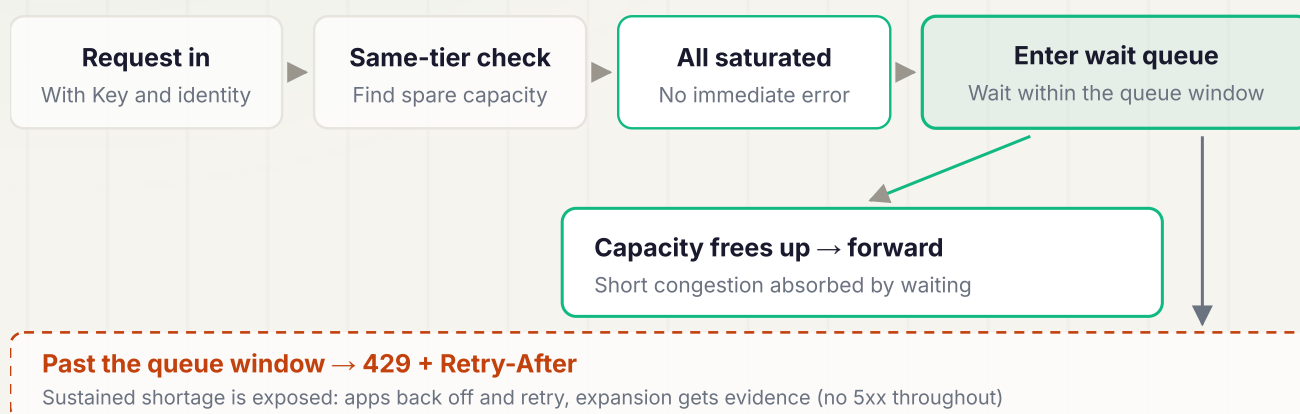
Half-open breaker

Automatic return

When compute is full, queue first instead of pushing load onto model services

GPUs are scarce. Failing requests the moment upstream concurrency is exceeded amplifies business jitter and hides the real capacity demand.

QUEUE INSTEAD OF FAIL FAST



Handling order at saturation

1. AIGW first looks for a same-tier upstream with spare capacity.
2. If the current tier has none, it evaluates later candidates as configured.
3. If all candidates are full, the request enters a wait queue instead of failing at once.
4. When capacity frees up, it is forwarded; past the queue window it returns 429 with Retry-After.

Why this matters

It separates short congestion from sustained under-capacity: the first is absorbed by waiting, the second is exposed through explicit backpressure, so applications can back off and retry, and expansion decisions have evidence.

Capacity queue

429 backpressure

Back-off and retry

Expansion evidence

Rotate Keys, switch models, upgrade nodes, with room to transition

What threatens continuity is often not daily calls but changes that must happen. AIGW builds the change process into how it runs.

Dual-active Key rotation

After a new Key is issued, old and new Keys are both valid during a set grace window; once the business system has switched, the old Key retires, so there is no hard cut-over.

Model alias canary

Business keeps requesting the existing logical model name; the gateway maps it to the actual model per application or upstream scope. Observe locally, then widen; to roll back, remove the mapping.

Node draining

Before an upgrade, the node leaves the new-request entry; once in-flight requests finish and usage is flushed, it restarts and rejoins. The load balancer resolves nodes dynamically and retries a POST on a draining 503 or connection failure only before processing starts; once the response starts, no retry.

Cluster-tier drill results

Conditions: k6 constant 30 concurrency, non-streaming POST via nginx, any non-200 counted as a failure. Two-node rolling upgrade over 6 minutes: 46,389 requests, 0 failures. Control plane down for 60 seconds: 7,638 requests, 0 failures.

时间 ①	请求 ID ②	应用 ②	用户标识 ②	模块 ②	实际模型 ②	响应模型 ②	状态码 ②	TOKEN ②	算力单元 ②	延迟 ②
2026/8/17 02:41:33	b6dbb2d7f357b94f3984dd238f8f96bf	智能会议纪要系统	zhangsan ①	transcription	qwen3-32b	—	200	19	33 ③	28 ms
2026/8/17 02:40:50	f9399ef387cc7933545d21df6145c05c	智能会议纪要系统	zhangsan	transcription	qwen3-235b-a22b ②	—	200	19	33	29 ms
2026/8/17 02:39:06	073f020621a6afe864ccd1af796e2d82	智能会议纪要系统	zhangsan	transcription	qwen3-235b-a22b	—	200	19	33	20,012 ms
2026/8/17 02:39:06	f8bbbe29fa4acb8a144e3039f2081b0b	智能会议纪要系统	zhangsan	transcription	qwen3-235b-a22b	—	200	19	33	20,013 ms
2026/8/17 02:39:06	2a638b2c39884110295c627cfab56d0a	智能会议纪要系统	zhangsan	transcription	qwen3-235b-a22b	—	200	19	33	20,015 ms
2026/8/17 02:39:06	43dff5ebcc3268a98f46bd7784cc641c5	智能会议纪要系统	zhangsan	transcription	qwen3-235b-a22b	—	200	19	33	20,015 ms

① The canary applies only to the chosen application: business keeps requesting the same logical model name, while the executed model is already the new one.

① Calls resolve to a specific employee identifier ② The executed model is already the new canary model ③ TOKENS, compute units and end-to-end latency per call

Dual-active Keys

Model canary

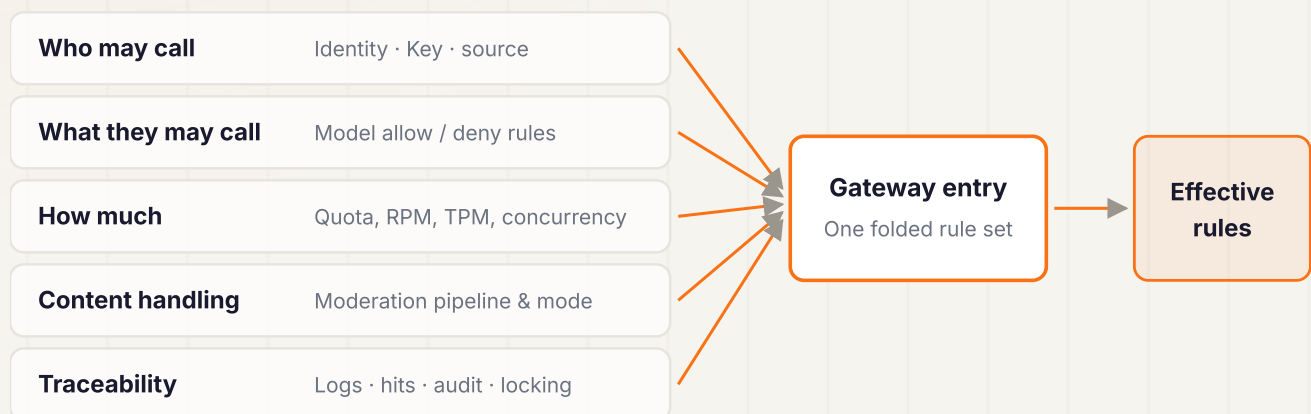
Node draining

Drill results

Rules converge at the entry, and every action leaves one trail

Tenants, applications and Keys have different requirements, but before reaching model services they must resolve to one explicit, auditable set of effective rules.

FIVE CONTROLS · ONE ENTRY



Scattered across systems, rules disagree; converged at one entry, they are single, clear and auditable.

One entry for five kinds of control

Who may call (identity, Key status, source range); what they may call (model allowlists and deny rules); how much (quota, RPM, TPM, concurrency); how content is handled (moderation pipeline and mode, see Chapter 3); how actions are traced (call logs, hit records, audit logs and monthly locking).

A closed audit chain

Call details record request ID, identity, model, status and usage; hit records explain moderation results and actions; audit logs record sensitive operations such as Key revocation and rotation, policy changes, configuration edits, viewing hit details and ledger locking: who, when, on which object and why. Entries are human-readable, with raw values kept for technical checks.

Monthly locking

A locked period is never recalculated, and its weight version is kept; backfills and recalculations do not change locked figures, and locking itself is audited.

Unified admission

Quotas and rate limits

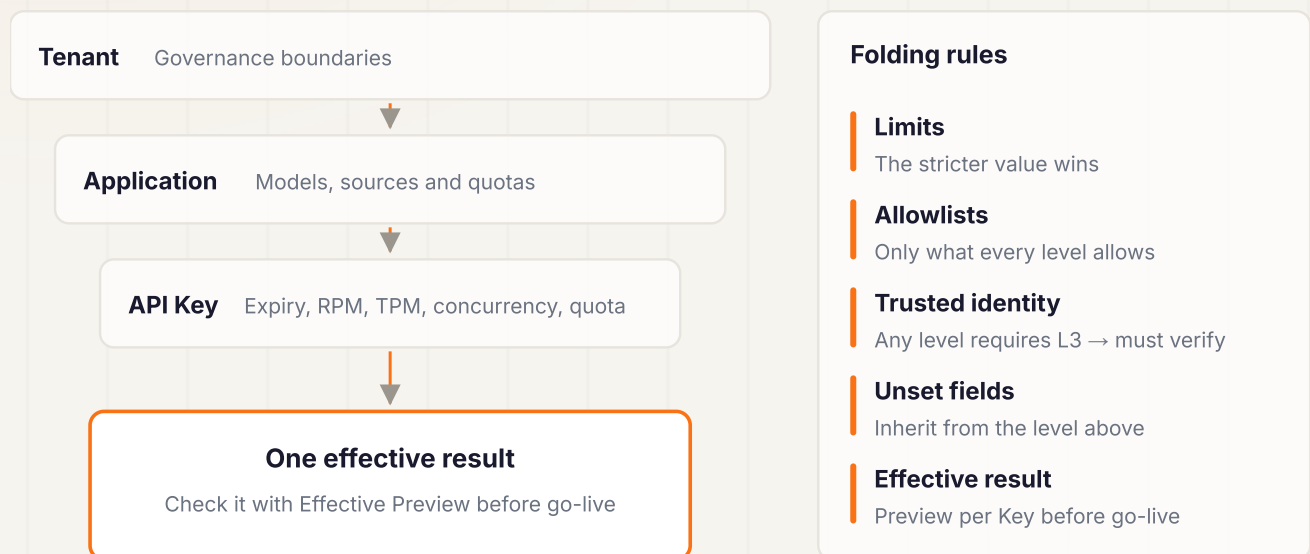
Audit trail

Monthly locking

Tenants set boundaries, applications set scenarios, Keys set the entry

Policies inherit and fold across three levels: tenant, application and API Key. More specific levels add their own requirements, producing one effective result.

TENANT · APP · KEY



Why three levels instead of one rule set for everything

Governance boundaries are set by the tenant, business scenarios by application owners, and specific entry points by operators when they issue Keys. The three care about different granularity; written together, they overwrite each other. With levels, each maintains only its own part and unset fields inherit from above; the gateway computes the folded result, and every Key can be previewed before go-live.

Tenant policy

Application policy

Key policy

Effective preview

Choose the topology by availability, then size nodes by in-flight load

One binary supports single-node, active-standby and cluster deployment, for controlled private delivery. The core reason for a cluster is availability and rolling upgrades, not more processes.

SINGLE · ACTIVE-STANDBY · CLUSTER

Single node

On-site POC / regional pilot

- AIGW binary ×1
- PostgreSQL
- Local counters

Active-standby

Standard production

- AIGW ×2 + front VIP
- Shared PostgreSQL
- Redis shared counters

Cluster

Multi-region · rolling upgrade

- Data plane scales out
- Central config & accounts
- Pending usage in local WAL

One binary, three topologies; a cluster is for availability and rolling upgrades, not process count.

Capacity sizing baseline

For a public-sector or enterprise system with 20,000 registered users: about 6,000 daily actives, about 120,000 calls a day, peak QPS 25–40, 200–300 in flight, about 120,000 usage detail rows a day and 3.6 million a month. One data-plane node comfortably handles about 500 in-flight connections; 3 nodes cover that peak with 5× headroom.

Single-node all-mode test

Single node, all mode, macOS arm64, k6 v2.2.0; 300 concurrency steady for 60 seconds, then a 600-concurrency spike, with a simulated upstream injecting 120ms. 38,633 requests, 0.00% failure rate, 0 5xx; p95 1361ms (including the 120ms simulated upstream injection); 38,633 new rows stored, matching the requests sent.

v2.0.00 engineering evidence

76,765 lines of Go source and 486 test functions; 145 Playwright UI cases expanded at runtime, covering every console page; 33 database migration files.

Single-node pilot

Active-standby

Cluster

Capacity sizing

One governance, two kinds of traffic

AI calls and ordinary HTTP APIs run on different data planes and share one control plane's identity, subscriptions, policies, RBAC and auditing.

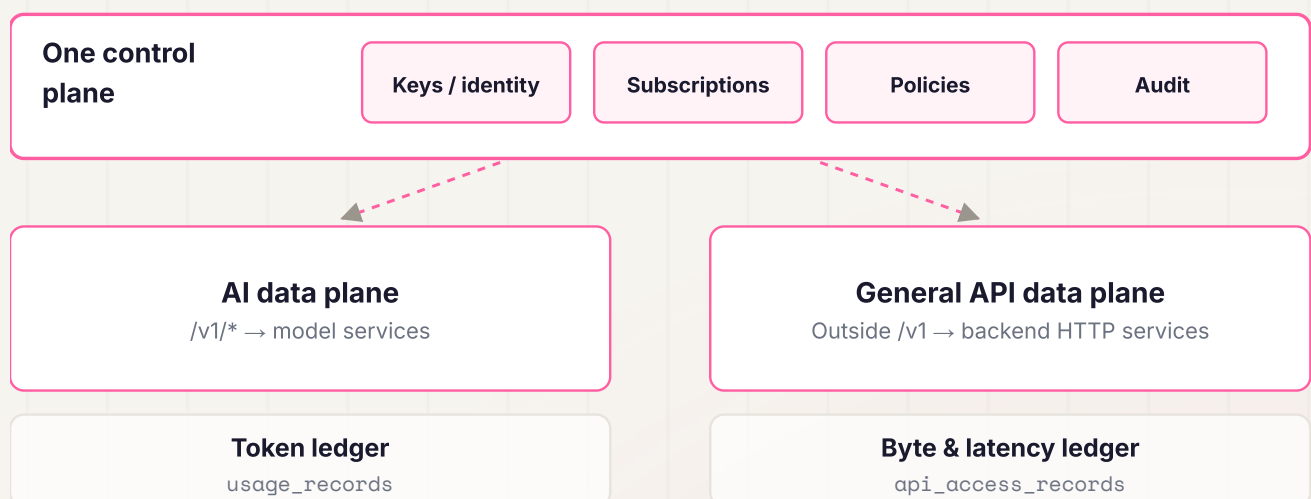
This chapter answers three questions

- How the two data planes divide the work, and why the two ledgers keep separate bases
- How an API asset goes from registration and publication to stable service
- How plan subscriptions and five-level policies produce an explainable effective result

Switch boundary

The general API data plane is controlled by `AIGW_API_CATALOG_ENABLED` / `AIGW_API_PLANE_ENABLED`, disabled by default and enabled as needed; when off, behavior is identical to v1.9.x, with zero changes to the AI path.

ONE GOVERNANCE · TWO PLANES



Two switches control the general API data plane; both are off by default.

One control plane

Two data planes

Off by default

Enabled as needed

Carried separately, governed together

Each data plane carries one kind of traffic; both share identity, auditing and operations.

	AI data plane	General API data plane
Carries	Nine AI endpoints under /v1/*	Paths outside /v1, matched by published config
Specific capabilities	Model aliases, protocol conversion, content moderation, capacity-aware routing	Route matching, backend pools, SSRF protection, subscription admission, policies
Metering	Token ledger	Byte and latency ledger (separate table)

The catalog shows only what it should

Only published and deprecated services are listed; backend addresses and pools are never exposed. Reports never total across the two bases.

路由

当前生效版本对消费者开放的调用路径。

1

Method	Path	名称	认证
GET	/archive/v1/records	listArchiveRecords	API Key
POST	/archive/v1/search	searchArchiveRecords	API Key
GET	/archive/v1/records/{id}	getArchiveRecord	API Key

环境地址

已部署当前服务的可用入口。

2

生产

prod · Revision 2

api.demo.local

测试

test · Revision 2

api-test.demo.local

套餐

选择符合认证方式、额度 and 有效期要求的套餐。

3

标准套餐

standard

人工审批

适用于日常办公系统的人工审批套餐。

认证: API Key RPM: 60 有效期: 90 天

The catalog shows only published and deprecated services; callers never see backend addresses or pool configuration.

① Service name and status ② Subscribable plans ③ Calling instructions (demo data)

Laboratory benchmark

k6, 300 VUs / 60s, GET, all on one 8-core machine: bare backend 74,808 req/s; through the gateway, anonymous 22,008 and API Key 17,404 req/s; 0 failures in all three groups. p95 reflects queueing under CPU saturation on the shared machine, not gateway processing cost; in-process benchmarks show about 1.5 μ s net overhead per request.

Shared governance

Separate metering

Service catalog

Lab benchmark

Published config never changes history; every change is a new record

Treat APIs as governed assets: register a service, edit a draft, publish it to an environment, and let the data plane run the published config reliably.

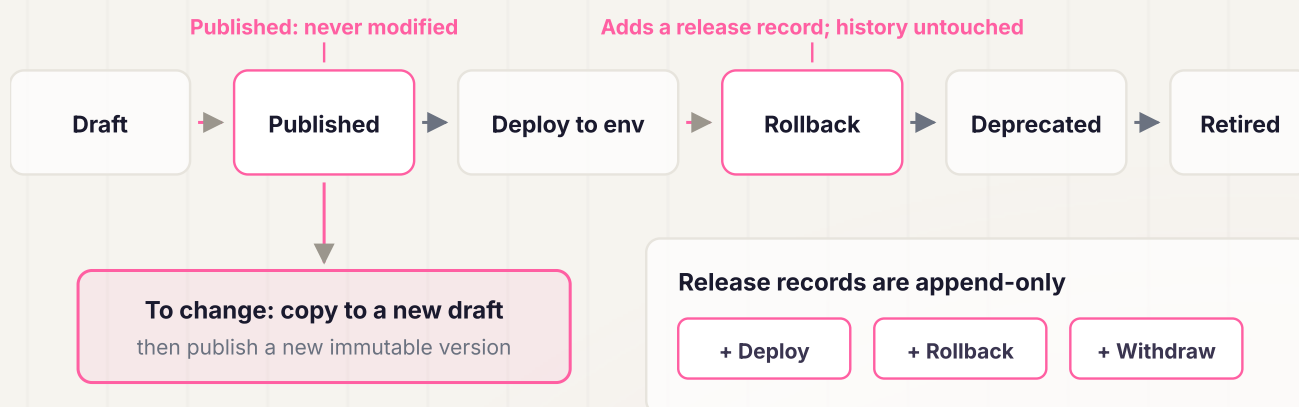
Immutable versions

A published version cannot be modified; changes and rollbacks are made as new drafts and new release records.

Pre-publish validation

Publishing checks reserved paths, routes, backend pools, backend addresses and path templates; backend addresses are re-validated on every dial, and internal backends must be explicitly allowed.

IMMUTABLE VERSION · APPEND-ONLY RELEASE



Published versions never change; a rollback is a new action, not an undo or rewrite of old records.

Consistent matching and hot reload

Routes match on Host, path, method and headers, with a fixed priority order so every node gets the same result. The data plane compares versions every 5 seconds, precompiles and switches atomically; in-flight requests are unaffected, and if compilation fails the previous version stays and an alert is raised.

Admission is approved, limits are explained

The catalog tells callers what exists; plans and subscriptions decide whether they may call; the policy engine works out how much, and under which rules.

From request to activation

Subscriptions run request → approval → active → expiry reminder; approval is concurrency-safe and safe to retry. Once a service has plans, Keys without an active subscription get 403 not_subscribed. In this version, expiry notifications are stored only, not sent.

Authentication boundary

The data plane requires a same-tenant API Key and can also verify JWTs from an external IdP. AIGW only verifies assertions; MFA / SAML / password policies belong to the IdP, and console login tokens cannot be used on the data plane.

Policies that explain themselves

Policies cover IP access, rate / concurrency / daily quota, request size and header transformation, bound at five levels: global → service → route → application → subscription; for the same type, later levels override earlier ones. The “effective explanation” lists the final policies and what was overridden.

生效解释

查看指定路由与应用最终采用的策略链。

路由 [?]

应用 ID (可选) [?]

1

r-get-archive-v1-records

2

查询

1	全局	全局基础限流	rate_limit	被下层覆盖
2	服务	办公网段放行	ip_access	
3	服务	统一追踪头	header_ops	
4	路由	标准限流	rate_limit	3 被下层覆盖
5	路由	标准请求大小	request_size	被下层覆盖
6	应用	应用级限流	rate_limit	被下层覆盖
7		plan:standard	rate_limit	被下层覆盖
8		plan:standard	request_size	
9	订阅	标准限流	rate_limit	

The effective explanation lists the final policies level by level and marks which settings later levels overrode.

① Policy source level ② Final effective items ③ Overridden items and reason (demo data)

[Plan subscriptions](#)[Approval and activation](#)[Five-level policy](#)[Effective explanation](#)

Who it fits, and who it does not fit at this stage

Stating the boundary up front saves both sides a great deal of pre-sales explanation and mismatched comparisons.

A good fit

- Government, enterprise groups and large organizations that need private deployment
- Public-facing AI apps whose content risk must be controlled before leaving the gateway
- Several regions, departments and apps sharing AI, needing traceability to people and modules
- Self-hosted model services, shared compute pools or several model sources
- Existing HTTP interfaces to open to sister units, subsidiaries or external partners
- Internal accounting, allocation, auditing or monthly locking
- Bringing end-employee identity into the model call chain

Not our focus at this stage

- A simple model proxy for one application or one team
- Public API top-ups, plans and resale operations
- The broadest possible model protocol conversion and provider coverage
- A complete enterprise identity platform (IAM / SSO)
- A scheduler for GPU pod placement, memory partitioning or model replicas

General API data plane boundary

Disabled by default and enabled as needed; when off, behavior is identical to v1.9.x, with zero changes to the AI path. AIGW is a governance entry point for identity, subscriptions, policies, auditing and access records; it does not take on every role of site-wide traffic ingress and layer-7 load balancing.

Delivery model

AIGW is delivered as a controlled, project-based private deployment: on-site POC, deployment in the customer's environment, and upgrade and maintenance support with each release.

Private deployment

Content compliance

API governance entry

Clear boundaries

From the next call on, make people, content, rules and accounts clear

One control plane governs AI calls and ordinary APIs: every call has an identity, rules, moderation, an owner and a record.

One call, six links, all accounted for



Who, in which org, via which app and module, under which rules, on which model, in which ledger.

Govern the content

Off / Observe / Block modes, a multi-stage pipeline and self-maintained word lists; under the current retention policy hits keep only masked summaries, and viewing them is audited.

Trace the caller

L1/L2 support statistical attribution; L3 is signed by the server and verified by the gateway, for trusted attribution and accountability.

Account for every call

System, user and department ledgers; six-dimension combinations; frozen snapshots; monthly locking.

Keep business running

Circuit breaking and return, queue on saturation, dual-active Keys, model canaries, node draining; cluster-tier drills measured 0 failures at constant 30 concurrency with non-streaming POST.

One governance, two kinds of traffic

AI calls and ordinary APIs share identity, subscriptions, policies and auditing; the general API data plane is disabled by default and enabled as needed.

Next step: request the POC acceptance checklist

It lists the verification scope, environment prerequisites, step-by-step acceptance procedures and pass criteria, for internal evaluation or to schedule an on-site POC. Sent on request.

Chongqing HongChuangDa Technology Co., Ltd.

Web: www.cqagc.com · Email: dengyao@cqagc.com

Address: Room 11-7, Unit 1, Building 13, No. 18 Qixia Road, Liangjiang New Area, Chongqing, China

AIGW

AI + API governance

Content compliance

Controlled private delivery