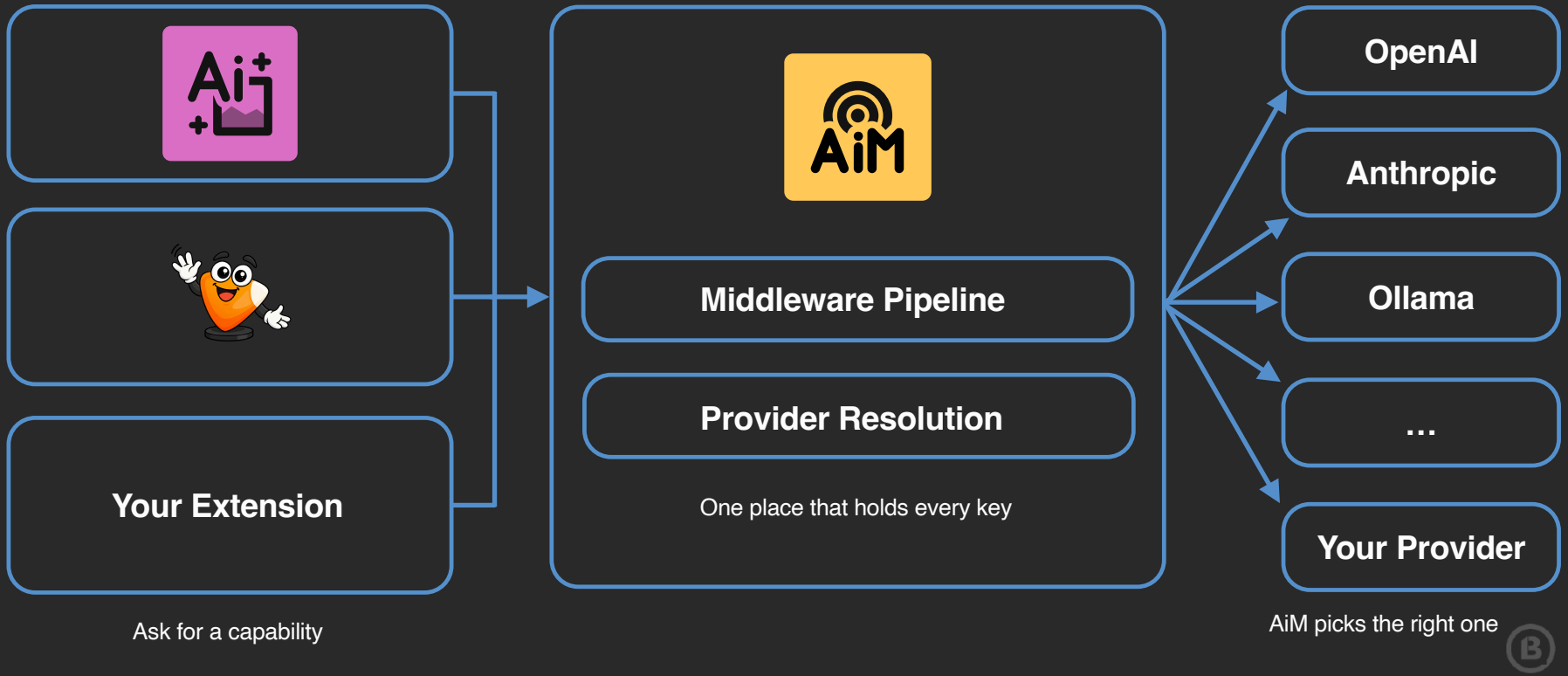


Intelligent Ai Proxy for TYPO3

One layer. All providers. Full control.

One layer between extensions and providers



AiM

Not just a place to store API keys



Simple API for developers

no provider knowledge needed



Governance

Access control, budgets, rate limits, privacy levels



Central provider management

one place, all extensions



Smart routing

routes simple prompts to cheaper models
uses fallbacks for incompatible models



Full visibility

every request logged with cost and tokens



Extensible

middleware pipeline, events, custom providers

AiM

is …

- A central AI proxy layer
- A governance and transparency framework
- A provider-agnostic API for extension developers
- A multi-layer middleware pipeline

is not …

- A specific AI feature (no chatbot)
- Tied to any provider, and ships none
- Replacement for your AI tool(s) extension(s)

can be used for …

- Image analysis and alt text generation
- Content generation
- Image Generation
- Translation
- Conversation and chatbots
- Embeddings and semantic search
- Tool calling and agentic workflows

… and so much more

AiM in four numbers

11

Middleware Stages

7

Capabilities, from vision to embeddings

3

Backend modules: Providers, Request Log, Prompt Management

0

AI provider dependencies



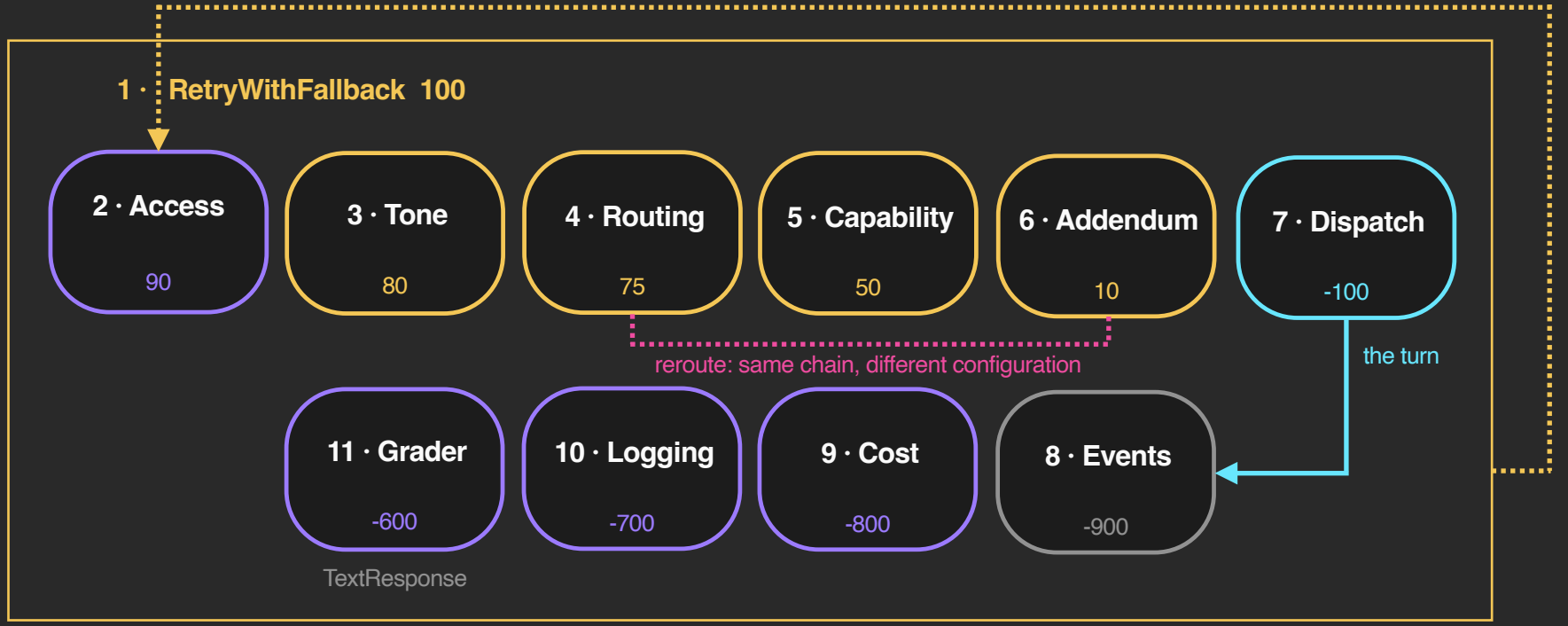
Middleware Pipeline

The Heart of AiM



AiM

Middleware Pipeline



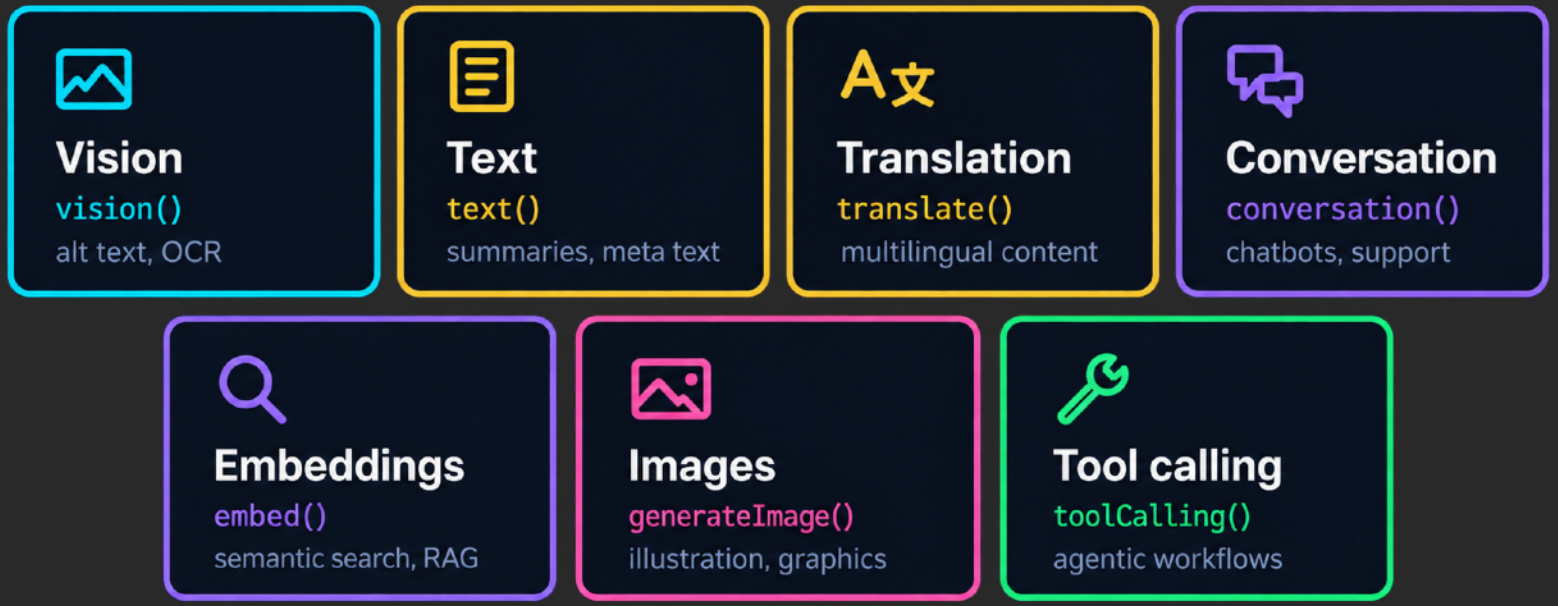
Six on the way in, the provider call is the turn, four on the way out

Capabilities

Right model, right job

AiM

Capabilites



Each maps to a provider interface. Providers implement what they support.



TIER 1 · The Proxy

Eight methods are the whole surface

```
$this->ai->text($prompt);
$this->ai->vision($imageData, $mimeType, $prompt);
$this->ai->translate($text, 'English', 'German');
$this->ai->conversation($messages);
$this->ai->conversationStream($messages);
$this->ai->toolCalling($messages, $tools);
$this->ai->embed('TYPO3 is an open-source CMS');
$this->ai->generateImage('A TYPO3 logo made of cake’);

$this->ai->request();  // ... and the builder, tier 2
```

TIER 1 · The Proxy

A single call

```
$response = $this->ai->vision(
    imageData: base64_encode($fileContent),
    mimeType: 'image/jpeg',
    prompt: 'Generate alt text for this image',
    extensionKey: 'my_extension',
);
```

No API key. No provider. No SDK. Just ask.

The same call inside a real class

```php
class MyImageProcessor
{
    public function __construct(private readonly Ai $ai) {}

    public function generateAltText(string $imageBase64): string
    {
        $response = $this->ai->vision(
            imageData: $imageBase64,
            mimeType: 'image/jpeg',
            prompt: 'Generate alt text for this image',
            extensionKey: 'my_extension',
        );
        return $response->content;
    }
}
```

*Constructor injection, one call, read **content**. That is the whole integration.*

TIER 1 · The Proxy

Chat, embeddings, images

```
$this->ai->conversation([new UserMessage('What is TYPO3?')], extensionKey: 'my_ext');

$this->ai->embed(input: 'TYPO3 is the best CMS', dimensions: 256, extensionKey: 'my_ext');

$this->ai->generateImage('TYPO3 logo', options: ['quality' => 'high'], extensionKey: 'my_ext');
```

Same proxy, three more capabilities. Named arguments throughout.



TIER 1 · The Proxy

Asking for a specific provider

```
$response = $this->ai->text(
    prompt: 'Summarize this.',
    provider: 'openai:*',
    extensionKey: 'my_ext',
);

$response = $this->ai->vision(
    imageData: $data,
    mimeType: 'image/jpeg',
    prompt: 'Describe this image',
    provider: 'openai:gpt-4.1',
    extensionKey: 'my_ext',
);
```

TIER 1 · The Proxy

Asking for a specific provider without hardcoding one

- Pass nothing and you get the configured default for that capability.
- *'openai:*'* · this provider, the admin's choice of model.
- *'ollama:llama3.3'* · this provider, this exact model.
- Unavailable? AiM falls back to the default and logs a warning.
- A borrowed credential brings its group, privacy and rerouting rules with it.
- Rerouting Decision is logged.

TIER 1 · The Proxy

Streaming, token by token

```
$response = $this->ai->conversationStream(
    $messages,
    extensionKey: 'my_ext',
);

new ($response->cstreamIterator ?? [] as $chunk) {
    echo $chunk; // Send as SSE to the browser
    ob_flush();
    flush();
}
```

TIER 2 · The Builder

When you want the knobs

```
$response = $this->ai->request()
    ->vision($imageData, 'image/jpeg')
    ->prompt('YouGenerate alt text for this image')
    ->systemPrompt('You are an accessibility expert.')
    ->maxTokens(100)
    ->temperature(0.3)
    ->provider('openai:*')
    ->from('my_ext')
    ->send();
```

Still provider-agnostic where you want it. Specific where you need it.

TIER 2 · The Builder

Structured output, when you need JSON

```
$response = $this->ai->request()
    ->text()
    ->prompt('Extract name and price from "MacBook Pro, $2449.99"')
    ->responseFormat(ResponseFormat::jsonSchema('product', [
        'type' => 'object',
        'properties' => ['name' => ['type' => 'string'], 'price' => ['type' => 'number']],
        'required' => ['name', 'price'],
        'additionalProperties' => false,
    ]))
    ->from('my_ext')
    ->send();

$data = json_decode($response->content, true);
```

TIER 3 · The Pipeline

When you want all of it

```
$resolved = $this->providerResolver->resolveForCapability(
    TextGenerationCapableInterface::class,
);
$request = new TextGenerationRequest(
    configuration: $resolved->configuration,
    prompt: 'Write a meta description for a bakery site.',
    maxTokens: 160,
    metadata: ['extension' => 'my_ext'],
);
$response = $this->pipeline->dispatch($request, $resolved);
```

Resolve, build the request yourself, dispatch. Same eleven stages.

> All three tiers flow through the same middleware chain: logging, governance, cost tracking and events always fire, whichever way the request was made.


bin/typo3 aim:test

Command line usage

```
# Text generation (default capability)
bin/typo3 aim:test text --prompt "Write a haiku about TYPO3"

# Conversation, against a specific provider
bin/typo3 aim:test conversation -p "anthropic:*" --prompt "Explain AI”

# Translation
bin/typo3 aim:test translate --prompt "Hello world" --from "English" --to "German"

# Embeddings
bin/typo3 aim:test embed --prompt "TYPO3 is an open-source CMS"
```

TIER 1 · The Proxy

What your extension stops containing

- API key
- Provider name(s)
- Model name(s)
- SDKs
- HTTP clients

Ask for a capability; AiM resolves the configuration behind it.



For administrators

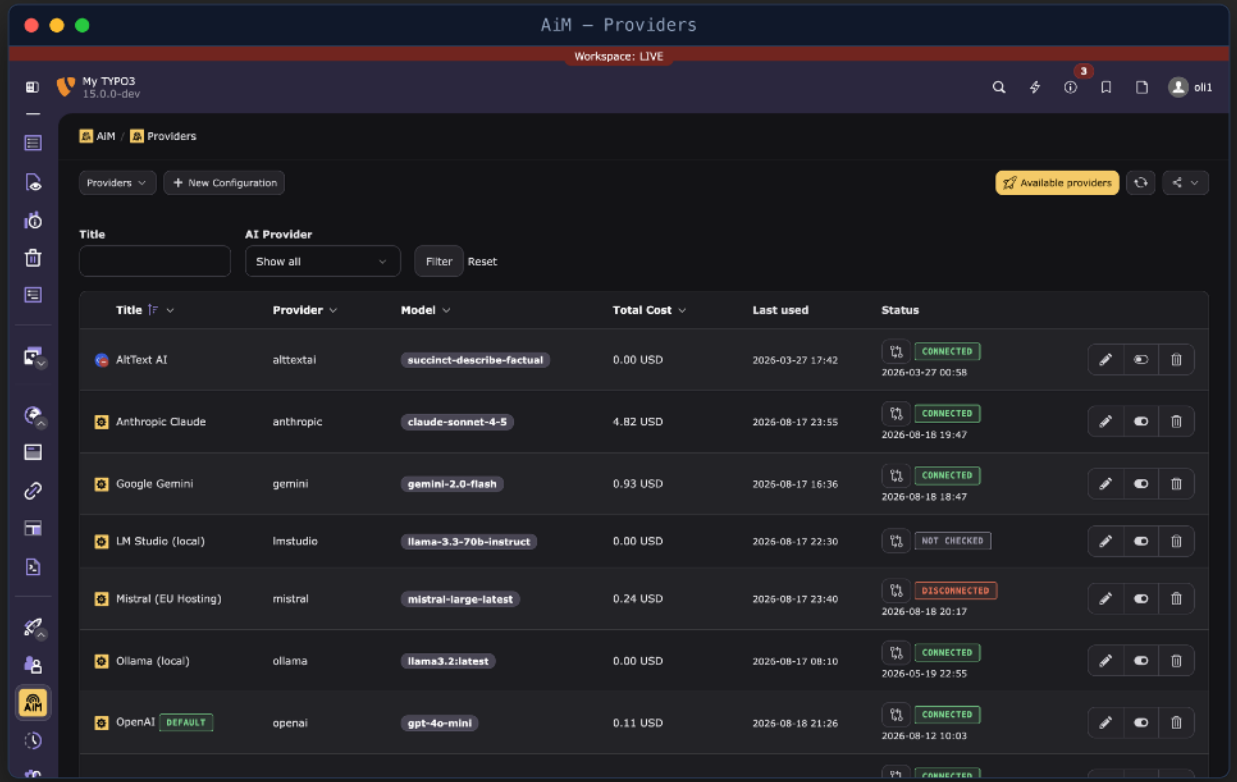
No PHP from here on.

Provider Configuration

Configure once. Use everywhere.

Providers Module

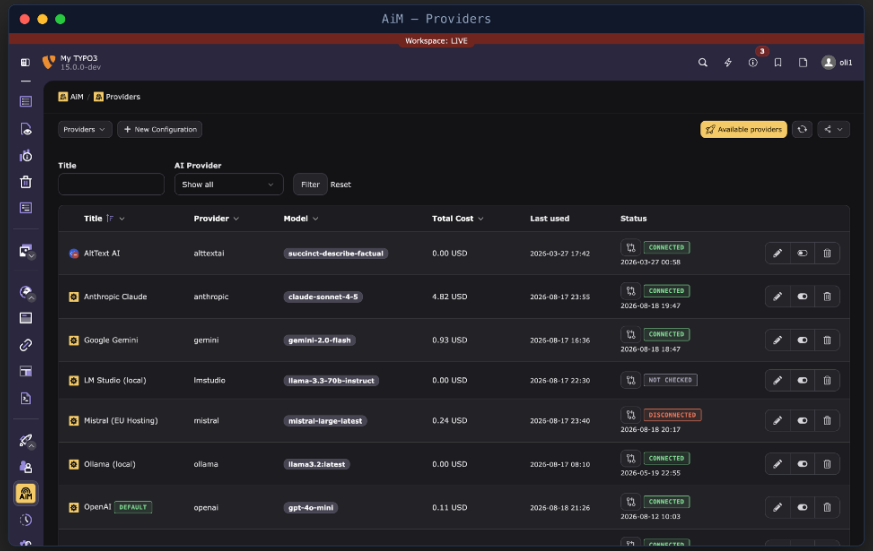
Every AI slot in the installation



Providers Module

Every AI slot in the installation

- Manage provider configurations (Model, API key, …)
- Test connection status with one click
- Track total cost per configuration
- Enable / Disable models via clickable badges



Switch models site-wide by changing one record. No code changes. No deployments.

AiM

One record is one AI slot

What you fill in

- **Provider** · *OpenAI*
- **Endpoint URL** · *http://host.docker.internal:11434*
- **API Key** · *sk-proj-*
- **Model** · *gpt-4o-mini*



What it does untouched

- **Privacy level** · *standard*
- **Rerouting** · **both directions** *on*
- **Auto model switch** · *on*
- **Group Restrictions** · *off*

AiM

Provider auto-discovery

Zero hardcoded providers. Install what you need. Or build your own!

```
composer require symfony/ai-open-ai-platform
composer require symfony/ai-anthropic-platform
composer require symfony/ai-gemini-platform
composer require symfony/ai-ollama-platform
composer require symfony/ai-mistral-platform
```

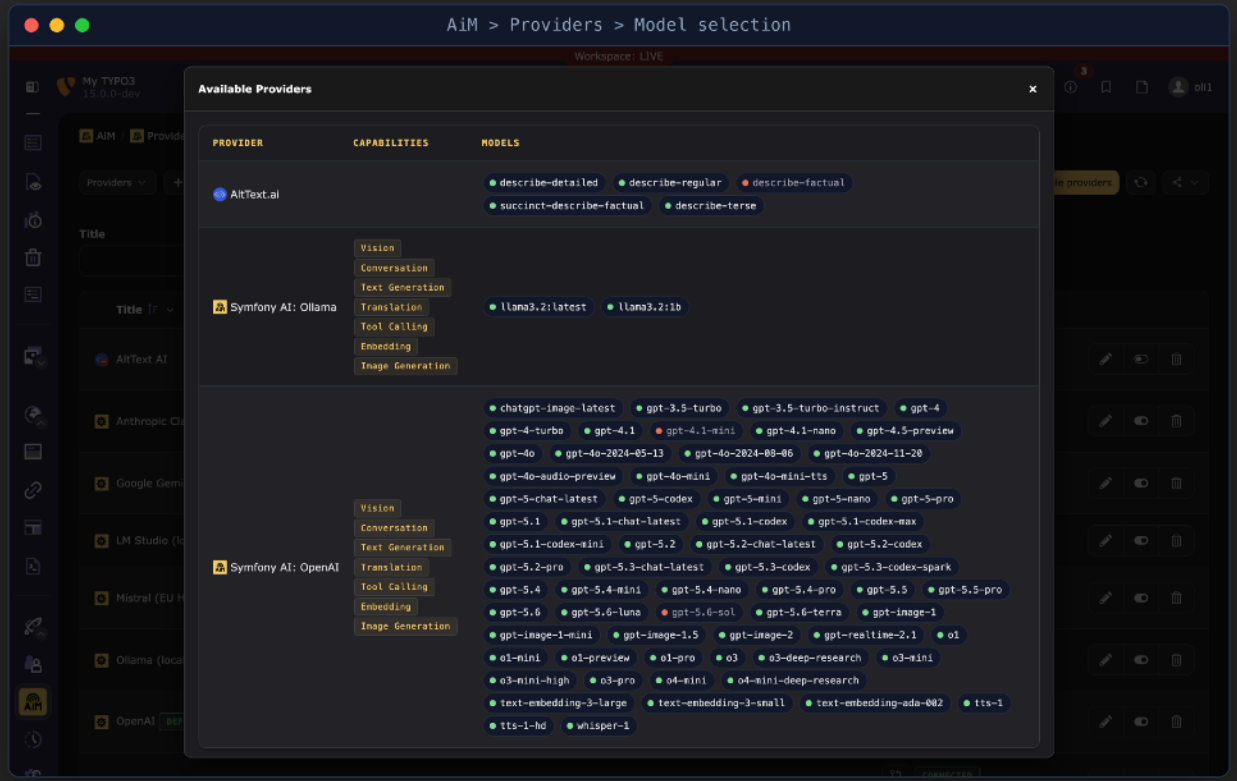
```
#[AsAiProvider]
```

*Any installed **symfony/ai*-platform** package is **auto-discovered**. Models, capabilities, and features read automatically from the bridge.*



AiM

Models & Capabilities



Models auto-discovery



The credential is used for this one request, on the server.

It is never rendered into the form, the dropdown or the page source.

a host that answered

900 s

unreachable, refusing or empty

60 s

Credentials

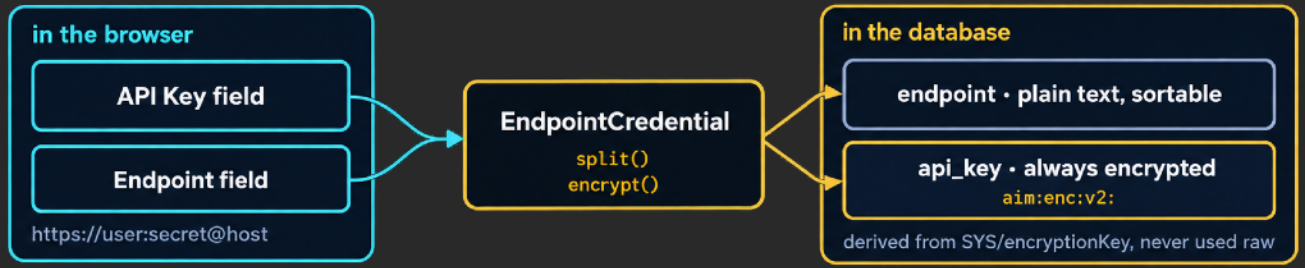
The key goes in once, and never comes out



Credentials

The key goes in once, and never comes out

Security by default





Automatic key rotation included `bin/typo3 aim:rotateApiKeys`

Governance

Your DPO has opinions. AiM has settings.

Access control

Group restrictions

Provider configs can be limited to specific backend user groups

Capability permissions

Each capability can be restricted via custom TYPO3 permissions per user group

Budget limits (via User TSconfig)

```
aim.budget {
   period = monthly
   maxCost = 50.00
   maxTokens = 500000
   maxRequests = 1000
}
```

Rate limiting

```
aim.rateLimit.requestsPerMinute = 10
```



Privacy levels

Three levels, configurable per provider configuration

Level	*Prompts logged*	*Responses logged*	*Token counts*	*Costs*
Standard				
Reduced				
None				

```
aim.privacyLevel = reduced
```



***Strictest wins**: admin sets minimum, user can escalate via TSconfig, individual request can override further. Nobody can relax it downward.*

Smart Routing

Don't pay for GPT-4o to answer "Hello"

AiM

Smart Routing

AiM can automatically downgrade simple requests to cheaper models.

Complexity analysis

- Character count and sentence structure
- Presence of code blocks, URLs, enumerations
- Question density and structural markers
- Multi-language keyword signals (extensible)

Downgrade only if all hold

- Historical cost is lower than current model
- Success rate ≥ 90% (min. 10 requests in history)
- With grading enabled: ~ quality grade ≥ 0.65
- Not more than twice as slow.

Everything is logged, including the CO_2e it avoided. You see exactly what was rerouted and why.

AiM

Smart Routing

rerouting_allowed = 0

- May this configuration's requests go elsewhere? No.
- No smart-router downgrade, no retry after a failure.
- Pins the local Ollama that sees the HR data.

accepts_rerouted_requests = 0

- May other configurations send requests here? No.
- Keeps foreign traffic off this credential.
- Defaults to on.

Auto Model Switching

Your model can't do that. AiM finds one that can.

AiM

Auto Model Switching

What if the configured model doesn't support the requested capability?

Finds	text-embedding-3-small (same provider, embedding capable)
Picks	Cheapest model with positiv success rate in history
Creates	Ephemeral config (no DB write) with same API key
Logs	Original model, new model, reason
Result	Caller gets response. No error, no intervention needed.

One provider configuration covers all seven capability types.



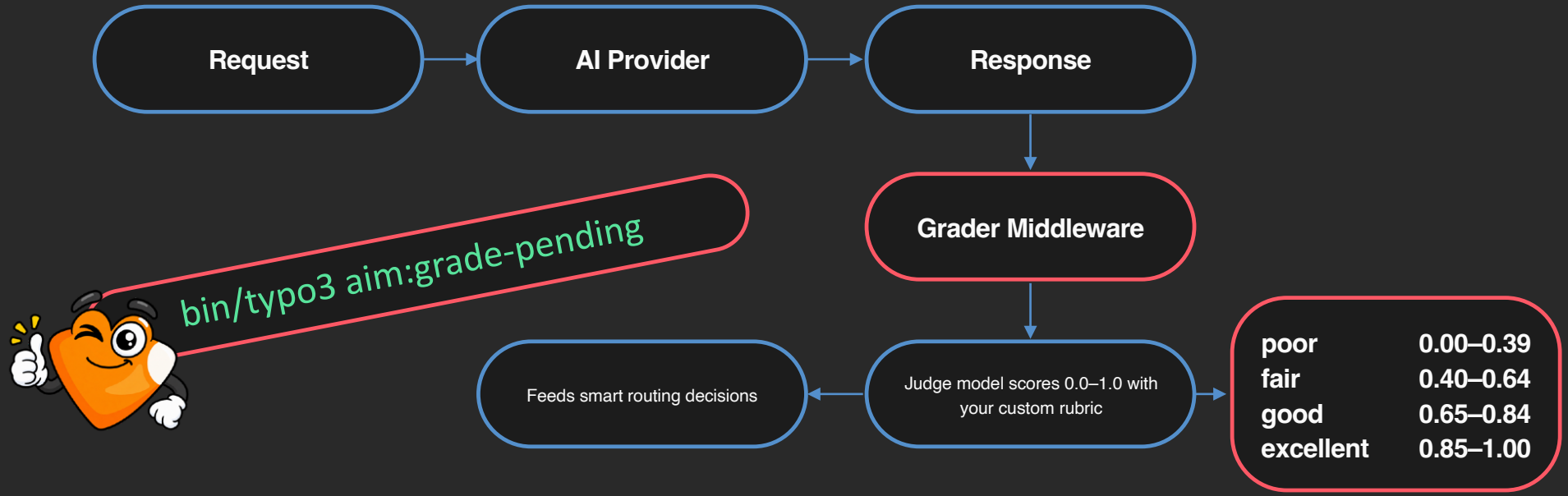
Performance Rating



AiM

LLM grading

Close the quality feedback loop: A second model acts as a judge for your AI responses.



Non-blocking: grading runs asynchronously after the response. Grades from smart routing history, so quality gates itself automatically.


Transparency
For not going bankrupt

AiM - Request Logging

Transparency

Every request recorded with

Origin

Extension that made the call, backend user who triggered it

Decision

Provider + model used (and what was *originally* requested)

Token breakdown

prompt / completion / cached / reasoning

Cost

Based on provider configuration, acutal used model and token usage

Complexity score

Including rerouting details

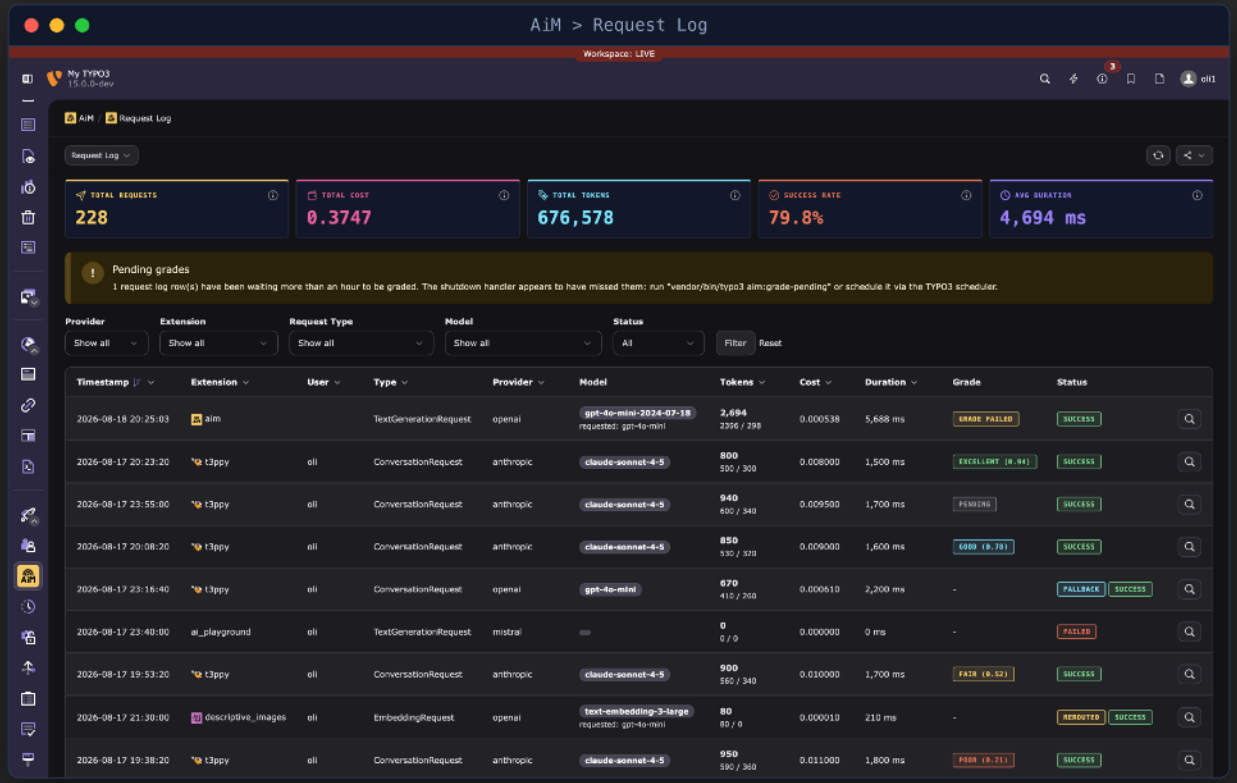
LLM quality grade

Grades the response, if enabled in the used provider configuration



AiM - Request Logging

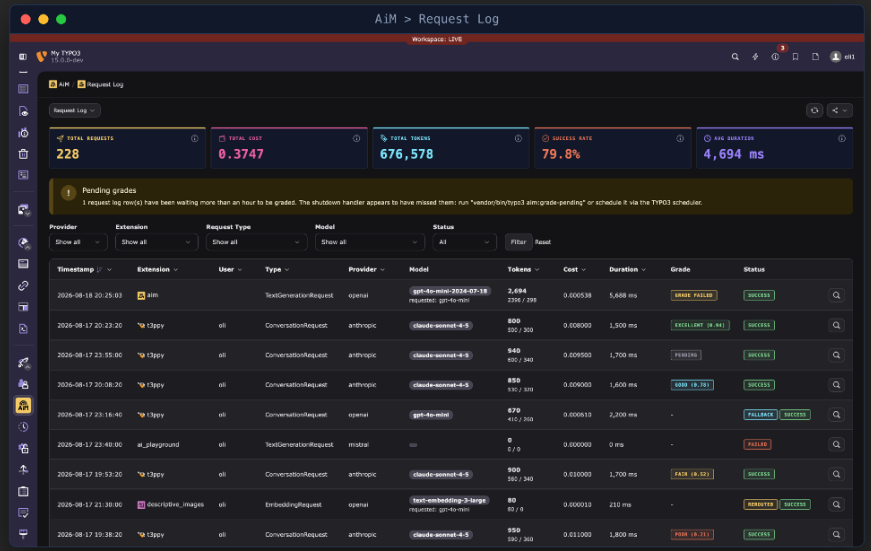
Overview



AiM - Request Logging

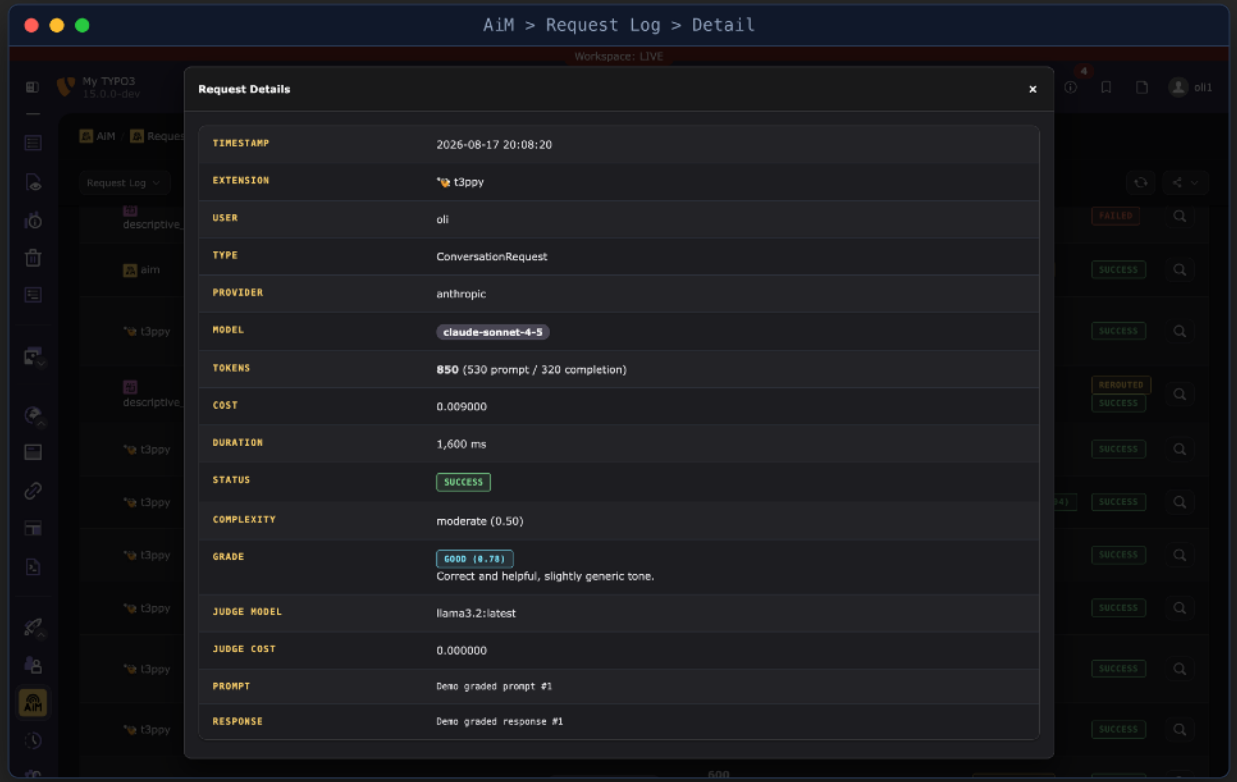
Overview

- Summary cards for a quick check
- Filter by provider, model, extension, user
- Inspect prompts, responses, token usage, cost, fallback, rerouting, duration, …



AiM - Request Logging

Details



AiM - Request Logging - Footprint

What did that request cost the planet?

Hosted (OpenAI, Anthropic, Gemini, Mistral)

EcoLogits model:
output tokens × model size × data center → a min–max

Local (Ollama & co.)

inference time × the power draw you measured

No data, never zero

every total says how many requests it covers

Smart routing

CO_2e avoided, or added, compared with the requested model

Estimates, not measurements. Usage phase only.

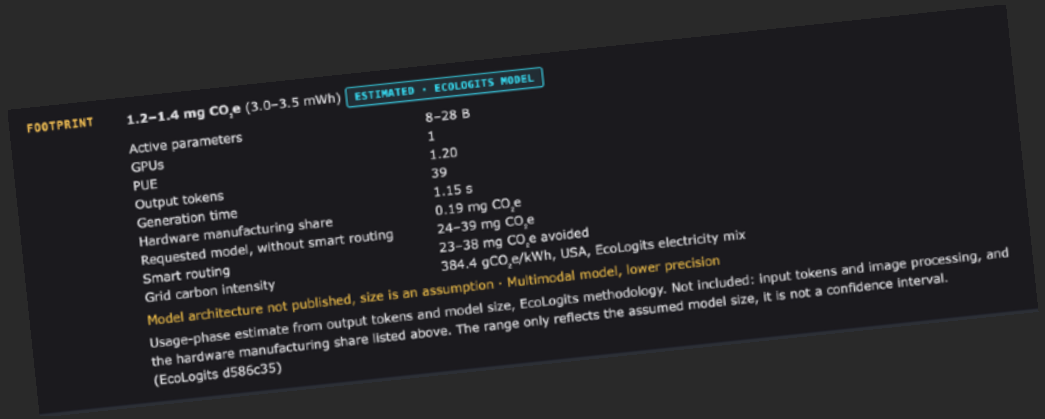
Frugal AI - measured

Same alt-text job

gpt-4o-mini: 0.64–1.0 mg CO_2e

local gemma3:4b on an M4 Pro: 5.7–6.0 mg CO_2e

> local isn't automatically greener. The grid and the hardware decide, and now you can see it.





AiM - Request Logging

Dashboard Widgets

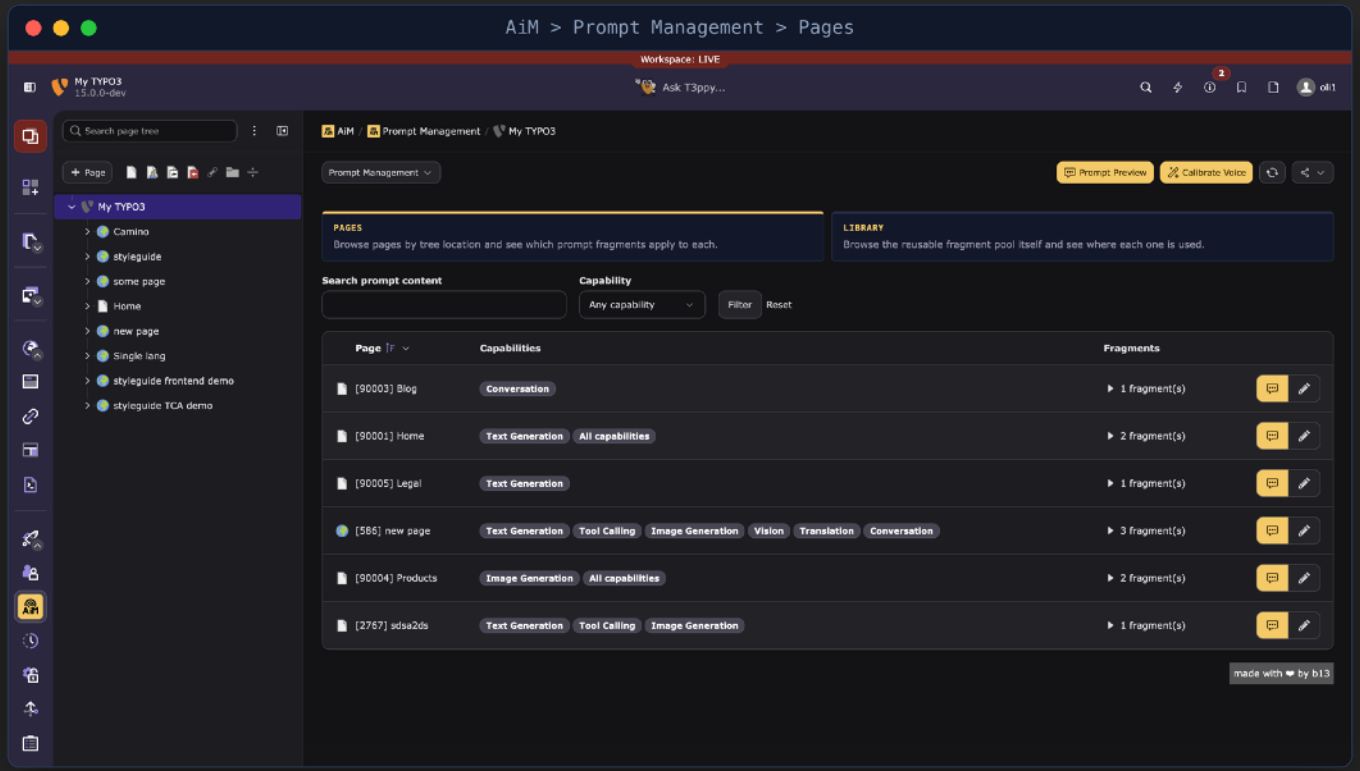


Prompt Management

Individualize your AI experience

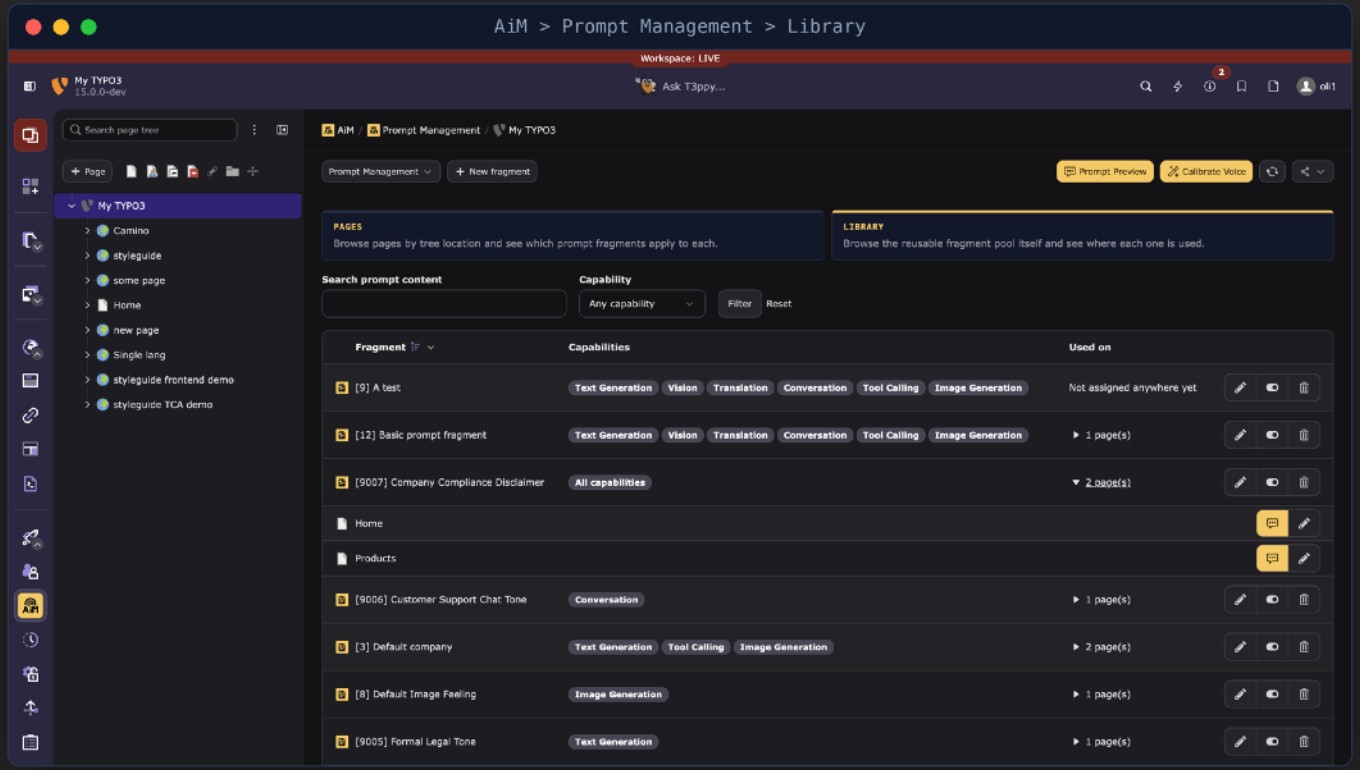
AiM - Prompt Management

The site's own tone of voice



AiM - Prompt Management

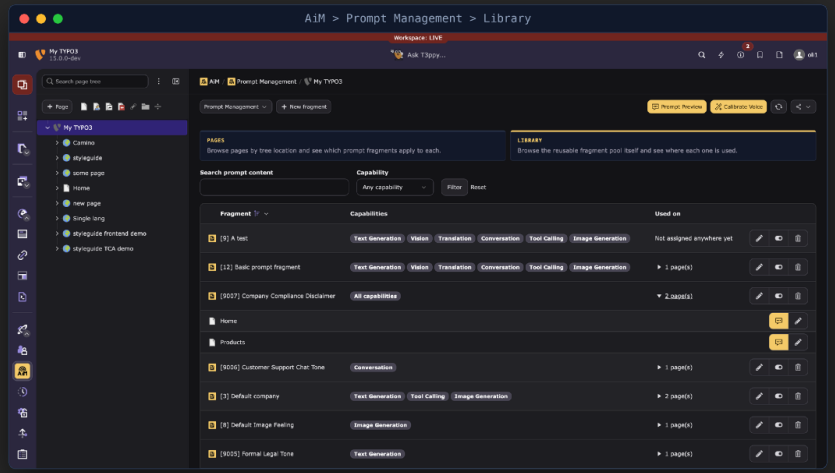
The site's own tone of voice



AiM - Prompt Management

The site's own tone of voice

- Written once, assigned to any number of pages.
- Inherited down the page tree. (Opt-Out possible)
- Scoped to the capabilities.
- Previewed with its token count.



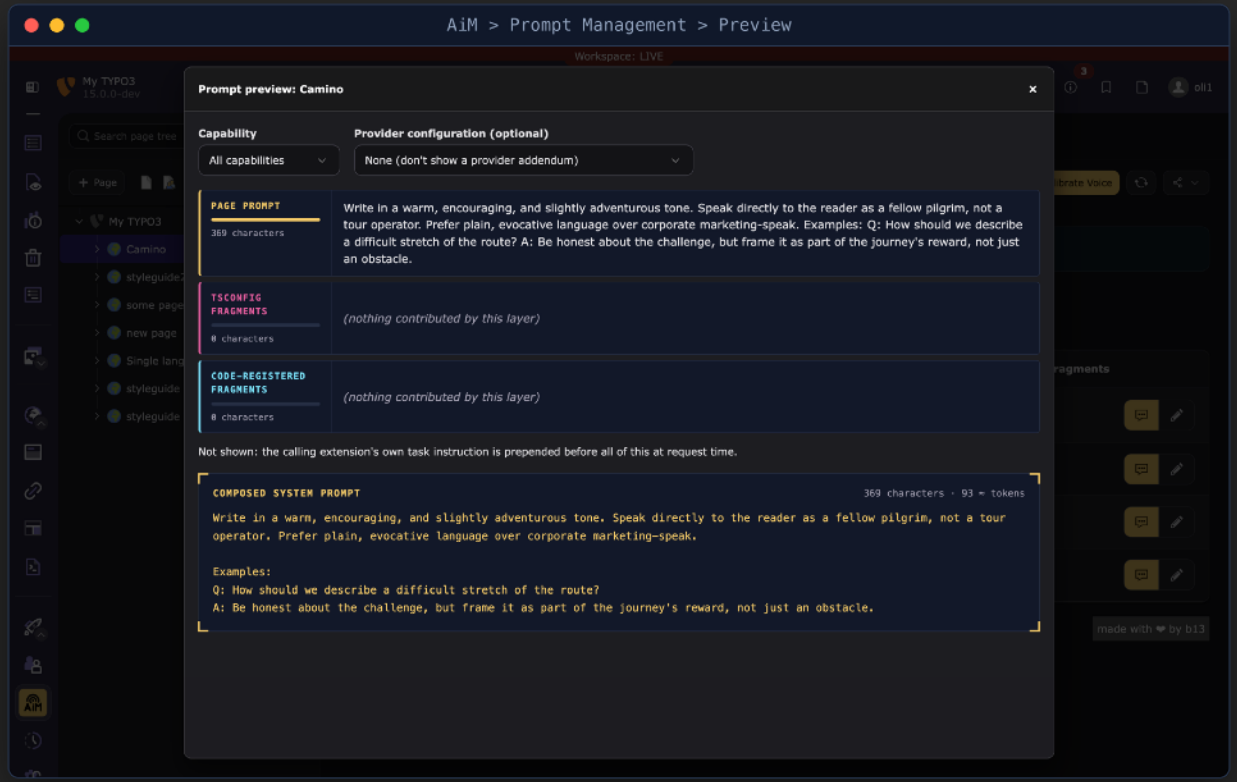
1	**the caller's own prompt**	the extension, in code
2	**the page tree**	fragments, inherited down the tree
3	**the backend user or group**	User/Group TSconfig
4	**registered in code**	an organisation-wide policy
5	**the provider's own addendum**	added once the configuration is final

layers 1-4 at priority 80, above SmartRouting at 75, so the complexity score sees the real prompt.

layer 5 at priority 10, when the configuration is final.

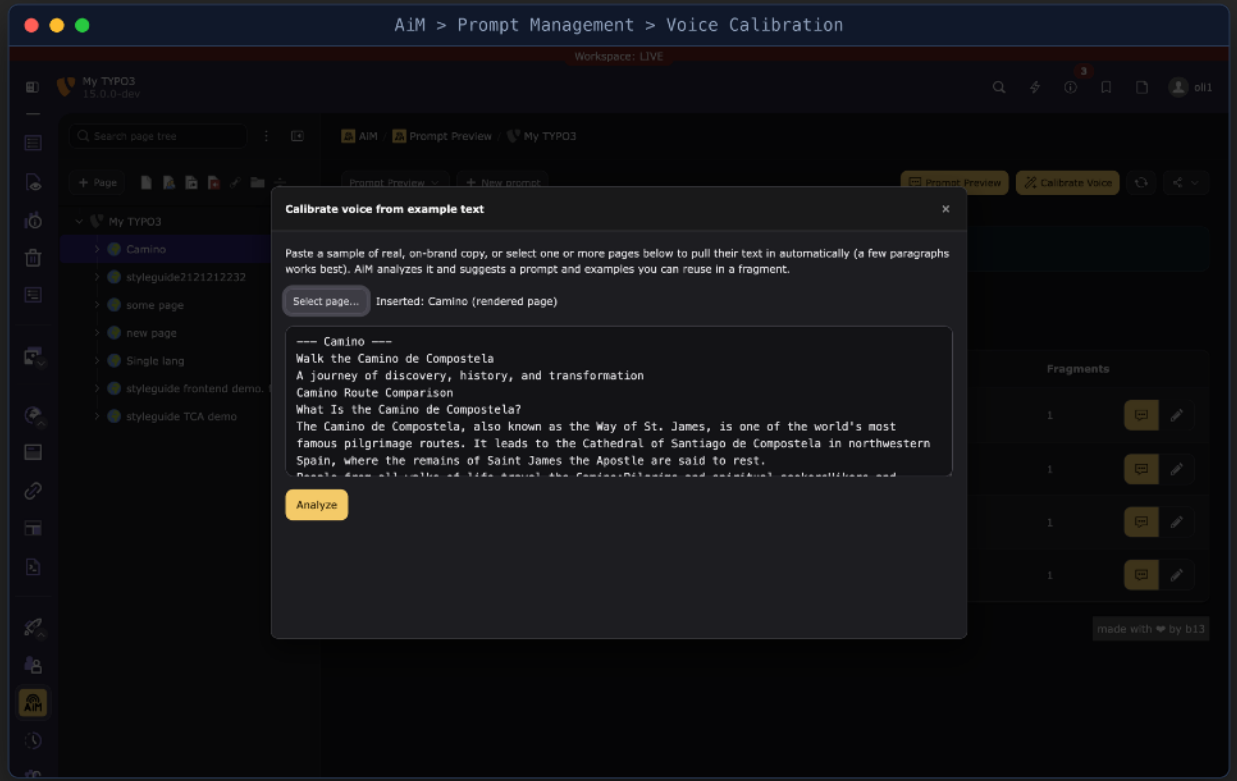
AiM - Prompt Management

The site's own tone of voice

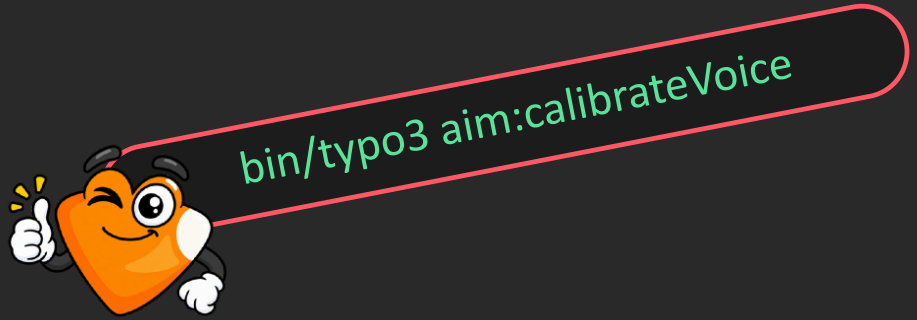


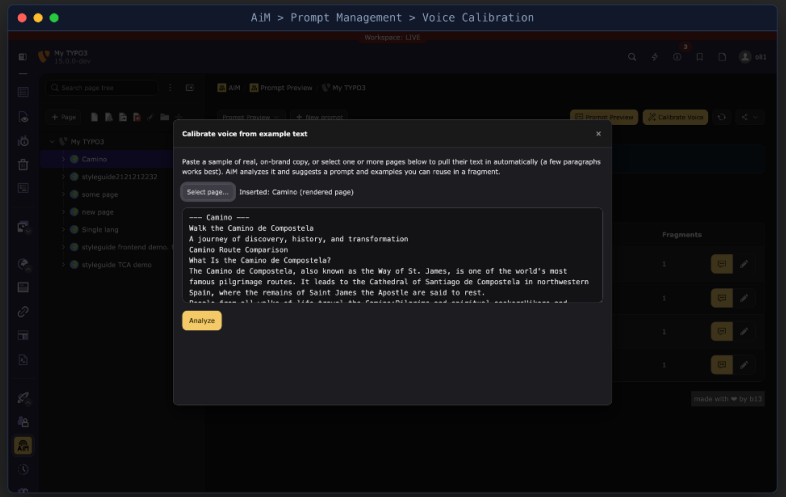
AiM - Prompt Management

Voice calibration



Voice calibration





AIM Takeaways

- Provider Agnostic API
- Request logging & footprint
- Smart Routing & Model Switching
- LLM Grading
- Middleware Pipeline
- Governance
- Privacy Levels
- Tone-Of-Voice & Voice Calibration
- Encryption
- Auto Discovery

Installation

AIM

Installation

Three steps, and none of them is a key file

```
// 1. AiM itself. Zero AI provider dependencies.
composer require b13/aim

// 2. The bridges you actually want.
composer require symfony/ai-openai-platform
composer require symfony/ai-ollama-platform

// 3. Admin Tools > AiM > Providers > New configuration
```

AIM

Requirements

What you need to run it

- **TYPO3 12.4, 13.4 and 14**
- **PHP 8.1+**
- **ext-sodium**
- **No AI provider dependencies** · *Use Symfony AI bridge, or write a provider*

AiM is in alpha
we'd love your **feedback!**

Thank you!





github.com/b13/aim