Filter by keyword

Forge LLMs API

ON THIS PAGE

Forge LLMs is available through Forge's Early Access Program (EAP). EAP grants selected users early testing access for feedback; APIs and features in EAP are experimental, unsupported, subject to change without notice, and not recommended for production — sign up here ↗ to participate.

For more details, see Forge EAP, Preview, and GA.

Forge LLMs lets your Forge app call Atlassian-hosted large language models (LLMs) to add secure AI features without leaving the Atlassian platform. Apps using this API are badged as **Runs on Atlassian**, indicating they leverage Atlassian's security, compliance, and scalability. The API provides optimized, governed access to supported models so you can focus on creating innovative AI experiences while Atlassian handles model integration and infrastructure.

Manifest reference for LLM module

See the LLM module reference for details on the `llm` module for your `manifest.yml`.

Important:

Versioning

The `llm` module is required to enable Forge LLMs. When you add the `llm` module to an app's `manifest.yml`, it triggers a major version upgrade and requires administrators of existing installations to review and approve the update.

EAP limitations

⚠ During the EAP, you are blocked from deploying your app to the `production` and `staging` environments. You cannot distribute your app or list it on the Atlassian Marketplace.

Tutorials and example apps

Create a basic LLM web trigger app

Learn how to build a simple Forge app that integrates with the 'llm' module using a web trigger.

Explore tutorial

Create an agentic LLM web trigger app

Learn how to leverage 'tools' usage to build an agentic Forge app.

Explore tutorial

Handling long-running LLM processes with Forge Realtime

Guide to managing long-running LLM calls using asynchronous functions and Forge Realtime for updates.

Explore tutorial

Example apps

We published some sample apps to help demonstrate the basics of using Forge LLMs.

Explore sample apps

Node.js SDK

The @forge/llm SDK ↗ gives you a lightweight, purpose-built client for invoking Atlassian-hosted LLMs directly from Forge runtime functions.

For runnable examples (tool wiring, retries, error handling), see the Forge LLMs tutorials and example apps section above.

Method signature

Please refer to the request and the response schemas.

```
list() => Promise<ModelListResponse>
chat(Prompt) => Promise<LlmResponse>
stream(Prompt) => Promise<StreamResponse>
```

Example usage

Using chat

```
import { chat } from '@forge/llm';
try {
  const response = await chat({
    model: 'claude-3-7-sonnet-20250219',
    messages: [
      {
        role: 'user', content: 'Write a short poem about Forge LLMs.'
      }
    ],
  });

  console.log("#### LLM response:", JSON.stringify(response));
} catch (err) {
  console.error('#### LLM request failed:', { error:  err.context?.respo
  throw err;
}
```

Using stream

```
import { stream } from '@forge/llm';
try {
  const response = await stream({
    model: 'claude-3-7-sonnet-20250219',
    messages: [
      {
        role: 'user', content: 'Write a short poem about Forge LLMs.'
      }
    ],
  });

```

```
15
16    response.close();
17
18  } catch (err) {
19    console.error('#### LLM request failed:', { error:  err.context?.respo
20    throw err;
21  }
```

Module validation

The SDK requires the `llm` module to be defined in your `manifest.yml`. If the SDK is used without declaring this module, linting will fail with an error like:

```
1  Error: LLM package is used but 'llm' module is not defined in the manifes
```

The SDK can automatically fix your manifest. After linting, the manifest will include:

Example of corrected manifest.yml:

```
1  modules:
2    llm:
3      - key: llm-app
4        model:
5          - claude
```

Please refer to the LLM module reference for details on how to define the module.

Request and response schemas

Please consult the @forge/llm ↗ package for the most up-to-date request and response schema definitions.

Request

```
1  interface Prompt {
2    model: string;
3    messages: {
4      role: "system" | "user" | "assistant" | "tool";
5      content: string | { type: 'text'; text: string; }[];
6    }[];
7    max_completion_tokens?: number;
8    temperature?: number;
9    top_p?: number;
10   tools?: {
```

```
      description: string;
      parameters: object;
    };
  }[];
  tool_choice?: "auto" | "none" | "required" | { type: "function"; funct
}
```

Response

```
interface LlmResponse {
  choices: {
    finish_reason: string;
    index?: number;
    message: {
      content: string | { type: "text"; text: string; }[];
      role: "assistant";
      tool_calls?: {
        id: string;
        type: "function";
        index: number;
        function: { name: string; arguments: object; };
      }[];
    };
  }[];
  usage?: { input_token?: number; output_token?: number; total_token?:
}

interface StreamResponse extends AsyncIterable<LlmResponse> {
  close(): Promise<void> | undefined;
}

interface ModelListResponse {
  models: {
    model: string;
```

Important validation rules

The following request validation rules apply to specific models:

Rule	Models
When adjusting sampling parameters, modify either `temperature` or `top_p`. Do not modify both at the same time.	`claude-haiku-4-5-20251001`, `claude-sonnet-4-5-20250929`

choose the model per request, allowing you to balance latency, capability, and cost for each use case.

Supported models

You can use the `list` method from the SDK to dynamically fetch the list of supported models and their respective status.

Forge

`claude-3-5-haiku-20241022`	Haiku	Claude	DEPRECATED	February 2026
`claude-haiku-4-5-20251001`	Haiku	Claude	ACTIVE	
`claude-3-7-sonnet-20250219`	Sonnet	Claude	DEPRECATED	February 2026
`claude-sonnet-4-20250514`	Sonnet	Claude	ACTIVE	
`claude-sonnet-4-5-20250929`	Sonnet	Claude	ACTIVE	
`claude-opus-4-20250514`	Opus	Claude	DEPRECATED	February 2026
`claude-opus-4-1-20250805`	Opus	Claude	ACTIVE	
`claude-opus-4-5-20251101`	Opus	Claude	ACTIVE	

As AI models evolve quickly, check regularly for deprecated status and associated EOL dates so your apps do not break.

Claude - Opus

- Most capable (best for complex, deep reasoning tasks)
- Slowest (higher latency due to depth)
- Highest cost

Claude - Sonnet

- Balanced capability

Claude - Haiku

- Fast and efficient (best for lightweight or high-volume tasks)
- Lowest cost

AI models evolve quickly, so specific versions may change before launch. Initially only text input/output is supported; multimodal support may be considered later.

Admin experience

Administrators will be informed (Marketplace listing and during installation) when an app uses Forge LLMs. Adding Forge LLMs—or a new model family—to an existing app triggers a major version upgrade requiring admin approval.

Usage tracking

The Forge LLM API reports usage data per request (the number of input and output tokens consumed) in the API response.

During the EAP, usage tracking is limited to the data provided in the API response.

Pricing

LLMs will become a paid Forge feature soon. Usage (token input/output volume) will appear in the developer console under usage and costs. Specific pricing will be published before preview.

Responsible AI

Requests to Forge LLMs undergo the same moderation checks as Atlassian first-party AI and Rovo features. High-risk messages (per the Acceptable Use Policy) are blocked.