ChatVPet Process Setup Tutorial

This tutorial will guide you on how to correctly fill in API information, Embedding information, and other advanced parameters in the **VPet.Plugin.ChatVPet** settings window（`winSetting`）enabling ChatVPet to properly call Large Language Models for dialogue.

Table of Contents

---

1. Opening the Settings Window

Right-click the pet on the VPet main interface → **ChatVPet Settings** to open the settings window.

---

2. Basic Settings — API Configuration

The settings window stays on the **Basic Settings** tab by default，you must fill in the following fields to make ChatVPet function properly:

ChatVPet uses OpenAI-compatible API calling methods. Most AI service providers and local AI deployers support the OpenAI API call format.

Field	Description
API URL	Chat interface address, recommended to end with /v1
API Key	Secret key used for authentication
Web 代理	Optional. Enter the HTTP/SOCKS proxy address. If accessing OpenAI or similar services directly from China, please ensure you fill this in or use an API relay.
Model	Name of the language model to use; can be selected from the dropdown or entered manually.
Initial Text	System Prompt. It will not be forgotten; more words increase the cost.

Temperature	Response randomness (0.1–2). Smaller values are more stable, larger values are more random. Default is 0.6.
Max Cost per Request	Maximum number of Tokens allowed for a single request (set in “Other Settings,” see Section 5).

Tip：The`{Name}` placeholder in the Initial Text will be replaced with the pet's actual name. You can use it in the prompt, for example:
You are a desktop pet named`{Name}`。

Note: The model must support the ToolCall (Function Calling) feature

Currently, models such as OpenAI, DeepSeek, Claude, Gemini, Gemma, Qwen, Glm, and gpt-oss are known to support tool calls. It is recommended to check the relevant documentation before use.

This tutorial uses two API providers as examples; you may also use other APIs or local models (such as LM Studio).

2.1 Using OpenAI API

1. Go to the OpenAI Platform to register an account and add credits.
2. Navigate to the **API Keys** page and click **Create new secret key** to generate a key，starts with `sk-` 。
3. Fill in the settings window：
   - **API URL**：`https://api.openai.com/v1/`
   - **API Key**：Paste your `sk-xxxxxxxx` key
   - **Model**：Recommended `gpt-4o-mini`，cost-effective，or `gpt-4o`

If accessing from Mainland China, you must fill in a proxy address in Web Proxy or use a relay service.

Common Models Reference

Model	Features
`gpt-5.4-nano`	Cheap
`gpt-5.4-mini`	Expensive
`gpt-5.4`	Very expensive

2.2 Using DeepSeek API

DeepSeek provides an API that is fully compatible with the OpenAI interface, performing excellently in Chinese dialogue scenarios with more competitive pricing.

1. Go to the DeepSeek Open Platform to register an account and add credits.
2. Navigate to the **API Keys** 页面，page, create and copy your API Key (starts with `sk-`).
3. Fill in the settings window：
   - **API URL**：`https://api.deepseek.com/v1`
   - **API Key**：Paste your DeepSeek API Key
   - **Model**：`deepseek-chat`，General，or `deepseek-reasoner`，Reasoning。

> DeepSeek API servers are directly accessible in mainland China; the**Web 代理** field can usually be left blank.

Common Models Reference

Model	Features
`deepseek-chat`	General dialogue, fast speed, strong Chinese language capability
`deepseek-reasoner`	Chain-of-thought reasoning, suitable for logic-intensive tasks

---

2.3 Using Locally Deployed Models

In addition to using cloud APIs, you can also deploy large language models locally via **LM Studio** or **llama.cpp** to achieve offline operation, data privacy protection, and cost control.

Option 1: LM Studio (GUI)

LM Studio is a cross-platform desktop application (supporting Windows, macOS, and Linux) that provides an intuitive graphical interface to easily download, load, and manage various local models in GGUF format.

Deployment Steps:

1. **Download and Install LM Studio**
   Visit https://lmstudio.ai/ to download the installer for your operating system and complete the installation.
2. **Download a Model**
   Open LM Studio, go to the **Model Search** tab, then search for and download a model that supports Tool Call.**It is recommended to choose models with 8B parameters or more**（例如 `qwen3.5-35b-a3b`、`gemma-4-26b-a4b`、`glm-4.7-flash`）.Smaller models are more prone to formatting errors.
3. **Start Local Server**
   - Click the Developer tab in the left sidebar.
   - Select **Load Model** , choose your downloaded model, and click **Start Server** to launch the service.
   - The default API address is `http://localhost:1234/v1`。
4. **Configure ChatVPet**
   Fill in the settings window with the following information：
   - **API URL**：`http://localhost:1234/v1`
   - **API Key**：Any non-empty string (e.g., `local`); LM Studio's local service does not validate the key.
   - **Model**：Enter the name of the model currently loaded in LM Studio (this can be found on the Server interface, (e.g., `glm-4.7-flash`).
   - Fill in the remaining fields (such as Temperature, Proxy, etc.) according to your actual needs.

Note：LM Studio's `/v1/chat/completions` endpoint is fully compatible with the OpenAI API format and natively supports the **Tool Call(Function Calling)** feature. As long as the model itself possesses

Tool Call capabilities, ChatVPet will be able to call them normally.

Option 2: llama.cpp (Command Line, Superior Performance)

llama.cpp is a high-performance C/C++ inference framework that supports CPU and GPU acceleration (CUDA, Metal, Vulkan, etc.). It is ideal for users seeking peak performance or needing server-side deployment. Many popular local runners (such as Ollama and LM Studio) are built on this framework.

Deployment Steps:

1. **Obtain llama.cpp**
   Install pre-built version of llama.cpp
2. **Download a Model**
   Download model files in GGUF format from platforms like Hugging Face. (e.g., `qwen3.5-35b-a3b`、`gemma-4-26b-a4b`、`glm-4.7-flash`). Smaller models are more prone to formatting errors.
   3. **Start the API Server**
   Use the built-in `llama-server` to launch an OpenAI-compatible API service:
   `./llama-server -m /path/to/model.gguf --host 0.0.0.0 --port 8080 -c 4096 --jinja`
   - `-m`Path to the model file
   - `--host` / `--port`: Listening address and port
   - `-c`: Context length (should match the model's specifications)
   - `--jinja`、**Enable Jinja template parsing; this is a critical parameter for Tool Call to function correctly**
   4. **Configure ChatVPet**
   - **API URL**、`http://localhost:8080/v1`
   - **API Key**、Any non-empty string
   - **Model**、Enter the model name (you can check this by visiting `http://localhost:8080/v1/models`)

Tip: The llama.cpp API service is fully compatible with the OpenAI format and natively supports Tool Call. If GPU acceleration is required, please enable the corresponding backend during compilation based on your hardware (CUDA for NVIDIA, Metal for Apple Silicon).

Important Reminder on Model Selection for Self-Deployment: Small Models May Lead to Tool Call Formatting Errors

Not all local models perfectly support Tool Call..Many local models are not yet intelligent enough to accurately identify when a tool needs to be invoked, nor can they consistently output invocation commands in the specific JSON format required by ChatVPet.

- **Models below 9B are prone to the following issues:**
  - Outputting non-standard JSON tool call descriptions instead of the `tool_calls` field expected by ChatVPet.
  - Completely ignoring tool definitions and responding directly in plain text.

- Outputting incomplete JSON structures or missing required fields (such as `name` or `arguments`), resulting in parsing failures.
- Significant drops in tool call success rates due to precision loss in quantized models.

Therefore, it is recommended:

When deploying locally, **prioritize models with 8B parameters or more**, such as `Qwen2.5-7B-Instruct`,`Llama-3.1-8B-Instruct`,or `Mistral-7B-Instruct`. These models have undergone extensive verification for Tool Call support and are relatively mature.

---

3. 3. Basic Settings — Embedding Configuration

3.1 What is Embedding?

ChatVPet utilizes **Vector Embedding** technology to build a semantic index for your knowledge base, tool library, and chat history. This allows the system to intelligently retrieve the most relevant content for each conversation rather than cramming everything into the prompt—significantly reducing Token consumption.

字段	说明
Embedding URL	The address of the Embedding endpoint; leave blank to use the primary API URL.
Embedding Key	The secret key for the Embedding endpoint; leave blank to automatically use the primary API Key.
Embedding Model	The Embedding model to be used; defaults to `text-embedding-3-small` .

Note：After changing the Embedding model or API, please go to the **Database Preview** tab and click **Clear Vector Cache** to ensure old cached data is invalidated and new vectors are regenerated.

--

3.2 Using OpenAI Embedding

If the main API has already been configured with OpenAI information, all Embedding fields can be **left blank**. The application will automatically inherit the URL and Key from the main API and use the default model, `text-embedding-3-small`.

If you need to specify them separately:

- **Embedding URL**：`https://api.openai.com/v1`
- **Embedding Key**：Same as the primary API Key
- **Embedding Model**：`text-embedding-3-small`或 `text-embedding-3-large`

--

3.3 Using the bge-m3 Model Provided by LBGAME

Since this specific Embedding model is relatively niche and there happened to be an idle GPU server available, a bge-m3 model has been deployed on it.

Please note that this service may change in the future; long-term availability is not guaranteed. - **Embedding URL**：`https://lolisbr.exlb.net/v1` - **Embedding Key**：`sk-Lolis-provides-everyone-with-free-embeddings-Lolis-is-great` - **Embedding Model**：`text-embedding-bge-m3`

> Note: The server does not collect user information; however, to save performance overhead, it will cache text-to-vector mappings to avoid redundant computations.

3.4 Using Local Deployment

Refer to **2.3**; simply download an additional **BAAI/bge-m3** model.

---

4. Basic Settings — Other Parameters

Field	Description
Total Spent	Displays the total number of Tokens consumed to date; read-only.
Show Tokens	Whether to display the number of Tokens consumed for each turn in the chat history.
Submit Content	Whether to submit chat logs to LBGAME to help improve ChatVPet.

---

5. Other Settings — Voice Input and Advanced Parameters

Switch to the **Other Settings** tab to configure voice input and various limits.

Voice Input（Azure AI Speech）

ChatVPet supports voice input through Azure AI services. You must first create a **Speech** resource in the Azure portal .

Field	Description
Enable Voice Input	Toggle switch; when enabled, a microphone button will appear in the chat interface.
Speech Key	The subscription key for your Azure Speech resource.
Speech Region	The region where your Azure resource is located (e.g., `eastasia`、`westus`.

Speech Language	The language code for recognition (e.g., zh-CN、en-US、see supported list)

Chat and Memory Parameters

Field	Default	Description
Max Chat History	20	The maximum number of historical messages injected into the context for a single turn.
Max Tool Library	10	The maximum number of tools retrieved per conversation; higher values increase Token consumption.
Max Knowledge Base	10	The maximum number of knowledge base entries retrieved per conversation; higher values increase Token consumption.
Max Output Tokens	4000	The maximum number of Tokens allowed for a single response; it is recommended not to exceed the model's limit.
Max Rounds	5	The maximum number of recursive cycles allowed for Tool Calls (to prevent infinite loops).

History Compression and Diary Parameters

ChatVPet compresses excessively long history into diaries to save Tokens and implement long-term memory.

Field	Default	Description
Compression Trigger	24	Automatically triggers compression when chat history exceeds this number; set to 0 to disable.
Retention Count	10	The number of recent dialogue turns to keep intact (not summarized) during compression.
Diary Decay Rate	0.03	The rate at which the weight of un-retrieved diaries decreases after each turn (0.03 = 3%).
Max Diary Injection	10	The maximum number of diary entries injected into the system prompt for each conversation.

---

6. Knowledge Base Settings

Switch to the **Knowledge Base Settings** tab to add custom knowledge line by line in the text box:

```
{name}likes eating strawberry cake
{hostname}likes eating strawberries.
{hostname}dislikes working because the boss always makes them work overtime.
```

Rules - One piece of information per line

- Knowledge is not directly loaded into the prompt in its entirety; instead, it is dynamically retrieved by the embedding algorithm based on relevance.
- You can describe the Pet's persona, daily preferences, specialized domain knowledge, and more.
- Use `{name}` to refer to the Pet's name and `{hostname}` to refer to the player's name.
- Changes will take effect after restarting the game.

7. 7. Database Preview

Switch to the **Database Preview** tab to view all current knowledge base entries, tool libraries, and chat records.

- **Search**：Enter keywords and click **Search** to perform a precise text-match lookup.
- **Vector Search**：Enter natural language and click **Vector Search** to find results based on semantic similarity, which typically yields better results.
- **Clear Vector Cache**：After changing the Embedding API or model, click this button to clear old vectors and force a recalculation.
- **Delete Chat History**：In the “Chat History” sub-tab, right-click an entry and select Delete to remove specific historical conversations.

8. FAQ

Q: ChatVPet doesn't respond after I save the API information?
A: Please check:

1. Whether the API URL is correct.
2. Whether the API Key is valid and not expired.
3. If using OpenAI, a Web Proxy needs to be configured in Mainland China.
4. Check if the account balance is sufficient.

Q: Is configuring Embedding mandatory?
A: Yes, it is mandatory. If you are unsure what to fill in, please refer to **3.3**

Q: Search results worsened after changing the Embedding model?
A: Vectors generated by different models are incompatible. Please click **Clear Vector Cache** on the **Database Preview** page and wait for the vectors to be regenerated during the next conversation.

Q: How can I reduce Token consumption?
A: You can:

- Decrease the values for **Max Chat History**, **Max Tool Library** and **Max Knowledge Base**.
- Shorten the **Initialization Text**.
- Select a model with lower pricing rates.