

openKylin 系统智能体开发者手册

V0.9.1

openKylin AgentOS SIG

国防科技大学、哈尔滨工业大学（深圳）、麒麟软件有限公司

2026 年 06 月 25 日

目 录

1. 接入说明与路径选择.....	1
1.1 概述.....	1
1.2 两种接入模式.....	1
1.3 路径选择与能力边界.....	3
2. 基于 openKylin 系统智能体的能力扩展.....	4
2.1 openKylin 系统智能体能力扩展机制.....	4
2.2 Skill 设计与编写.....	5
2.3 Skill 接入、调用与验收.....	8
3. 直接接入运行时统一模型推理服务.....	10
3.1 服务准备、模型管理与标准接入流程.....	10
3.2 统一推理与 OpenAI 兼容调用.....	12
3.3 Embedding、语音服务与专用 Worker 调用.....	15
3.4 接口说明.....	17
4. 环境与依赖安装.....	20
4.1 安装范围.....	20
4.2 openKylin 系统智能体安装.....	20
4.3 openKylin 系统智能体源码开发依赖.....	20
4.4 运行时统一模型推理服务部署与开发依赖.....	21
5. 常见问题.....	22
6. 开源仓库.....	25
6.1 AgentOS SIG 介绍.....	25
6.2 开源代码地址.....	25

1. 接入说明与路径选择

1.1 概述

本手册面向应用开发者、智能体开发者、平台集成开发者和模型服务开发者，说明如何接入 openKylin 智能能力，并根据业务需求选择合适的接入层级。

本手册覆盖的核心接入能力包括 openKylin 系统智能体和运行时统一模型推理服务：

- **openKylin 系统智能体**：面向复杂任务执行，提供任务理解、任务拆解、Skill 调用、文件处理、系统操作、跨应用协同、CUA 桌面操作、过程追踪等能力；
- **运行时统一模型推理服务**：面向模型服务调用，提供模型注册、模型加载、Runtime 调度、统一推理、Worker 转发、日志和过程观测等能力。

openKylin 系统智能体用于处理面向用户或业务的综合任务。它可以理解自然语言任务，结合模型、Skill、文件处理、系统操作和跨应用协同能力完成执行。对于具备 API 的系统或应用功能，智能体可通过接口直接调用；对于未提供 API 的桌面应用，可通过 CUA Skill 进行界面识别和键鼠模拟操作。

运行时统一模型推理服务用于提供底层模型能力。它负责模型注册、模型加载、Runtime 调度、推理请求转发和运行状态观测，并将请求转发至 llama.cpp 或自定义 Worker。

1.2 两种接入模式

1.2.1 基于 openKylin 系统智能体扩展能力

openKylin 系统智能体作为统一入口，接收用户或业务系统提交的请求，并根据任务内容调用模型、内置 Skill 或自定义 Skill 完成处理。

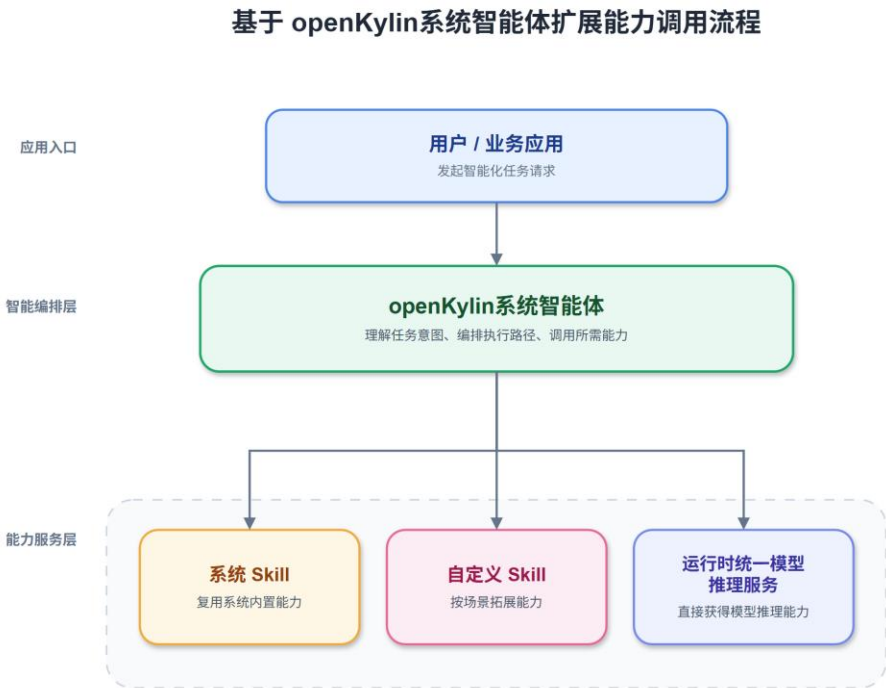


图 1 基于 openKylin 系统智能体扩展能力调用流程图

这种方式适合需要完成完整任务闭环的场景，例如：

- 文件处理、资料整理和内容生成；
- 系统信息查询、系统配置和运维辅助；
- 调用业务系统接口完成查询、填写或提交；
- 多个桌面应用之间的信息获取和操作协同；
- 通过 CUA Skill 操作没有开放 API 的桌面应用。

新增业务能力时，可将能力开发为 Skill 并接入 openKylin 系统智能体。Skill 用于描述一项明确的业务能力，包括其适用任务、输入信息、执行方式和结果形式。智能体结合任务内容、已加载 Skill 的元数据和能力说明选择可用能力。Skill 名称和描述应准确表达适用场景，具体选择策略以智能体运行时实现为准。

例如，可以将“批量整理指定目录中的文档”“查询设备状态”“导出业务系统报表”“通过桌面应用创建会议”等能力分别封装为独立 Skill。

现有 openKylin 系统智能体支持 Skill 的创建、导入、编辑、预览和卸载，Skill 可通过 SKILL.md、技能目录或压缩包进行管理。

这一方式的目标是让业务能力以插件形式持续扩展，同时保持 openKylin 系统智能体作为统一的任务入口和执行入口。

1.2.2 直接接入运行时统一模型推理服务

运行时统一模型推理服务可以独立使用，不依赖 openKylin 系统智能体。

统一模型推理服务调用流程

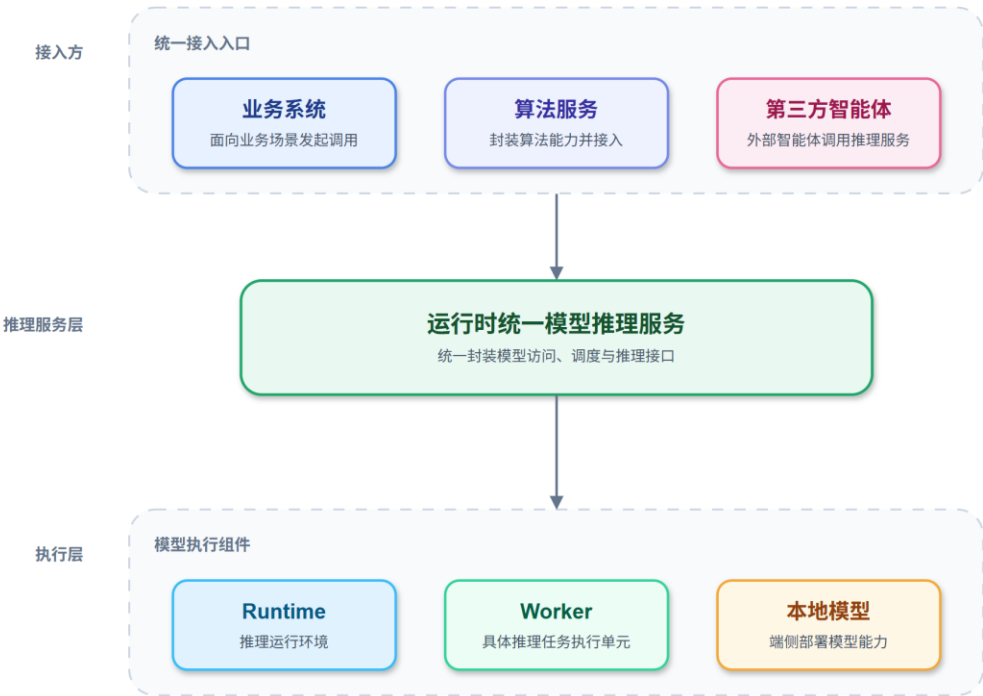


图 2 统一模型推理服务调用流程图

这种方式适合只需要模型或算法能力的场景，例如：

- 文本生成、对话问答和内容摘要；
- 知识库问答和 RAG 应用；
- Embedding 向量生成与检索；
- ASR 语音识别和音频处理；
- 图文等多模态推理；
- 接入自定义算法模型或 Worker；
- 通过 OpenAI 兼容接口接入已有应用。

运行时统一模型推理服务提供模型注册、模型查询、模型加载、统一推理、OpenAI 兼容 Chat、文件上传和运行状态观测等能力。业务系统通过 HTTP 接口调用模型，并根据业务需要自行组织提示词、上下文、任务流程和结果处理。

该方式更适合模型服务、算法服务和后端应用集成。模型选择、推理参数、调用频率和结果处理均可由接入方自行控制。

1.3 路径选择与能力边界

两条路径的主要区别如下：

对比项	基于 openKylin 系统智能体扩展能力	直接接入运行时统一模型推理服务
接入入口	openKylin 系统智能体	运行时统一模型推理服务
主要用途	完成综合任务和系统级操作	调用模型和算法能力
扩展方式	开发并接入 Skill	注册模型、接入 Worker 或扩展 Runtime
任务拆解	由 openKylin 系统智能体处理	由业务系统自行处理
Skill 调用	由 openKylin 系统智能体识别和调用	不涉及
系统操作与 CUA	可支持	不涉及
上下文与任务流程	由智能体结合任务处理	由接入方自行维护
典型场景	桌面助手、文件处理、系统协同、应用自动化	RAG、对话服务、语音服务、模型推理平台

选择时可参考以下原则：

需要处理文件、调用系统能力、操作桌面应用、执行跨应用流程时，使用 openKylin 系统智能体，并通过 Skill 扩展新增能力。

只需要调用模型、生成向量、识别语音或运行专用算法时，直接接入运行时统一模型推理服务。

2. 基于 openKylin 系统智能体的能力扩展

2.1 openKylin 系统智能体能力扩展机制

openKylin 系统智能体是任务处理和能力调用的统一入口。用户或业务应用提交任务后，openKylin 系统智能体负责理解任务内容、判断所需能力、选择合适的 Skill，并组织任务执行过程。

Skill 负责完成具体业务动作，例如调用业务接口、执行本地脚本、处理文件资源或操作桌面应用。运行时统一模型推理服务为 openKylin 系统智能体提供模型推理能力，用于任务理解、内容生成、结果分析等处理过程。对于没有开放 API 的桌面应用，可通过 CUA Skill 完成界面识别和键鼠模拟操作。

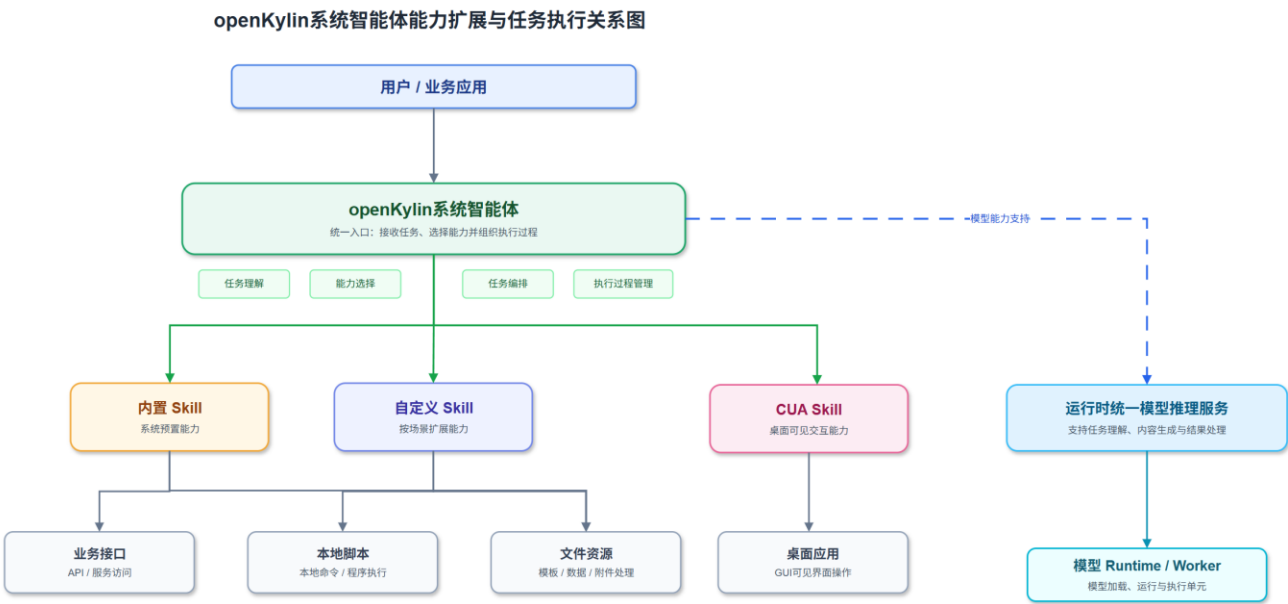


图 3 openKylin 系统智能体能力扩展与任务执行关系图

openKylin 系统智能体的能力扩展以 Skill 为基本方式。新增业务需求时，先判断现有 Skill 是否已经具备相应能力；现有 Skill 无法满足时，再将新增能力开发为自定义 Skill。

通常适合开发为 Skill 的能力包括以下几类：

能力类型	适合封装的内容	典型示例
文件处理能力	文件读取、整理、转换、提取、生成	批量整理文档、提取表格数据、生成报告
系统操作能力	系统信息查询、配置调整、状态检查	查询设备状态、导出系统日志、修改指定配置
业务接口能力	调用已有业务系统接口完成查询或提交	查询业务数据、生成业务报表、提交审批信息
本地工具能力	调用本地脚本、命令或软件工具	调用脚本处理文件、执行固定检查任务
桌面应用能力	操作已安装的桌面应用	创建会议、填写表单、导出应用数据
CUA 能力	操作没有开放 API 的可视化应用	在指定应用中导航、查询、填写和提交信息

开发 Skill 时，应按照业务动作划分能力边界。一个 Skill 应完成一类相对完整的任务，不宜将多个无关能力混合在同一个 Skill 中。

例如：

- “批量整理指定目录中的文件”可以作为一个 Skill；
- “根据模板生成周报”可以作为一个 Skill；
- “查询业务系统中的订单状态”可以作为一个 Skill；
- “通过桌面应用创建会议”可以作为一个 Skill。

“文件整理、会议创建、业务查询、文档生成”不应全部放在同一个“办公助手 Skill”中。能力范围过大时，系统智能体难以准确判断何时调用该 Skill，开发者也难以定位执行异常。

2.2 Skill 设计与编写

一个 Skill 通常由 SKILL.md 和可选的脚本、模板、配置文件或资源文件组成。

```
document-organizer/  
├── SKILL.md  
├── scripts/  
│   └── organize_documents.py  
├── templates/  
│   └── organize-rules.json  
└── examples/  
    └── usage.md
```

其中：

- SKILL.md 用于描述 Skill 名称、能力范围、适用条件、执行方式和结果形式；
- scripts/用于存放执行脚本，可根据实际需要使用；
- templates/用于存放规则文件、模板文件或静态资源；
- examples/用于存放典型任务示例和测试样例。
- Skill 目录不要求固定包含所有文件。只有在 Skill 确实依赖脚本、模板或其他资源时，再增加对应目录。

2.2.1 Skill 名称和描述

Skill 名称用于标识能力，描述用于帮助 openKylin 系统智能体判断何时调用该 Skill。

名称应直接体现业务能力，例如：

- document-organizer
- system-info-query
- meeting-scheduler
- business-report-export

描述应同时说明三件事：

1. 该 Skill 可以完成什么任务；

2. 适用于什么用户请求；
3. 会产生什么结果或操作。

不建议使用“万能办公助手”“通用工具”“帮助用户处理任务”等宽泛描述。这类描述难以帮助系统智能体区分不同 Skill。

2.2.2 Skill 应说明的内容

SKILL.md 建议至少包括以下内容：

内容	说明
名称和版本	标识 Skill 名称及当前版本。
能力描述	说明 Skill 解决的具体问题和适用任务。
使用条件	说明应用、文件、账号、目录或权限等前置条件。
输入要求	说明任务执行需要的路径、参数、文件或业务数据。
执行步骤	说明处理流程，以及调用脚本、接口或桌面应用的方式。
输出结果	说明返回内容、生成文件、状态信息和失败原因。
权限要求	说明是否读取或写入文件、访问网络、调用接口、操作桌面应用。
异常处理	说明输入缺失、资源不可用、执行失败或用户取消时的处理方式。
调用示例	提供典型用户任务，帮助验证 Skill 的触发条件。

2.2.3 Skill 编写示例

以下以“文档整理 Skill”为例。该 Skill 用于将指定目录中的文件按类型或日期分类，并输出处理结果。

```

---
name: document-organizer
description: 对指定目录中的文档、表格、图片和 PDF 文件按文件类型、创建时间或用户指定规则进行分类整理，适用于批量归档、生成整理清单和检查无法处理文件的任务。
version: 1.0.0
---

# 文档整理 Skill

## 适用场景

当用户需要整理指定目录中的文件，并按文件类型、日期或指定分类规则归档时使用。

不用于删除文件、修改文件或处理用户未授权访问的目录。

## 前置条件

- 用户已提供待整理目录；
- 当前用户对目标目录具有读取权限；
- 需要移动文件时，应获得用户明确确认；
- 如使用脚本，脚本运行环境已准备完成。

```

输入信息

- 待整理目录路径；
- 分类方式：按文件类型、日期或用户自定义规则；
- 操作方式：仅生成整理建议、复制文件或移动文件；
- 输出目录，可选。

执行步骤

1. 检查目录是否存在，并确认当前用户具备访问权限。
2. 读取目录中的文件信息，包括名称、类型、创建时间和大小。
3. 根据用户指定的分类方式生成分类目录。
4. 生成整理计划。
5. 当操作方式为移动文件时，向用户确认后再执行。
6. 输出处理结果和失败文件列表。

输出结果

- 处理状态；
- 已处理文件数量；
- 输出目录路径；
- 未处理文件及失败原因；
- 整理结果清单。

异常处理

- 目录不存在时，返回目录不存在提示；
- 无目录访问权限时，返回权限不足提示；
- 文件被占用或移动失败时，记录具体文件和失败原因；
- 用户未确认移动操作时，仅返回整理建议，不执行文件移动。

调用示例

用户请求：

“将下载目录中的 Word、PDF 和图片分别整理到对应文件夹，并生成一份整理结果清单。”

这个示例中，**description** 用于帮助系统智能体识别任务；“适用场景”和“前置条件”用于限制调用范围；

“执行步骤”定义实际处理顺序；“输出结果”和“异常处理”保证调用结果可验证、可追踪。

如 **Skill** 依赖脚本或业务接口，应在“执行步骤”中明确说明调用对象、所需参数、返回内容和失败处理方式。不要只写“调用脚本完成处理”或“请求接口获取结果”，应写清调用条件和返回结果如何使用。

2.3 Skill 接入、调用与验收

Skill 开发完成后，可通过 openKylin 系统智能体的技能管理功能接入。当前技能管理支持创建、导入、预览、编辑和卸载；导入方式支持 SKILL.md 文件、技能目录或压缩包。导入前会检查 Skill 名称和 SKILL.md 内容。

以“文档整理 Skill”为例，接入过程如下。

2.3.1 准备 Skill 包

将 SKILL.md 和依赖脚本、模板等资源放入同一目录：

```
document-organizer/
├── SKILL.md
├── scripts/
│   └── organize_documents.py
└── templates/
    └── organize-rules.json
```

如不依赖脚本或资源文件，也可以仅导入 SKILL.md。

2.3.2 导入 Skill

在 openKylin 系统智能体中进入“设置 > 技能”，选择“导入”，并选择以下任一内容：

- SKILL.md 文件；
- 完整 Skill 目录；
- 包含 Skill 目录的压缩包。
- 导入完成后，在技能列表中确认 Skill 名称、描述和版本信息是否正确显示。若当前使用远程

智能体运行时，应刷新技能列表，确认运行时已读取最新 Skill。

2.3.3 配置运行条件

Skill 依赖的资源应在导入后确认可用，包括：

项目	检查内容
文件目录	目标目录是否存在，当前用户是否具备读取或写入权限。
脚本环境	Python、Shell 或其他运行依赖是否已安装。
业务接口	服务地址、鉴权信息和网络连通性是否正常。
桌面应用	目标应用是否已安装、可启动且用户已登录。
CUA 环境	CUA 运行时、界面环境和目标窗口是否可用。

2.3.4 通过任务调用 Skill

导入后，可通过自然语言任务验证系统智能体是否调用了目标 Skill。

例如：将“下载”目录中的 Word、PDF 和图片分别整理到对应文件夹，并生成整理结果清单。

预期执行过程如下：

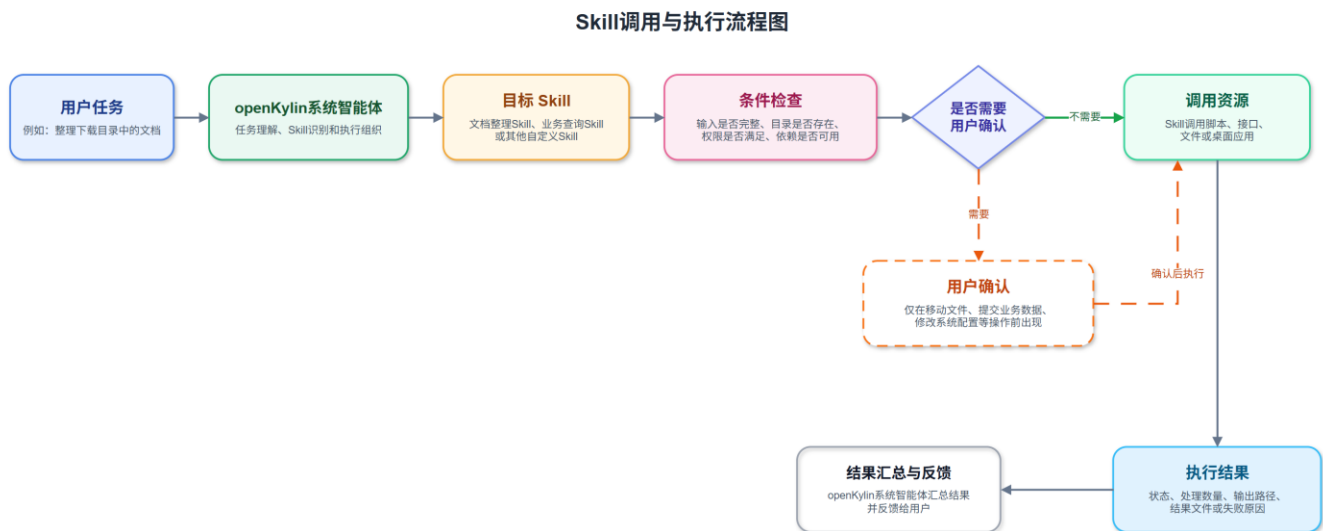


图 4 Skill 调用与执行流程图

对于文档整理 Skill，系统智能体识别任务后，先检查目标目录、分类规则和文件操作权限；涉及移动文件时，先请求用户确认；确认后调整理脚本完成处理，并返回处理数量、输出路径和失败文件清单。

2.3.5 验收和调试

Skill 接入后，至少验证以下场景：

验证项	预期结果
正常任务	系统智能体能够识别并调用正确 Skill。
输入缺失	系统智能体能够提示补充目录、文件、参数或操作规则。
权限不足	Skill 返回明确的权限不足信息，不继续执行。
资源不可用	脚本、接口或应用不可用时，返回具体失败原因。
用户取消	涉及确认操作时，取消后不产生实际变更。
结果返回	返回处理状态、结果文件、输出路径或失败清单。

调试时，可通过 openKylin 系统智能体的过程详情和运行日志查看 Skill 调用情况，包括工具调用记录和返回结果摘要。

Skill 验收通过后，建议保留 Skill 版本、依赖说明、测试任务、权限要求和变更记录，便于后续升级和问题追踪。

3. 直接接入运行时统一模型推理服务

运行时统一模型推理服务面向业务系统、算法服务和第三方智能体提供统一模型调用能力。服务负责模型注册、模型加载、Runtime 调度、请求转发和运行状态观测；业务侧只需要通过模型 ID 调用统一接口，无需直接连接底层 Runtime 或 Worker。

该方式适用于文本生成、知识问答、RAG、Embedding、语音服务、多模态处理及专用算法服务等场景。任务拆解、上下文管理、业务流程和结果使用由接入方自行处理。

3.1 服务准备、模型管理与标准接入流程

直接接入运行时统一模型推理服务时，通常涉及两类角色：

角色	主要工作
模型管理方	配置服务、注册模型或 Worker、设置 Runtime 参数、加载或卸载模型。
业务调用方	查询可用模型，调用推理接口，处理返回结果并记录调用信息。

业务调用方无需直接连接 Runtime 或 Worker，但应了解目标模型或 Worker 的任务类型、输入结构和输出约定。调用时只需要指定模型 ID，运行时统一模型推理服务会根据模型注册信息选择对应 Runtime 或 Worker。



图 5 运行时统一模型推理服务直接接入与调用关系图

运行时统一模型推理服务默认监听地址为：

```
http://127.0.0.1:18080
```

如服务配置了鉴权 Token，所有/v1/*接口均需要携带 Bearer Token。业务调用时建议同时传递调用方名称和请求标识，便于查询调用记录和关联上下游日志。

```
BASE_URL=http://127.0.0.1:18080
TOKEN=<auth_token>

curl -fsS "$BASE_URL/v1/models" \
-H "Authorization: Bearer $TOKEN" \
-H "X-Caller: business-service" \
-H "X-Request-Id: request-001"
```

未启用鉴权时，可不传 `Authorization` 请求头。

3.1.1 服务和模型检查

业务系统首次接入时，建议按以下顺序完成检查：

步骤	接口	检查内容
1	GET /health	确认运行时统一模型推理服务 HTTP 进程可访问。
2	GET /v1/system/config-check	确认配置、目录和 Runtime 可执行文件不存在严重错误。
3	GET /v1/runtimes	确认可用 Runtime 已正确配置。
4	GET /v1/models	查询当前已注册模型。
5	GET /v1/models/{model}	查看目标模型的任务类型、Runtime 和当前状态。
6	POST /v1/models/{model}:load	对高频模型进行预加载，减少首次推理等待时间。

GET /health 只表示网关 HTTP 进程可访问，不表示模型已经注册或加载完成；模型调用前还应检查模型状态和配置检查结果。

运行时统一模型推理服务推荐业务侧在启动时读取模型列表，而不是在业务代码中固定写死模型清单。对于高频调用模型，可通过:load 提前加载；低频模型可由服务按需加载。

3.1.2 模型注册、加载与卸载

模型注册由模型管理方执行。注册时需要指定模型 ID、任务类型、Runtime 和模型资源路径。

以下示例注册一个本地 GGUF 对话模型：

```
curl -fsS -X POST "$BASE_URL/v1/models:install" \
-H "Authorization: Bearer $TOKEN" \
-H "Content-Type: application/json" \
-H "X-Caller: model-admin" \
-H "X-Request-Id: register-qwen3-dev-001" \
-d '{
  "model": "qwen3-dev",
  "version": "1.0.0",
  "task": "chat",
  "runtime": "llama_cpp",
  "source": {
```

```
"type": "local_path",
"uri": "/var/lib/model-gateway/downloads/qwen3/model.gguf"
},
"runtime_options": {
"context_size": 32768,
"gpu_layers": 32
}
}'
```

主要字段说明如下：

字段	说明
model	对外暴露的模型 ID，业务调用时使用该名称。
version	模型版本，未指定时使用默认版本。
task	模型任务类型，可为 chat、embedding 或 asr。
runtime	运行方式，可为 llama_cpp、embedding_cpp、asr_cpp 或 auto。
source.uri	模型文件或资源目录所在路径，服务进程必须具有读取权限。
context_size	模型上下文长度。
gpu_layers	模型加载到 GPU 的层数，可设置为具体数值或 all。

模型注册只写入模型元数据和注册信息，不复制模型权重文件。模型文件应放在服务进程可访问的位置。

模型注册完成后，可按需加载或卸载：

```
# 加载模型
curl -fsS -X POST "$BASE_URL/v1/models/qwen3-dev:load" \
-H "Authorization: Bearer $TOKEN"

# 卸载模型
curl -fsS -X POST "$BASE_URL/v1/models/qwen3-dev:unload" \
-H "Authorization: Bearer $TOKEN"
```

模型状态通常包括：

状态	说明
installed	模型已注册，但尚未加载。
loaded	模型已加载，可接收推理请求。
error	模型加载或推理过程中出现异常。

查询单模型信息时，可查看模型 Runtime、启动参数、Worker 端点、日志路径和最近错误信息。

3.2 统一推理与 OpenAI 兼容调用

运行时统一模型推理服务提供两类主要调用接口：

接口	适用场景
POST /v1/infer	统一推理入口，适合 Chat、Embedding、语音服务和自定义 Worker 等不同任务类型。
POST /v1/chat/completions	OpenAI 兼容对话接口，适合已有 OpenAI SDK 或需要标准 SSE 流式输出的应用。

3.2.1 使用统一推理接口

POST /v1/infer 是通用推理接口。业务系统通过模型 ID 指定目标模型，输入内容由 input 字段传递，推理参数通过 params 字段传递。

```
curl -fsS "$BASE_URL/v1/infer" \
-H "Authorization: Bearer $TOKEN" \
-H "Content-Type: application/json" \
-H "X-Caller: knowledge-service" \
-H "X-Request-Id: infer-001" \
-d '{
  "model_name": "qwen3-dev",
  "input": {
    "messages": [
      {
        "role": "user",
        "content": "请总结以下内容的主要结论。"
      }
    ]
  },
  "params": {
    "temperature": 0.2,
    "max_tokens": 512
  },
  "stream": false,
  "priority": "medium"
}'
```

统一推理请求的主要字段如下：

字段	是否必填	说明
model_name 或 model	是	目标模型 ID。
input	是	模型输入内容，不同任务类型使用不同结构。
params	否	推理参数，透传给对应 Runtime 或 Worker。
stream	否	是否使用流式返回。对 Chat 流式请求，优先使用 OpenAI 兼容接口。

priority	否	调度优先级，可使用 low、medium、high。
----------	---	----------------------------

返回结果重点关注以下字段：

字段	说明
model	实际使用的模型 ID。
runtime	实际使用的 Runtime。
output	模型或 Worker 返回的核心结果。
usage	Token、音频时长等用量信息。
metrics	模型加载、推理和 Worker 内部耗时等指标。

统一推理接口会屏蔽不同 Runtime 之间的调用差异。对于 Chat 模型，请求会被转发至 llama-server；对于 Embedding、语音服务或其他 Worker，会被转发到对应 Worker 的/infer 接口。

3.2.2 使用 OpenAI 兼容接口

已有 OpenAI SDK 或 OpenAI 兼容客户端的业务系统，可直接使用：

POST /v1/chat/completions

示例如下：

```
curl -fsS "$BASE_URL/v1/chat/completions" \
-H "Authorization: Bearer $TOKEN" \
-H "Content-Type: application/json" \
-H "X-Caller: chat-service" \
-H "X-Request-Id: chat-001" \
-d '{
  "model": "qwen3-dev",
  "messages": [
    {
      "role": "user",
      "content": "请用三点说明运行时统一模型推理服务的作用。"
    }
  ],
  "temperature": 0.2,
  "max_tokens": 256,
  "stream": false
}'
```

接口字段与 OpenAI Chat Completions 接口保持一致，常用字段包括：

字段	说明
model	模型 ID。
messages	对话消息列表。

temperature	采样温度。
max_tokens	最大输出 Token 数。
stream	是否使用 SSE 流式返回。

使用流式输出时，将 stream 设置为 true。服务以 SSE 方式返回内容，结束标识为 data: [DONE]。

3.3 Embedding、语音服务与专用 Worker 调用

3.3.1 Embedding 调用

Embedding 模型通过统一推理接口调用。输入可使用单文本、查询文本或文本列表。

```
curl -fsS "$BASE_URL/v1/infer" \
-H "Authorization: Bearer $TOKEN" \
-H "Content-Type: application/json" \
-H "X-Caller: retrieval-service" \
-d '{
  "model_name": "qwen3-embedding-8b-gguf",
  "input": {
    "text": "请帮我打开蓝牙"
  },
  "params": {}
}'
```

常见输入形式如下：

场景	input 示例
单文本向量化	{"text": "待向量化文本"}
查询向量化	{"query": "查询内容"}
批量文本向量化	{"texts": ["文本一", "文本二"]}
专用查询匹配 Worker	{"query": "用户任务"}

对于已注册的专用 Worker，业务调用方式与普通模型保持一致：仍然调用/v1/infer 并指定对应模型 ID，运行时统一模型推理服务负责将请求转发到目标 Worker

3.3.2 文件上传与语音服务调用

音频、图片等文件资源建议先上传至运行时统一模型推理服务，再将返回的资源 URI 传入推理请求。这样业务系统无需与服务端共享本地文件路径。

上传音频文件：

```
curl -fsS -X POST "$BASE_URL/v1/uploads" \
-H "Authorization: Bearer $TOKEN" \
-H "X-Caller: asr-service" \
-F "file=@/path/to/audio.wav"
```

上传成功后，服务返回类似结果：

```
{
  "uri": "uploads://trace-xxx-audio.wav",
  "stored_name": "trace-xxx-audio.wav"
}
```

使用返回的文件 URI 发起 ASR 推理：

```
curl -fsS "$BASE_URL/v1/infer" \
-H "Authorization: Bearer $TOKEN" \
-H "Content-Type: application/json" \
-H "X-Caller: asr-service" \
-d '{
  "model_name": "enterprise-asr",
  "input": {
    "file_uri": "uploads://trace-xxx-audio.wav",
    "payload": {
      "language": "zh",
      "return_timestamps": true
    }
  },
  "params": {}
}'
```

语音服务 ASR Worker 可使用 file_path、file_uri 或 audio_base64 作为输入。业务系统与运行时统一模型推理服务分布在不同主机时，优先使用/v1/uploads 和 uploads://资源引用。

3.3.3 接入验收

直接接入运行时统一模型推理服务后，建议完成以下验证：

验证项	预期结果
服务可达	/health 返回正常。
配置可用	/v1/system/config-check 不存在严重错误。
模型可见	/v1/models 中存在目标模型。
模型可加载	调用:load 后模型状态变为 loaded。
推理可用	/v1/infer 或/v1/chat/completions 能够返回预期结果。
文件可用	上传后的 uploads://资源可被语音服务或多模态 Worker 读取。
调用可追踪	请求日志中可查询到 X-Request-Id、模型 ID、调用方和 trace_id。
异常可定位	模型加载失败、超时或 Worker 错误时，可通过模型日志、过程历史和事件日志定位。

出现问题时，优先检查以下接口：

```
/health
/v1/system/config-check
/v1/models/{model}
/v1/models/{model}/logs
/v1/system/processes
/v1/system/process-history
/v1/system/events
```

其中，`process-history` 用于查看历史请求、调用方、模型、状态和耗时；`events` 用于查询模型加载失败、推理失败和调度变化等结构化事件。

3.4 接口说明

运行时统一模型推理服务默认通过 HTTP 提供接口，基础地址通常为：

```
http://127.0.0.1:18080
```

当服务配置了 `auth_token` 时，所有 `/v1/*` 接口均需携带：

```
Authorization: Bearer <auth_token>
```

业务调用时建议同时传递 `X-Caller` 和 `X-Request-Id`，分别标识调用方和本次请求，便于查询过程历史、事件日志和错误信息。

以下为运行时统一模型推理服务的接口总表。

分类	调用方式	接口	鉴权	作用及调用效果
页面访问	GET	/	否	跳转至 Web 管理界面。
页面访问	GET	/ui	否	跳转至 Web 管理界面。
服务检查	GET	/health	否	检查运行时统一模型推理服务 HTTP 进程是否存活。返回正常仅表示服务可访问，不表示模型已注册或加载。
Runtime 管理	GET	/v1/runtimes	是	查询当前已配置的 Runtime，例如 <code>llama_cpp</code> 、 <code>embedding_cpp</code> 、 <code>asr_cpp</code> ，用于确认底层运行环境是否可用。
模型管理	GET	/v1/models	是	查询已注册模型列表，用于业务启动时发现可用模型。
模型管理	GET	/v1/models/{model}	是	查询指定模型详情，包括模型版本、任务类型、Runtime、加载状态、Worker 地址、启动参数和最近错误信息。
模型管理	POST	/v1/models:install	是	注册模型或模型资源目录。注册后写入模型元数据和注册表，不复制大模型权重文件。
模型管理	DELETE	/v1/models/{model}	是	删除指定模型注册信息。可通过参数控制是否保

				留模型相关文件。
模型管理	POST	/v1/models/{model}:load	是	加载模型，启动对应 Runtime 或 Worker，并执行健康检查。适合对高频模型进行预热。
模型管理	POST	/v1/models/{model}:unload	是	卸载模型，停止对应 Runtime 或 Worker，释放相关资源。
模型管理	GET	/v1/models/{model}/logs	是	查询指定模型的运行日志。可通过 max_bytes 控制返回日志尾部大小。
统一推理	POST	/v1/infer	是	统一推理入口。可调用 Chat、Embedding、ASR 和自定义 Worker，推荐业务系统优先使用该接口。
对话推理	POST	/v1/chat/completions	是	OpenAI Chat Completions 兼容接口，适合已有 OpenAI SDK 的业务系统。支持非流式返回和 SSE 流式返回。
文件资源	POST	/v1/uploads	是	上传音频、图片或其他二进制文件，返回 uploads:// 资源引用，供 ASR 或多模态模型调用。
GPU 观测	GET	/v1/system/gpu	是	查询 GPU 设备、显存和相关进程摘要。
配置检查	GET	/v1/system/config-check	是	检查服务配置、目录权限、Runtime 可执行文件和配置风险。联调前建议优先调用。
调度管理	GET	/v1/system/scheduler	是	查询当前调度策略、队列状态、并发情况、告警阈值和计数信息。
调度管理	POST	/v1/system/scheduler	是	更新调度策略，例如启停调度或切换队列调度模式。
运行过程	GET	/v1/system/processes	是	查询当前正在执行的请求、等待队列、Runtime 进程、调用方和模型摘要。
运行过程	GET	/v1/system/process-history	是	查询历史请求，可按模型、调用方、状态、时间范围和路由过滤，用于统计和问题回溯。
运行过程	DELETE	/v1/system/process-history	是	清空历史请求记录，建议仅在测试或明确运维场景下使用。
事件日志	GET	/v1/system/events	是	查询结构化事件日志，可用于定位模型加载失败、推理失败和调度策略变化等问题。

运行时统一模型推理服务对外主要提供模型管理、统一推理、文件上传及系统观测四类接口。其中，业务系统通常使用 /v1/infer 或 /v1/chat/completions 发起模型调用；模型管理方使用模型注册、加载和卸载接口管理模型；运维或联调人员使用配置检查、GPU、调度、过程历史和事件日志接口进行状态观测与问题定位。

业务系统调用模型时，优先使用以下两类接口：

使用场景	推荐接口	说明
普通模型或 Worker 调用	/v1/infer	统一支持 Chat、Embedding、ASR 和自定义 Worker。
已使用 OpenAI SDK 的对话应用	/v1/chat/completions	使用 OpenAI 兼容请求和响应格式，接入改造量较小。
音频、图片等文件输入	/v1/uploads + /v1/infer	先上传文件，获取 uploads:// 资源引用后再发起推理。
模型预热	/v1/models/{model}:load	高并发或低首包延迟场景下，可提前加载模型。
推理问题排查	/v1/system/process-history、/v1/system/events、 /v1/models/{model}/logs	用于查看请求耗时、执行状态、结构化事件和模型日志。

/v1/infer 支持通过模型 ID 统一调用不同类型的 Runtime 和 Worker；/v1/chat/completions 会在服务内部转换为统一推理请求。

4. 环境与依赖安装

4.1 安装范围

依赖安装根据使用方式区分。

使用场景	需要安装的内容
仅使用 openKylin 系统智能体	安装 kylin-agent 及其运行依赖；需要本地模型能力时，同时安装 uni-model-infer。
开发或修改 openKylin 系统智能体	安装 Qt5 开发组件、SQLite 驱动、构建工具及常用辅助工具。
仅调用远程运行时统一模型推理服务	业务侧只需具备 HTTP 调用能力，不需要本地安装模型、Runtime 或 Worker。
本地部署运行时统一模型推理服务	安装构建工具、SQLite 开发库、Python、Curl，以及实际推理所需的 llama-server 或自定义 Worker。
开发自定义 Skill	根据 Skill 实际功能补充 Python、Shell、业务 SDK、桌面应用或其他依赖。
开发自定义 Worker	在运行时统一模型推理服务基础依赖之外，补充 Worker 所依赖的模型框架、算法库或运行环境。

openKylin 系统智能体运行依赖 Qt5 和 SQLite 等基础组件；启用本地模型下载、注册或本地推理能力时，还需要安装运行时统一模型推理服务。

4.2 openKylin 系统智能体安装

仅使用 openKylin 系统智能体时，可通过软件商店安装，也可使用以下命令：

```
sudo apt update
sudo apt install kylin-agent
```

需要使用本地模型、下载本地模型或调用运行时统一模型推理服务时，安装：

```
sudo apt update
sudo apt install uni-model-infer
```

openKylin 系统智能体首次启动时会检查智能体运行时、CUA 运行时和相关 Skill 状态；发现缺失或版本不一致时，可按界面提示完成安装或更新。

4.3 openKylin 系统智能体源码开发依赖

开发、编译或二次修改 openKylin 系统智能体时，安装以下依赖：

```
sudo apt update
sudo apt install -y build-essential cmake qtbase5-dev libqt5sql5-sqlite libqt5svg5-dev \
zlib1g-dev git curl rsync python3-pip uni-model-infer
```

如果运行时提示 SQLite、SVG 或本地模型推理服务相关组件无法加载，应优先检查 libqt5sql5-sqlite、libqt5svg5-dev、zlib1g-dev 和 uni-model-infer 是否已安装。

4.4 运行时统一模型推理服务部署与开发依赖

本地部署或源码开发运行时统一模型推理服务时，基础环境应满足：

类别	依赖或要求	说明
驱动环境	NVIDIA 显卡驱动，CUDA13 及以上	用于 llama.cpp 采用 GPU 推理
构建工具	CMake 3.20 及以上	用于配置和构建服务。
编译器	支持 C++20 的编译器	用于编译服务源码。
数据库	SQLite3 开发库	用于过程历史、事件和运行数据管理。
基础工具	python3、curl	用于脚本、验证、接口调用和联调。
JSON 与 HTTP 库	nlohmann_json、cpp-httplib	优先使用系统包；系统未提供时可使用项目内置依赖或开启依赖拉取。
实际推理环境	llama-server 或对应 Worker	仅完成网关逻辑验证时可使用 dry_run=true；实际推理时需要可执行 Runtime 或 Worker。

可先安装基础依赖：

```
sudo apt update
sudo apt install -y build-essential cmake libsqlite3-dev python3 curl git
```

系统仓库提供对应软件包时，可额外安装 JSON 和 HTTP 库：

```
sudo apt install -y nlohmann-json3-dev libcpp-httplib-dev
```

如当前系统仓库未提供上述开发包，可使用项目 `third_party/` 目录中的内置依赖，或按项目构建配置启用依赖拉取。实际模型推理还需要部署可执行的 llama-server 或对应 Worker；仅验证网关配置、模型注册和转发逻辑时，可使用 `dry_run=true`。

5. 常见问题

基于 openKylin 系统智能体扩展 Skill，以及直接接入运行时统一模型推理服务时常见的问题和处理方式。Skill 侧主要关注导入、识别、执行和权限；推理服务侧主要关注服务配置、模型注册、Runtime、Worker 和调用链路。现有服务已提供配置检查、模型日志、过程历史和结构化事件等排查入口。

问题	处理方式
openKylin 系统智能体无法启动	确认 openKylin 系统智能体、智能体运行时及相关依赖已安装；如启动时提示运行时或 CUA 组件版本不一致，按提示完成安装或更新后重新启动。
openKylin 系统智能体可以启动，但无法处理任务	检查智能体运行时服务地址、服务密钥和网络连通性；确认至少已配置一个可用模型。
聊天界面没有可选模型	在“设置 > 模型”中添加模型，或确认模型已成功注册至运行时统一模型推理服务，并已写入智能体运行时配置。
本地模型已下载，但聊天界面中不可选择	等待模型注册和智能体运行时配置完成；确认运行时统一模型推理服务可访问；仍未恢复时，可删除模型后重新下载或重新注册。
导入 Skill 失败	检查导入内容是否为有效的 SKILL.md、完整 Skill 目录或压缩包；确认 Skill 名称、目录结构和 SKILL.md 格式正确。
Skill 导入后未显示在技能列表中	在“设置 > 技能”中刷新技能列表；远程智能体运行时场景下，确认 Skill 已同步到运行时可读取的位置。
Skill 可以显示，但无法编辑	仅本地存储模式且 Skill 目录实际存在时支持直接编辑；远程运行时场景下，修改后应重新导入或覆盖对应 Skill 包。
用户任务未触发目标 Skill	检查 Skill 名称和 description 是否准确描述能力范围、适用任务和结果；避免使用“通用工具”“万能助手”等宽泛表述；补充典型调用示例。
系统智能体调用了错误的 Skill	检查多个 Skill 之间是否存在能力重叠；缩小每个 Skill 的职责范围，并在描述中写明适用场景和不适用场景。
Skill 执行前缺少路径、文件或业务参数	在 SKILL.md 中明确输入要求和前置条件；执行前先提示用户补充缺失信息，不直接进入后续处理。
Skill 无法访问本地文件或目录	检查目标路径是否存在，以及当前用户或服务进程是否具有读取、写入或执行权限；涉及用户目录时，确认目录授权范围。
Skill 依赖的 Python、Shell 或其他脚本无法执行	检查解释器、脚本执行权限、依赖包和环境变量；先在终端独立执行脚本，确认脚本本身能够正常返回结果。
Skill 调用业务接口失败	检查接口地址、网络连通性、鉴权信息、请求参数和接口返回

	码；在 Skill 中保留接口错误信息并返回可理解的失败原因。
Skill 调用桌面应用失败	确认目标应用已安装、可启动且用户已登录；检查应用窗口是否处于可操作状态。
CUA Skill 无法完成界面操作	确认 CUA 运行时已安装并与系统智能体版本匹配；检查目标窗口是否可见、未被遮挡，并避免在界面状态不明确时连续执行操作。
涉及移动文件、提交数据或系统修改时未进行确认	在 Skill 执行步骤中明确高风险操作的确认条件；未获得用户确认时，仅生成操作建议或执行计划，不实施实际变更。
Skill 执行完成但结果不完整	在 SKILL.md 中明确输出内容，包括处理状态、输出路径、生成文件、处理数量和失败原因；脚本或接口返回结果应与该约定保持一致。
无法查看 Skill 调用过程	在 openKylin 系统智能体中打开消息过程详情或会话详情；确认智能体运行时已正常产生日志。
运行时统一模型推理服务无法访问	先调用 GET /health 检查服务 HTTP 进程；若服务不可达，检查监听地址、端口、防火墙和服务进程状态。
/health 正常，但模型调用仍然失败	/health 只表示网关进程存活，不表示模型已注册或加载；继续调用 GET /v1/system/config-check、GET /v1/models 和模型详情接口检查。
调用接口返回 401 或鉴权失败	检查服务是否配置了 auth_token；确认请求中携带 Authorization: Bearer <token>，并检查 Token 内容是否正确。
调用接口时返回 INVALID_JSON	检查请求体是否为合法 JSON，特别注意引号、逗号、编码格式和请求头 Content-Type: application/json。
/v1/models 中没有目标模型	由模型管理方调用 POST /v1/models:install 注册模型；确认模型 ID、任务类型、Runtime 和资源路径填写正确。
模型注册失败或提示资源路径不可访问	检查 source.uri 是否存在，模型文件或资源目录是否可被服务进程读取；模型文件不要求固定放置在某一目录，但必须具备访问权限。
提示 INVALID_GGUF_SOURCE	确认使用 llama_cpp Runtime 时，注册的资源为有效 gguf 模型文件；多模态模型还应按需提供 mmproj 文件。
提示 RUNTIME_EXECUTABLE_MISSING	检查 gateway.json 中对应 Runtime 的 executable 路径，确认 llama-server 或自定义 Worker 可执行文件已安装且具备执行权限。
模型已注册但无法加载	调用 GET /v1/system/config-check 检查配置，再通过 GET /v1/models/{model}/logs 查看模型日志；重点检查 Runtime、模型

	路径、GPU 资源和启动参数。
模型首次加载时间较长	对高频模型调用 <code>POST /v1/models/{model}:load</code> 进行预热；同时检查模型大小、存储性能、 <code>health_check_timeout_ms</code> 和 GPU 资源情况。
CUDA 或 GPU 显存不足	降低 <code>gpu_layers</code> 或 <code>context_size</code> 后重新注册或更新模型；调用 <code>GET /v1/system/gpu</code> 查看显存和进程占用。
推理请求超时	适当降低 <code>max_tokens</code> 、缩短输入内容，或调整 <code>infer_timeout_ms</code> ；检查调度队列、模型日志和 <code>Worker</code> 执行状态。
自定义 <code>Worker</code> 无法启动	确认 <code>Worker</code> 可通过 <code>--port</code> 指定端口，并在资源加载完成后使 <code>GET /health</code> 返回成功；检查 <code>Runtime</code> 配置中的启动命令和参数。
自定义 <code>Worker</code> 返回非 JSON	检查 <code>Worker</code> 的 <code>POST /infer</code> 响应格式，确保始终返回 JSON 对象；异常场景也应返回结构化错误信息。
OpenAI SDK 调用失败	确认基础地址指向 <code>/v1</code> ，并使用 <code>POST /v1/chat/completions</code> ；检查模型 ID、 <code>messages</code> 结构和流式输出处理逻辑。
流式对话未正常结束	确认请求中 <code>stream=true</code> ，客户端按照 SSE 读取响应，并以 <code>data: [DONE]</code> 作为结束标识。
语音服务或多模态任务找不到文件	优先调用 <code>POST /v1/uploads</code> 上传文件，并将返回的 <code>uploads://...</code> URI 传入推理请求；避免依赖业务系统与服务端共享本地磁盘路径。
Embedding 或语音服务输入格式错误	Embedding 使用 <code>text</code> 、 <code>query</code> 或 <code>texts</code> ；ASR 使用 <code>file_path</code> 、 <code>file_uri</code> 或 <code>audio_base64</code> 。不同模型或 <code>Worker</code> 的具体输入结构以其任务约定为准。
请求排队或响应变慢	调用 <code>GET /v1/system/scheduler</code> 和 <code>GET /v1/system/processes</code> 查看队列、并发和活动请求；必要时调整调度策略或增加推理资源。
无法定位一次具体调用的异常	请求时统一传递 <code>X-Caller</code> 和 <code>X-Request-Id</code> ；记录响应中的 <code>trace_id</code> ，再通过 <code>process-history</code> 、 <code>events</code> 和模型日志关联排查。
修改模型 <code>Runtime</code> 参数后配置未生效	修改 <code>context_size</code> 、 <code>gpu_layers</code> 等参数后，应重新注册或更新模型，并重新加载对应 <code>Runtime</code> 进程。

模型加载失败、路径权限异常、GPU 显存不足、推理超时和 `Worker` 返回非 JSON 等问题，可优先按照运行时统一模型推理服务的模型日志、过程历史和事件日志进行定位。

6. 开源仓库

6.1 AgentOS SIG 介绍

AgentOS SIG 致力于推动面向操作系统的 Agent 生态建设，围绕 AI PC 与智能操作系统场景，建设从模型推理、Agent 运行时、核心能力组件、评测基准到系统级 Agent 应用的基础设施与协作规范。SIG 维护范围覆盖统一模型推理服务、Agent Runtime、Agent Harness、记忆、规划、技能、工具、调度、优化、评测与基准测试等基础能力，并孵化 system-agent、os-agent、kylin-agent、computer-use-agent 等面向操作系统的人机协同与自动化应用。

AgentOS SIG 希望通过开放的组件体系和工程实践，降低操作系统接入大模型与智能代理能力的门槛，支持系统任务理解、桌面环境交互、统一推理接入、工具调用、多步骤规划、长期记忆、任务调度、执行评测、基准测试、性能优化与多智能体协作等能力在 openKylin 生态中持续演进。

访问链接：<https://gitee.com/openkylin/community/tree/master/sig/AgentOS>

6.2 开源代码地址

openKylin 系统智能体及运行时统一模型推理服务的源代码已托管于 Gitee。开发者可通过以下仓库获取源码、查看项目说明，并结合实际需求开展部署、二次开发和能力扩展。

- openKylin 系统智能体：<https://gitee.com/openkylin/kylin-agent>
- 运行时统一模型推理服务：<https://gitee.com/openkylin/uni-model-infer>
- CUA 智能体：<https://gitee.com/openkylin/computer-use-agent>
- Agent 运行时：<https://gitee.com/openkylin/agent-runtime>