AI 智能体接入指南

# OpenSAI 智能体接入指南

通过 MCP 工具服务与使用 Skill，让 AI 智能体把自然语言计算需求转化为可控的分布式执行。

产品概览 | 快速上手 | 使用手册 | FAQ 与维护

---

接入 OpenSAI 后，支持 Model Context Protocol（MCP）的智能体（如 Claude Desktop、Claude Code、Cursor、OpenClaw 等）可以自主查询在线设备、发现已注册应用、按截止时间或质量目标提交作业、持续跟踪进度，并用通俗语言向你解释平台的设备与参数决策。例如，你可以直接对智能体说：*“用 `cluster_analytics` 处理每分片 30000 个点，目标 3 秒内完成，跑完告诉我耗时、结果质量以及平台选了哪档参数。”*

OpenSAI 的智能体能力由两部分协同构成：

- **MCP 服务（`opensai_mcp`）**：向客户端暴露 17 个结构化工具（`opensai_*`），负责真实的集群查询、任务提交与权限限流；
- **使用 Skill（SKILL.md）**：向智能体提供操作规范，指导它如何优先选用应用、如何将用户意图映射为 QoS 与旋钮参数、如何避免超时重复提交以及如何准确解读指标。




OpenSAI 智能体 MCP 与 Skill 协同架构

> ✦ **实用提示**
>
> **推荐接入路径**：首次体验建议在同一台电脑上采用 **本机 stdio 模式**（第 1～4 节），无需额外配置令牌即可快速跑通闭环；跨机器共享集群或仅在笔记本安装轻量客户端时，再参考第 5～6 节。

## 1. 准备本机 MCP 运行环境

请先按 快速上手指南 启动本地集群，并确认 `opensai devices list` 能够列出在线设备。随后在项目根目录安装智能体扩展依赖并执行连通性自检：

**Linux / macOS**：

```bash
bash scripts/setup.sh --agent
source .venv/bin/activate
python -m opensai_mcp --check
```

**Windows PowerShell**：

```powershell
powershell -ExecutionPolicy Bypass -File scripts\setup.ps1 -Agent
.venv\Scripts\Activate.ps1
python -m opensai_mcp --check
```

（若虚拟环境已创建，也可直接执行 `python -m pip install -r Offloading-System/requirements-agent.txt`。）`--check` 通过即表明 MCP 服务已能正常访问当前中枢与应用目录。

## 2. 配置本机 stdio 客户端连接

在支持 MCP 的客户端中新增一个名为 `opensai` 的本地 stdio 服务器，核心字段如下：

| 配置项 | 填写内容 |
| --- | --- |
| `command` | 项目 `.venv` 内 Python 解释器的**绝对路径** |
| `args` | `["-m", "opensai_mcp"]` |
| `cwd` | 解压后的 OpenSAI 项目根目录绝对路径 |
| `env` | `OPENSAI_HOST`（默认 `127.0.0.1`）与 `OPENSAI_PORT`（默认 `8888`） |

采用标准 `mcpServers` JSON 配置文件的客户端可直接参考以下模板（请将 `/path/to/OpenSAI_Release` 替换为你本机的实际解压路径）：

```json
{
  "mcpServers": {
    "opensai": {
      "command": "/path/to/OpenSAI_Release/.venv/bin/python",
      "args": ["-m", "opensai_mcp"],
      "cwd": "/path/to/OpenSAI_Release",
      "env": {
        "OPENSAI_HOST": "127.0.0.1",
        "OPENSAI_PORT": "8888"
      }
    }
  }
}
```

Windows 下 `command` 指向 `.venv\Scripts\python.exe`，在 JSON 字符串中请注意将反斜杠转义为双反斜杠（例如 `"C:\\OpenSAI_Release\\.venv\\Scripts\\python.exe"`）。保存后刷新客户端 MCP 连接或新开对话，工具列表中应出现 `opensai_list_devices`、`opensai_list_apps` 等工具。

## 3. 加载使用 Skill

在解压后的代码包目录中，已随附中英文两套等价的技能定义文件：

- 中文版：`skills/zh-CN/opensai/SKILL.md`
- 英文版：`skills/en/opensai/SKILL.md`

将所选语言下的 `opensai/` 文件夹放入客户端支持的技能目录，或使用客户端的“导入技能（Import Skill）”功能载入（两份文件同名 `opensai`，按团队日常对话语言任选一份即可）。若客户端暂不支持独立技能目录，也可将 `SKILL.md` 正文粘贴到工作区级自定义指令（Workspace Rules / Instructions）中。

◆ **重要约定**

**注意先后顺序**：加载 `SKILL.md` 仅提供操作说明，不会自动建立网络连接；必须先完成第 2 节的 MCP 服务器配置，智能体才能真正看到并调用 `opensai_*` 工具。

## 4. 完成第一次引导式对话验证

建议按以下三个循序渐进的自然语言请求验证完整链路：

1. **只读发现**：*“查看当前在线的 OpenSAI 设备和已注册应用，告诉我现在能跑什么。”*
2. **基础计算**：*“提交一次采样数为 200000 的 Pi 计算任务，等待完成后告诉我结果和总耗时。”*
3. **目标驱动适配**（需先按快速上手注册 `cluster_analytics` 并在执行设备安装 `numpy`）：*“运行 `cluster_analytics`，每分片 30000 个点，目标 3 秒内完成；返回聚类结果、实际耗时、质量得分以及平台选中的旋钮配置。”*

若长任务单次等待返回未完成，可继续说：*“用刚才拿到的作业号继续查询结果，不要重新提交。”*

✓ **接入验证标志**

智能体不仅能看到工具名称，还能准确列出真实在线设备、返回作业执行结果，且你能在 Web 控制台（`http://127.0.0.1:5000`）的 **“任务”（Runs）** 页面查到同一个 `job_id`。

## 5. 远程 Streamable HTTP 接入

当智能体客户端运行在另一台电脑、需要远程连接团队共享的 MCP 服务时，请在运行 `opensai_mcp` 的机器上生成高强度随机令牌并启动 HTTP 监听：

```bash
export OPENSAI_HOST=127.0.0.1
export OPENSAI_MCP_TOKEN="$(python -c 'import secrets; print(secrets.token_urlsafe(32))')"
echo "Bearer token: $OPENSAI_MCP_TOKEN"
python -m opensai_mcp --http --listen 0.0.0.0 --listen-port 8765
```

在远端 AI 客户端中按如下参数配置：

| 客户端配置项 | 示例值 |
| --- | --- |
| Transport | `Streamable HTTP` |
| Server URL | `http://192.168.1.10:8765/mcp`（替换为 MCP 主机可达地址） |
| Authorization Header | `Bearer <OPENSAI_MCP_TOKEN>` |

▲ **安全与边界提醒**

**安全与路径须知**：监听非回环地址时，服务端默认强制校验 Bearer 令牌（缺失或错误将返回 `HTTP 401`）；跨公网访问请务必配合 VPN / Tailscale 或反向代理 HTTPS 使用。此外，远程模式下工具参数中的本地路径（`path`）均解析在 **MCP 服务所在机器** 上；若应用文件仅在你的个人电脑上，可通过注册工具直接传入 `manifest` 与 `files` 源码文本。

## 6. 轻量客户端独立部署（仅连接已有集群）

如果团队已有运行中的 OpenSAI 中枢与工作设备集群，而你只想在个人笔记本上安装轻量客户端与 MCP 服务（不在本机部署 Redis、中枢或设备端重依赖），可直接克隆仓库（`git clone https://github.com/INSAI-institute/OpenSAI.git`）或解压 `OpenSAI_Release.zip` 并仅安装客户端组件：

**Linux / macOS**：

```bash
python -m venv .venv
source .venv/bin/activate
python -m pip install ".[mcp]"
python -m opensai_mcp --hub-host 192.168.1.10 --check
```

**Windows PowerShell**：

```powershell
python -m venv .venv
.venv\Scripts\Activate.ps1
python -m pip install ".[mcp]"
python -m opensai_mcp --hub-host 192.168.1.10 --check
```

将 `192.168.1.10` 替换为实际中枢 IP，检查通过后即可按第 2～3 节配置本地 stdio 连接并载入同目录下的 `skills/zh-CN/opensai/SKILL.md`。

## 7. MCP 工具速查与参数约定

| 工具名称 | 功能说明 |
| --- | --- |
| `opensai_list_devices`、`opensai_cluster_status` | 查看在线设备、算力槽位与集群整体健康状态 |
| `opensai_list_algorithms`、`opensai_catalog` | 查询可用底层算法、输入参数与可调旋钮定义 |
| `opensai_list_apps` | 列出已注册应用包及其输入、旋钮与默认 QoS 目标 |
| `opensai_run_app`、`opensai_submit_task` | 按目标参数运行已注册应用，或提交单算法任务 |
| `opensai_validate_dag`、`opensai_submit_dag` | 校验并提交自定义多步骤 DAG |
| `opensai_get_result`、`opensai_wait_result` | 获取作业当前快照，或阻塞等待至终态 |
| `opensai_explain_job` | 查询作业的设备放置、核数分配与旋钮选择解释 |
| `opensai_validate_app`、`opensai_register_app`、`opensai_unregister_app` | 校验、注册或注销授权范围内的应用包 |
| `opensai_list_data`、`opensai_upload_data` | 查询各设备本地数据集，或向指定设备上传小文件 |

注意：MCP 工具中 `deadline_s`、`mode`、`min_quality` 等 QoS 参数直接位于顶层字段（而在 Python SDK 中则置于 `qos={...}` 字典内）。`opensai_run_app` 调用参数示例：

```json
{
  "app": "cluster_analytics",
  "inputs": {"points": 30000, "seed": 1},
  "deadline_s": 3,
  "wait": true,
  "wait_timeout": 60
}
```

## 8. 团队权限、限流与审计配置

多成员共用一套 MCP 服务时，管理员可通过环境变量收紧工具暴露范围、提交速率与写操作权限：

| 环境变量 | 默认值 | 控制作用 |
| --- | --- | --- |
| `OPENSAI_MCP_TOOLS` | 全部启用 | 逗号分隔的允许开放工具白名单 |
| `OPENSAI_MCP_ALLOW_REGISTER` | `1` | 设为 `0` 可禁止智能体注册或注销应用包 |
| `OPENSAI_MCP_ALLOW_UPLOAD` | `1` | 设为 `0` 可禁止通过 MCP 上传数据文件 |
| `OPENSAI_MCP_MAX_NODES` | `64` | 单次提交 DAG 允许的最大节点数 |
| `OPENSAI_MCP_MAX_WAIT_S` | `120` | 单次 `opensai_wait_result` 允许的最大等待秒数 |
| `OPENSAI_MCP_SUBMIT_RATE` | `2` | 按提交工具与调用方标识限制的每秒提交上限 |
| `OPENSAI_MCP_ACTOR` | `openclaw` | 调用方身份标识（写入审计日志并参与限流统计） |
| `OPENSAI_MCP_AUDIT_LOG` | 未设置 | 审计日志 JSONL 输出文件路径 |

由于应用注册会将步骤 Python 代码分发至工作设备运行，请仅注册用户本人提供或明确审阅认可的代码包，并始终先执行 `opensai_validate_app`。

## 9. 常见连接问题速查

| 遇到的现象 | 排查与解决方法 |
| --- | --- |
| 客户端看不到任何 `opensai_*` 工具 | 检查 JSON 配置中的 `.venv` Python 绝对路径、`cwd` 目录及 `--agent` 依赖；终端手动运行 `python -m opensai_mcp --check` |
| 工具能看到，但调用报连接错误 | 确认 OpenSAI 中枢已启动，且 `OPENSAI_HOST` / `OPENSAI_PORT` 指向可达中枢地址（默认 `8888`） |
| 远程连接返回 `HTTP 401` | 核对客户端 `Authorization: Bearer <TOKEN>` 与服务端环境变量 `OPENSAI_MCP_TOKEN` 是否完全一致 |
| 注册或上传时报“找不到路径” | 远程 HTTP 模式下文件路径基于 MCP 服务端机器；请将文件放到该机器或改传 `manifest` + `files` 文本 |
| 等待返回 `done=false` | 表示作业仍在后台运行；保留原 `job_id` 继续调用 `opensai_wait_result` 或 `opensai_get_result` |

| 遇到的现象 | 排查与解决方法 |
| --- | --- |
| 返回 `RATE_LIMITED` | 触发提交频率保护；按返回字段中的 `retry_after_s` 秒数等待后重试 |

更多平台配置与 SDK 接口说明请查阅 软件使用手册（英文版见 Agent Integration Guide）。