云点播

媒体 AI Agent

腾讯云

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

媒体 AI Agent
智能成片 Agent 接入

最近更新时间：2026-08-18 17:33:32

产品能力介绍

智能成片 Agent，只需输入一批视频素材，用自然语言输入成片要求，Agent 即可自动完成素材解析、文案创作、AI 配音、智能配乐、精细化剪辑、字幕生成全流程，直接输出可对外发布的完整成片。它并非传统剪辑工具，而是具备剪辑决策能力的智能 Agent。您仅需提供素材与创作意图，即可交由 Agent 闭环完成「解析素材内容 → 搭建视频叙事结构 → 筛选适配镜头 → 对齐节奏卡点 → 渲染输出 → 成片自检」的完整生产链路，一站式交付成品视频。

详细功能介绍

智能成片 Agent 支持**原声成片和旁白成片**两种成片模式，通过自然语言输入可覆盖从原始素材到可直接发布成片的全流程。所有成片均支持字幕、背景音乐、画幅等通用配置。

成片模式配置

成片模式	能力简介	可支持的配置
原声成片	保留素材中的原始声音，由 AI 完成片段的筛选、排序、裁剪与拼接，生成剪辑方案并成片。适用于口播测评、访谈实录、直播回放切片、赛事/活动高光等需要保留真实原声的场景。	• 内容组织策略：视频目标（带货、种草、品牌宣传、活动回顾等）、受众；可补充核心卖点、参考结构（开场钩子—主体—结尾 CTA）、必须出现的片段。 • 成片时长：AI 自动决定 / 15s / 20s / 25s / 30s / 60s / 自定义。 • 素材声音处理：保留完整原声 / 去除 BGM 仅保留人声。 • 音频：使用视频原声。

旁白成片	移除素材原声，由 AI 生成或校验旁白脚本，合成 TTS 配音，并根据旁白逐句匹配画面。适用于产品讲解、解说二创、品牌宣传片等需要统一解说口径的场景。	• 旁白脚本来源：用户提供脚本 / AI 生成脚本（直接生成或分析素材后结合要求生成）。 • 脚本内容要求：仅 AI 生成脚本时可配，视频目标、受众，可补充核心卖点、参考结构、必现片段。 • 成片时长：AI 自动决定 / 15s / 20s / 25s / 30s / 60s / 自定义。 • 素材声音处理：移除全部原声。 • 音频：使用 TTS。

通用视频配置

配置项	支持范围
字幕	不压制字幕 / 压制字幕（支持配置字幕大小、描边等）。
背景音乐	无需 BGM / 上传 BGM / AI 生成 BGM。
画幅	横屏 / 竖屏。

适用场景

场景	为什么合适
电商带货短视频（抖音/千川/小红书/B 站）。	五段式营销结构 + 产品识别 + 黄金3秒 + 质检 SOP，直接对齐投放要求。
一堆实拍素材没文案，要一条成片。	自动看片写脚本，最省事。
已有爆款文案，要快速配画面成片。	脚本质检 + 裂变多版钩子 + 精准配画。
口播/测评/知识分享（保留本人声音）。	按转录文本剪，句子完整不切字。
访谈/直播长视频切精华。	按语音事件点和静音间隙找剪切点。
带货直播回放切片。	从口播里识别产品型号，只留同款片段。
高光剪辑（直播/赛事/游戏/活动切集锦）。	长视频转录/概览后挑高能片段，保留原声拼成短片。
解说二创（给现成视频重配解说）。	拿已有成片当素材，重写解说词 + AI 配音 + 配画面。

纪录片/Vlog/活动混剪。	通用化，不预设类型。

API 接入

此能力使用 AG-UI 协议接入。

1. 协议概述

AG-UI（Agent-User Interaction Protocol）是基于 HTTPS + SSE 的流式 Agent 交互协议，提供丰富的能力：

- 完整的事件体系（文本消息、推理过程、工具调用生命周期、步骤追踪）。
- 原生 HITL 中断/恢复机制（通过 RunFinished outcome 实现）。
- 每个工具独立中断与恢复（每个待确认工具生成独立 Interrupt，可分别批准/拒绝）。
- 消息快照查询（/history 端点）。

2. 请求

创建 API Token

在进行调用前，需要通过 创建 AIGC API Token 接口 创建 API Token。

> ⓘ **注意：**
> - Token 没有过期时间，只要不主动删除，Token 长期有效，无需每次调用都创建一个 Token。
> - 创建的 Token 会与选择的子应用关联，后续调用 Agent 时，Agent 只能访问该子应用下的文件。
> - 请妥善保管您创建的 Token，避免泄露。

启动 Agent 对话

VOD Agent 采用 AG-UI 协议，您也可以参考 AG-UI 官方文档 了解各个字段的含义。
请求地址 (URL)：https://smartmedia.vod-qcloud.com/agui/chat

```
POST /agui/chat
Host: smartmedia.vod-qcloud.com
Content-Type: application/json
Authorization: Bearer <TOKEN>


{
  "threadId": "my-thread-001",
  "runId": "run-001",
  "messages": [
```

```
    {"role": "user", "content": "帮我根据提供的素材剪一条抖音竖屏带货短视频，时长30 秒左右，主推产品是XXX，突出 XXX 卖点。你来写口播文案、配 AI 配音-1.2倍速、加背景音乐、去掉素材里的原声、加字幕-字号50px，描边3px，节奏明快一点，结尾引导下单。素材URL：XXXX "}
  ],
  "forwardedProps": {
    "approval_mode": "level:high",
    "scenario_name": "video-mixcut"
  }
}
```

字段	类型	必填	说明
threadId	string	是	会话线程 ID，同一线程共享上下文。
runId	string	是	本次运行的唯一 ID。
messages	array of Message	是	消息列表。
tool	array	否	外部工具列表，详见 外部工具说明 。
forwardedProps	object	否	扩展配置，详见 下表 。
resume	array	否	HITL 恢复条目，详见 下文说明 。

Message 字段

字段	类型	必填	说明
role	string	是	消息角色，可选值： • user • tool
content	string or array of ContentPart	是	消息内容。 • 对于纯文本输入，可以直接填写字符串。 • 对于包含图片等附件的输入，请使用 ContentPart 数组。

ContentPart 字段

字段	类型	必填	说明
type	string	是	内容类型，可选值： • text

			• mixcut_assets
text	string	否	当 type 为 text 时填写文本内容。
metadata	object	否	内容片段的扩展元数据。

forwardedProps 字段

字段	类型	必填	默认值	说明
model	string	否	wand-1.0-lite	agent 所使用的模型，可选值： • wand-1.0-lite 轻量版，快速且高性价比。 • wand-1.0-standard（即将支持） 速度、成本与质量的平衡。 • wand-1.0-pro 更强的推理能力和更长的上下文，适合复杂问题与生产级任务。 • wand-1.0-max 面向最具挑战性任务的顶级专家档，出色的推理与指令跟随能力。
approval_mode	string	否	never	审批模式： • level:high 在高风险工具调用时触发审批。（适用场景：审核剪辑方案后成片） • never 从不触发审批。（适用场景：直接成片）
scenario_name	string	是	-	场景名称，当前可选值为： • video-qa 视频问答场景。 • video-mixcut 视频混剪场景。
database	string	否	default	智能媒资库名，目前仅在 video-qa 场景下有效。

查询消息历史

请求地址 (URL)：https://smartmedia.vod-qcloud.com/agui/history

```
POST /agui/history
Host: smartmedia.vod-qcloud.com
Content-Type: application/json
```

```
Authorization: Bearer <TOKEN>

{
    "threadId": "my-thread-001"
}
```

返回该线程的完整消息快照。

3. 响应（SSE 事件流）

```
id: <ID>
data: {"type":"<EventType>","timestamp":<timestamp>,...}
```

事件类型

生命周期事件

事件类型	说明
RUN_STARTED	运行开始。
RUN_FINISHED	运行结束（正常完成或中断）。
RUN_ERROR	运行出错。

消息事件

事件类型	说明
TEXT_MESSAGE_START	文本消息开始。
TEXT_MESSAGE_CONTENT	文本内容增量。
TEXT_MESSAGE_END	文本消息结束。

推理事件

事件类型	说明
REASONING_START	推理阶段开始。
REASONING_MESSAGE_START	推理消息开始。
REASONING_MESSAGE_CONTENT	推理内容增量。
REASONING_MESSAGE_END	推理消息结束。

REASONING_END	推理阶段结束。

工具调用事件

事件类型	说明
TOOL_CALL_START	工具调用开始（携带 toolCallId、toolCallName）。
TOOL_CALL_ARGS	工具参数增量。
TOOL_CALL_END	工具调用结束。
TOOL_CALL_RESULT	工具调用结果。

正常对话事件流

以下是一次完整对话的实际 SSE 输出（简化）：

```
id: RUN_STARTED_1784102011155
data: {"type":"RUN_STARTED","timestamp":1784102011155,"threadId":"thread-
001","runId":"run-001"}

id: REASONING_START_1784102012264
data:
{"type":"REASONING_START","timestamp":1784102012264,"messageId":"msg-001"}

id: REASONING_MESSAGE_START_1784102012264
data:
{"type":"REASONING_MESSAGE_START","timestamp":1784102012264,"messageId":"m
sg-001","role":"reasoning"}

id: REASONING_MESSAGE_CONTENT_1784102012264
data:
{"type":"REASONING_MESSAGE_CONTENT","timestamp":1784102012264,"messageId":
"msg-001","delta":"用户想要搜索海浪视频..."}

id: REASONING_MESSAGE_END_1784102013400
data:
{"type":"REASONING_MESSAGE_END","timestamp":1784102013400,"messageId":"msg
-001"}

id: REASONING_END_1784102013401
```

```
data: {"type":"REASONING_END","timestamp":1784102013401,"messageId":"msg-
001"}

id: TEXT_MESSAGE_START_1784102019000
data:
{"type":"TEXT_MESSAGE_START","timestamp":1784102019000,"messageId":"msg-
002"}

id: TEXT_MESSAGE_CONTENT_1784102019050
data:
{"type":"TEXT_MESSAGE_CONTENT","timestamp":1784102019050,"messageId":"msg-
002","delta":"我来帮您搜索。"}

id: TEXT_MESSAGE_END_1784102024858
data:
{"type":"TEXT_MESSAGE_END","timestamp":1784102024858,"messageId":"msg-
002"}

id: TOOL_CALL_START_1784102126631
data:
{"type":"TOOL_CALL_START","timestamp":1784102126631,"toolCallId":"call-
001","toolCallName":"search_media_by_semantics","parentMessageId":"msg-
002"}

id: TOOL_CALL_ARGS_1784102126631
data:
{"type":"TOOL_CALL_ARGS","timestamp":1784102126631,"toolCallId":"call-
001","delta":"{\"query\":\"海浪\"}"}

id: TOOL_CALL_END_1784102126631
data:
{"type":"TOOL_CALL_END","timestamp":1784102126631,"toolCallId":"call-001"}

id: TOOL_CALL_RESULT_1784102126886
data:
{"type":"TOOL_CALL_RESULT","timestamp":1784102126886,"messageId":"msg-
003","toolCallId":"call-001","content":"{\"Recall\":[]}","role":"tool"}

id: RUN_FINISHED_1784102025004
```

```
data: {"type":"RUN_FINISHED","timestamp":1784102025004,"threadId":"thread-
001","runId":"run-001"}
```

HITL 中断事件流

当工具需要确认时，RUN_FINISHED 的 data 中携带 outcome.type = "interrupt"。每个待确认工具生成一个独立的 Interrupt：

```
id: TOOL_CALL_START_1784102316642
data:
{"type":"TOOL_CALL_START","timestamp":1784102316642,"toolCallId":"call-
001","toolCallName":"search_media_by_semantics","parentMessageId":"msg-
001"}

id: TOOL_CALL_END_1784102316642
data:
{"type":"TOOL_CALL_END","timestamp":1784102316642,"toolCallId":"call-001"}

id: RUN_FINISHED_1784102316731
data: {"type":"RUN_FINISHED","timestamp":1784102316731,"threadId":"thread-
001","runId":"run-001","outcome":{"type":"interrupt","interrupts":
[{"id":"interruptId-001","reason":"tool_call","message":"在视频库中语义检索与
海浪相关的视频片段","toolCallId":"call-001","responseSchema":{"properties":
{"feedback":{"description":"Optional user feedback for the approval
decision.","type":"string"}},"type":"object"}}]}}
```

其中，outcome.interrupts[N].message 为使用自然语义描述的工具调用说明，客户端可以把它展示给用户，并请求用户审批。

Interrupt 字段

字段	说明
id	中断 ID，恢复时需作为 resume[].interruptId 传入。
reason	中断原因，固定 tool_call。
message	工具操作描述（LLM 生成）。
toolCallId	关联的工具调用 ID。
responseSchema	恢复响应 schema，包含可选的 feedback 字段。

每个待确认工具都有独立的 Interrupt ID，客户端可据此对每个工具分别恢复。

4. HITL 中断与恢复

审批模式

模式	说明
never（默认）	所有工具直接执行。
level:high	仅高风险工具中断。

取值不为 never 时，可能在 tool 调用前触发 HITL 中断。

恢复请求

用户做出决策后，发送新请求，在 resume 数组中为每个 Interrupt 传入一个恢复条目。

全部批准

```
{
  "threadId": "my-thread-001",
  "runId": "run-002",
  "messages": [{"role": "user", "content": ""}],
  "forwardedProps": {"approval_mode": "level:low", "scenario_name":
"video-mixcut"},
  "resume": [
    {"interruptId": "interruptId-001", "status": "resolved"},
    {"interruptId": "interruptId-002", "status": "resolved"}
  ]
}
```

全部拒绝（附带反馈）

```
{
  "threadId": "my-thread-001",
  "runId": "run-002",
  "messages": [{"role": "user", "content": ""}],
  "forwardedProps": {"approval_mode": "level:low", "scenario_name":
"video-mixcut"},
  "resume": [
    {"interruptId": "interruptId-001", "status": "cancelled", "payload":
{"feedback": "不需要裁剪"}},
```

```
    {"interruptId": "interruptId-002", "status": "cancelled", "payload":
{"feedback": "不需要水印"}}
  ]
}
```

每工具独立决策（部分批准、部分拒绝）

```
{
  "threadId": "my-thread-001",
  "runId": "run-002",
  "messages": [{"role": "user", "content": ""}],
  "forwardedProps": {"approval_mode": "level:low", "scenario_name":
"video-mixcut"},
  "resume": [
    {"interruptId": "interruptId-001", "status": "resolved"},
    {"interruptId": "interruptId-002", "status": "cancelled", "payload":
{"feedback": "不要添加水印"}}
  ]
}
```

Resume 条目字段

字段	类型	必填	说明
interruptId	string	是	中断响应中返回的 id，不能为空。
status	string	是	resolved（批准）或 cancelled（拒绝）。
payload	object	否	可包含 feedback 字段，传递用户反馈。

说明：

- 每个条目的 interruptId 不能为空。
- 同一 interruptId 不能在 resume 数组中出现多次。
- resume 条目应与 interrupts 一一对应。
- 恢复请求仍需携带相同的 forwardedProps.approval_mode、forwardedProps.scenario_name。
- 恢复请求的 messages 可传空字符串内容（{"role": "user", "content": ""}），无需重复用户指令。

5. Scenario 场景配置

通过 forwardedProps.scenario_name 引用服务端预设场景：

```
{
  "threadId": "thread-001",
  "runId": "run-001",
  "messages": [{"role": "user", "content": "搜索海浪视频"}],
  "forwardedProps": {"scenario_name": "video-mixcut"}
}
```

当前支持的场景为：

- video-qa
- video-mixcut

video-qa（视频问答）

视频问答场景支持在 forwardedProps.database 中指定 agent 可见的智能媒资知识库，您可以在 腾讯云控制台 中查看和管理已有知识库。

video-mixcut（智能成片）

- 视频混剪场景支持在用户消息中带上混剪素材，素材使用自定义的 ContentPart，类型为 mixcut_assets 。
- 附件列表放在 metadata.attachments 中，每个附件只填写一个来源字段：url 或 fileId。

字段	类型	必填	说明
type	string	是	固定填写 mixcut_assets。
metadata	object	是	混剪素材的扩展信息。
metadata.attachments	array of object	是	当前 user message 的附件列表。
metadata.attachments[].url	string	二选一	外部素材 URL。
metadata.attachments[].fileId	string		VOD 文件 ID。

完整请求示例：

```
{
  "threadId": "thread-001",
  "runId": "run-001",
  "messages": [
    {
      "role": "user",
```

```
      "content": [
        {
          "type": "text",
          "text": "帮我混剪这些素材"
        },
        {
          "type": "mixcut_assets",
          "metadata": {
            "attachments": [
              {"url": " http://example.com/video-a.mp4"},
              {"fileId": "vod-file-001"}
            ]
          }
        }
      ]
    }
  ]
}
```

6. 外部工具

外部工具是 Agent 调用，但不由服务端直接执行的工具。LLM 决策调用后，服务端挂起执行将工具调用信息返回给客户端，由用户在客户端执行工具并回填执行结果，然后继续 Agent 流程。这适用于需要人工介入或客户端侧执行的场景（如内部知识库查询、审批操作等）。

工具 Schema 定义

每个外部工具通过标准 JSON Schema 定义，由客户端传入到请求的 tools 字段进行注册：

```
{
  "name": "<工具名>",
  "description": "<工具用途描述，LLM 据此决策何时调用>",
  "parameters": <工具参数的JSON schema定义>
}
```

示例一：clarify 工具 Schema：

```
{
  "name": "clarify",
  "description": "Ask the user a question when you need clarification...",
  "parameters": {
```

```
    "type": "object",
    "properties": {
      "question": {
        "type": "string",
        "description": "The question itself. Do NOT embed options here."
      },
      "choices": {
        "type": "array",
        "items": { "type": "string" },
        "maxItems": 4,
        "description": "Selectable options (up to 4). Omit for free-text."
      }
    },
    "required": ["question"]
  }
}
```

交互流程

外部工具的完整交互模型：

```
请求 → LLM 决策调用外部工具
  ↓  SSE 事件流
  TOOL_CALL_START    { toolCallId, toolCallName }
  TOOL_CALL_ARGS     { delta (JSON fragment) }
  TOOL_CALL_END      { toolCallId }
  ↓  工具执行被挂起，Agent结束当前轮次
  RUN_FINISHED

[客户端执行外部工具，填入结果]
  ↓
  发送新请求（/agui/chat）
  {
    "threadId": "...",
    "runId": "...",
    "messages": [ { "role": "tool", "content": "<工具执行结果>",
"toolCallId": "<id>" } ],
    "tools": [ ... ]
  }
  → Agent 继续执行
```

SSE 事件示例

```
id: TOOL_CALL_START_xxx
data: {"type":"TOOL_CALL_START","timestamp":...,"toolCallId":"call-ext-
001","toolCallName":"external_search","parentMessageId":"msg-001"}

id: TOOL_CALL_ARGS_xxx
data: {"type":"TOOL_CALL_ARGS","timestamp":...,"toolCallId":"call-ext-
001","delta":"{\"query\":\"VOD 架构文档\"}"}

id: TOOL_CALL_END_xxx
data: {"type":"TOOL_CALL_END","timestamp":...,"toolCallId":"call-ext-001"}

id: RUN_FINISHED_xxx
data: {"type":"RUN_FINISHED","timestamp":...,"threadId":"thread-
001","runId":"run-001"}
```

RUN_FINISHED 为普通结束事件，不携带 outcome 或 interrupts。

恢复请求

外部工具执行完毕后，客户端发起新请求，在 messages 中以 role: "tool" 回传执行结果：

```
{
  "threadId": "my-thread-001",
  "runId": "run-002",
  "messages": [
    {
      "role": "tool",
      "content": "<工具执行结果（字符串，LLM 会解读）>",
      "toolCallId": "call-ext-001"
    }
  ],
  "tools": [
    {
      "name": "clarify",
      "description": "...",
      "parameters": {"type":"object","properties":{}}
    }
  ]
```

```
}
```

字段	类型	必填	说明
messages[].role	string	是	固定为 "tool"。
messages[].content	string	是	工具执行结果，任意字符串。
messages[].toolCallId	string	是	对应中断事件中的 toolCallId。
tools	array	是	工具 Schema 列表，与首次请求保持一致。
forwardedProps	object	否	需携带首次请求中的 approval_mode 等扩展配置。

tools、forwardedProps 参数请与前一次请求保持一致。

7. SDK 调用示例

Python（AG-UI Python SDK）

安装：pip install ag-ui-protocol

```
from ag_ui.client import HttpAgent
from ag_ui.core import RunAgentInput, Message, ResumeEntry

BASE_URL = "https://smartmedia.vod-qcloud.com/agui"
API_KEY = "<your-token>"

def run_agent(thread_id, run_id, message, resume=None,
forwarded_props=None):
    agent = HttpAgent(
        base_url=BASE_URL,
        api_key=API_KEY,
    )

    input_params = RunAgentInput(
        thread_id=thread_id,
        run_id=run_id,
        messages=[Message(role="user", content=message)],
    )
    if forwarded_props:
        input_params.forwarded_props = forwarded_props
    if resume:
```

```
        input_params.resume = resume

    interrupts = []
    for event in agent.run(input_params):
        etype = event.type
        if etype == "TEXT_MESSAGE_CONTENT":
            print(event.delta, end="", flush=True)
        elif etype == "TOOL_CALL_START":
            print(f"\n[工具] {event.tool_call_name}")
        elif etype == "RUN_FINISHED":
            outcome = getattr(event, "outcome", None)
            if outcome and outcome.type == "interrupt":
                interrupts = outcome.interrupts
                for i, intr in enumerate(interrupts):
                    print(f"\n[中断 {i + 1}] {intr.message} (id:
{intr.id})")
        elif etype == "RUN_ERROR":
            print(f"\n[错误] {event.message}")
    return interrupts

# 1. 首次对话
interrupts = run_agent(
    thread_id="thread-001",
    run_id="run-001",
    message="搜索海浪视频并裁剪前10秒",
    forwarded_props={
        "approval_mode": "level:low",
        "scenario_name": "media_management",
    },
)

# 2. 每工具独立决策 — 批准第一个，拒绝第二个
if interrupts:
    print(f"\n共 {len(interrupts)} 个工具需要确认")
    resume = []
    for i, intr in enumerate(interrupts):
        if i == 0:
            resume.append(ResumeEntry(
                interrupt_id=intr.id,
                status="resolved",
            ))
```

```
        else:
            resume.append(ResumeEntry(
                interrupt_id=intr.id,
                status="cancelled",
                payload={"feedback": "不需要此操作"},
            ))

    # 3. 恢复（需携带相同的 approval_mode）
    run_agent(
        "thread-001", "run-002", "",
        resume=resume,
        forwarded_props={"approval_mode": "level:low"},
    )
```

Node.js / TypeScript（AG-UI TypeScript SDK）

安装：npm install @ag-ui/client @ag-ui/core

```
import { HttpAgent } from "@ag-ui/client";
import type { RunAgentInput, Interrupt } from "@ag-ui/core";

const BASE_URL = "https://smartmedia.vod-qcloud.com/agui";
const API_KEY = "<your-token>";

async function runAgent(params: {
  threadId: string;
  runId: string;
  message: string;
  forwardedProps?: Record<string, unknown>;
  resume?: Array<{
    interruptId: string;
    status: "resolved" | "cancelled";
    payload?: Record<string, unknown>;
  }>;
}): Promise<Interrupt[]> {
  const agent = new HttpAgent({
    serverUrl: BASE_URL,
    apiKey: API_KEY,
  });

  const input: RunAgentInput = {
```

```
    threadId: params.threadId,
    runId: params.runId,
    messages: [{ role: "user", content: params.message }],
    forwardedProps: params.forwardedProps,
    resume: params.resume,
  };

  const interrupts: Interrupt[] = [];

  // 订阅事件流
  const unsubscribe = agent.subscribe((event: any) => {
    switch (event.type) {
      case "TEXT_MESSAGE_CONTENT":
        process.stdout.write(event.delta || "");
        break;
      case "TOOL_CALL_START":
        console.log(`\n[Tool] ${event.toolCallName}`);
        break;
      case "RUN_FINISHED":
        if (event.outcome?.type === "interrupt") {
          interrupts.push(...event.outcome.interrupts);
          interrupts.forEach((intr, i) =>
            console.log(`\n[Interrupt ${i + 1}] ${intr.message} (id:
${intr.id})`)
          );
        }
        break;
      case "RUN_ERROR":
        console.error(`\n[Error] ${event.message}`);
        break;
    }
  });

  // 运行 Agent
  await agent.run(input);
  unsubscribe();
  return interrupts;
}

// 使用示例：每工具独立审批
async function main() {
```

```
  // 1. 首次对话
  const interrupts = await runAgent({
    threadId: "thread-001",
    runId: "run-001",
    message: "搜索海浪视频并裁剪前10秒",
    forwardedProps: {
      approval_mode: "level:low",
      scenario_name: "media_management",
    },
  });

  // 2. 每工具独立决策 — 批准第一个，拒绝第二个
  if (interrupts.length > 0) {
    console.log(`\n共 ${interrupts.length} 个工具需要确认`);
    await runAgent({
      threadId: "thread-001",
      runId: "run-002",
      message: "",
      forwardedProps: { approval_mode: "level:low" },
      resume: interrupts.map((intr, idx) => ({
        interruptId: intr.id,
        status: idx === 0 ? "resolved" as const : "cancelled" as const,
        payload: idx === 0 ? undefined : { feedback: "不需要此操作" },
      })),
    });
  }
}

main();
```

cURL

```
# 启动对话（可能触发多工具中断）
curl -N -X POST https://smartmedia.vod-qcloud.com/agui/chat \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer <token>" \
  -d '{
    "threadId": "thread-001",
    "runId": "run-001",
    "messages": [{"role": "user", "content": "搜索海浪视频并裁剪前10秒"}],
```

```
    "forwardedProps": {
      "approval_mode": "level:low",
      "scenario_name": "media_management"
    }
  }'

# 恢复 — 每工具独立决策
curl -N -X POST https://smartmedia.vod-qcloud.com/agui/chat \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer <token>" \
  -d '{
    "threadId": "thread-001",
    "runId": "run-002",
    "messages": [{"role": "user", "content": ""}],
    "forwardedProps": {"approval_mode": "level:low"},
    "resume": [
      {"interruptId": "lineage-uuid:ckpt-uuid:call-001", "status":
"resolved"},
      {"interruptId": "lineage-uuid:ckpt-uuid:call-002", "status":
"cancelled", "payload": {"feedback": "不需要裁剪"}}
    ]
  }'

# 查询历史
curl -X POST https://smartmedia.vod-qcloud.com/agui/history \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer <token>" \
  -d '{"threadId": "thread-001"}'
```

8. 事件处理建议

```
RUN_STARTED              → UI: 显示"正在思考..."
REASONING_START          → UI: 开始推理区域
REASONING_MESSAGE_*      → UI: 展示推理过程（可折叠）
REASONING_END            → UI: 结束推理区域
TEXT_MESSAGE_*           → UI: 流式渲染回复
TOOL_CALL_START/END      → UI: 显示工具调用卡片
TOOL_CALL_RESULT         → UI: 展示工具结果
RUN_FINISHED             → 检查 outcome:
                              无 outcome → 正常结束
```

```
                                type=interrupt → 进入 HITL 确认流程
RUN_ERROR                       → UI: 显示错误
```

附录：Prompt 指南

以下仅为示例 Prompt，请根据实际场景需要撰写提示词。

场景	Prompt 示例
电商带货	帮我根据提供的素材剪一条抖音竖屏带货短视频，时长30秒左右，主推产品是 XXX，突出 XXX 卖点。你来写口播文案、配 AI 配音-1.2倍速、加背景音乐、去掉素材里的原声、加字幕-字号50px，描边3px，节奏明快一点，结尾引导下单。
商品营销	文案我已经写好，请严格按这段文案配画面、配音、烧字幕，不要改文案： 「熬夜脸、暗沉、干纹？这瓶精华一次搞定。第三代玻色因，早晚各一次，28天肉眼可见提亮。现在下单送同款眼霜。」 平台是小红书竖屏，时长跟文案时长匹配即可，画面要对上每句话讲的内容。
活动宣传	帮我混剪成一条45秒的活动宣传片，突出 XXX，配 AI 配音和大气一点的背景音乐，加字幕。注意所有素材都是横屏 16:9，保持画幅统一。
解说二创	这是一部关于 XXX 的纪录片，帮我做成「3分钟看懂」的解说二创：你来提炼重点、重新写解说脚本、配 AI 配音，画面从原片里挑对应的镜头，配上字幕，去掉素材里的原声。风格轻松科普一点。注意只用素材里真实出现的画面，不要编造原片没有的内容。
高光集锦	帮我挑出最精彩的高能时刻，剪成一条60秒的高光集锦，保留原声，节奏紧凑一点，加字幕。