

L3 — 数据流与 Skill 体系层

文档版本: 1.0 适用项目: sgClaw (业数融合一平台 AI Agent 底座) 编制日期: 2026-03-03

读者: 高级开发者、Skill 开发者 —— 需要理解 Agent 内部数据流转、编写业务 Skill、集成 LLM、调优感知层和记忆系统。

1. 核心数据流时序

1.1 端到端数据流全景

一次完整的用户任务执行涉及以下数据流转路径。从用户在 Side Panel 输入自然语言指令开始，到最终任务完成结果返回，数据流经前端 → C++ → Pipe → Rust → LLM → Pipe → C++ → DOM。



└─ MAC Whitelist Check 校验

└─ CommandRouter → CdpCommandExecutor → 页面 DOM 操作

响应路径：

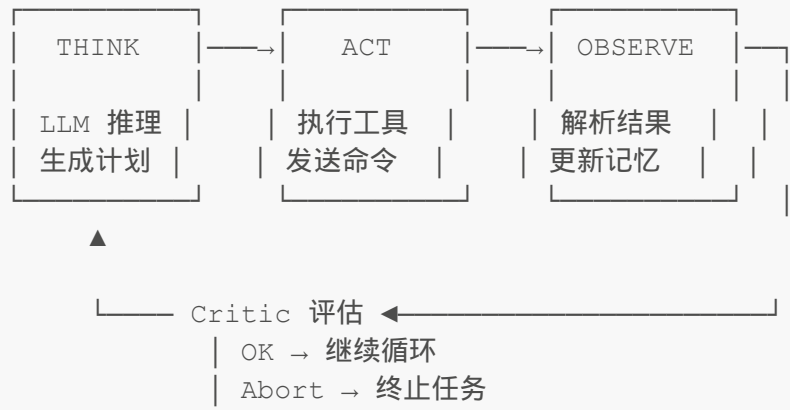
CommandRouter result → PipeWriter → sgClaw

+ AOM Snapshot （操作后的页面状态快照）

STDIO Pipe (JSON Line)

sgClaw Rust Process

Agent Runtime (ReAct Loop)



数据交互：

└─ BrowserPipeTool → Pipe → Browser (command)

└─ LLM Provider → HTTP(S) → Claude / GPT / Ollama

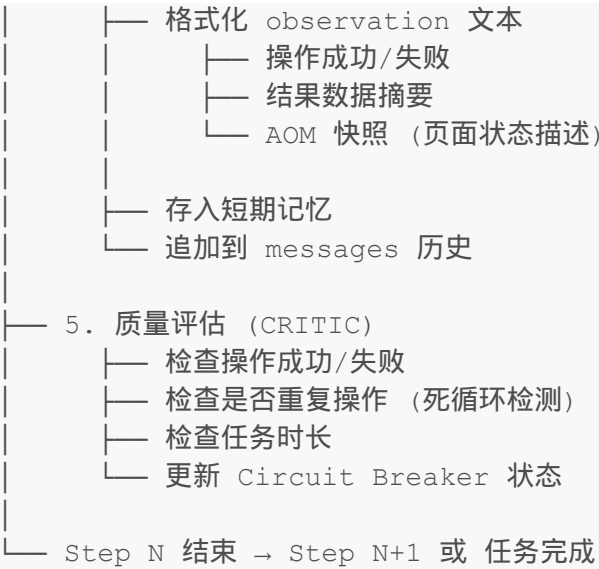
└─ Memory → SQLite (本地文件)

└─ SkillLoader → 文件系统 (技能脚本)

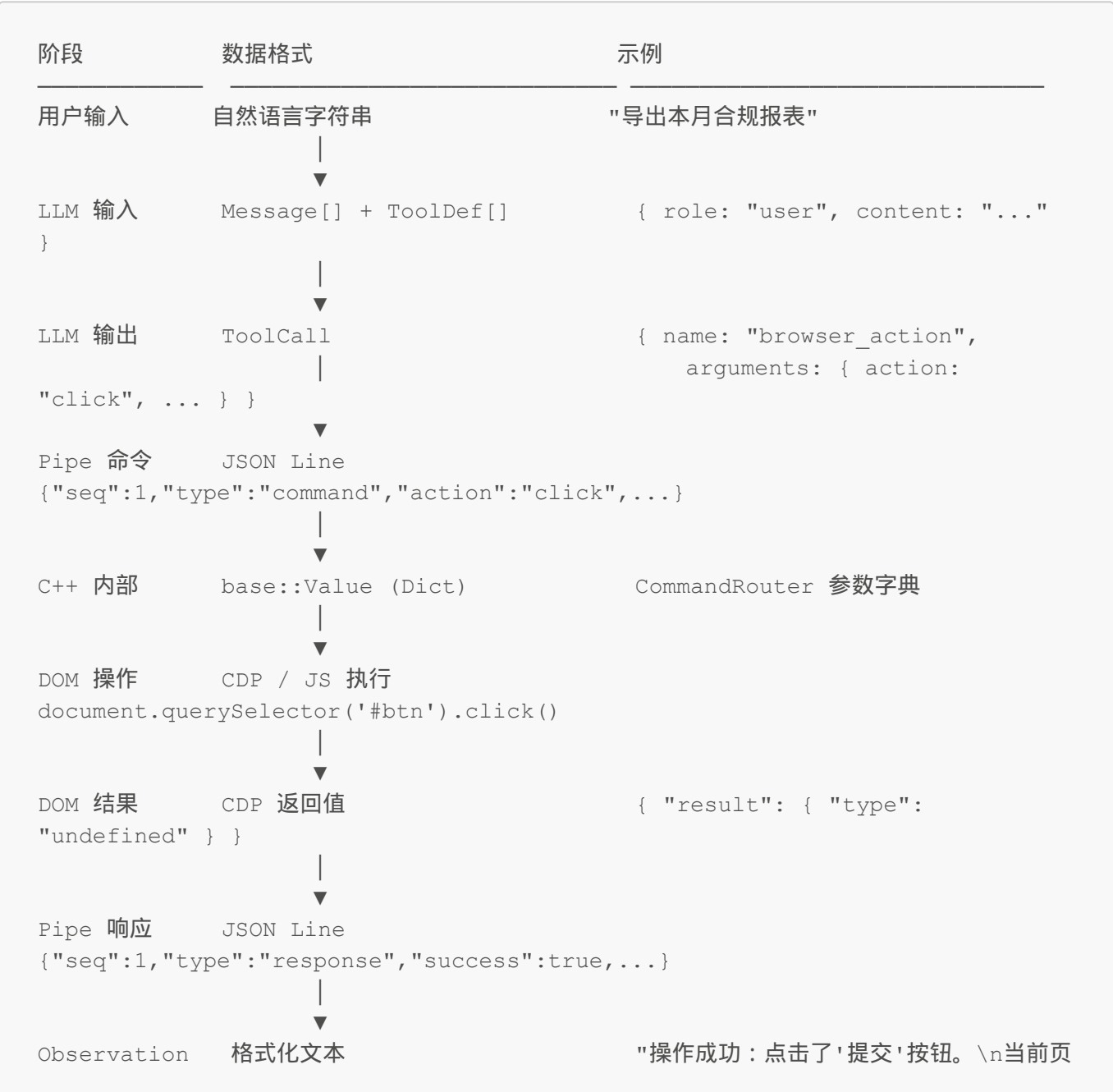
1.2 单步数据流详解

Agent 每一步（ReAct loop 的一次迭代）的数据流如下：





1.3 数据格式在各阶段的变换

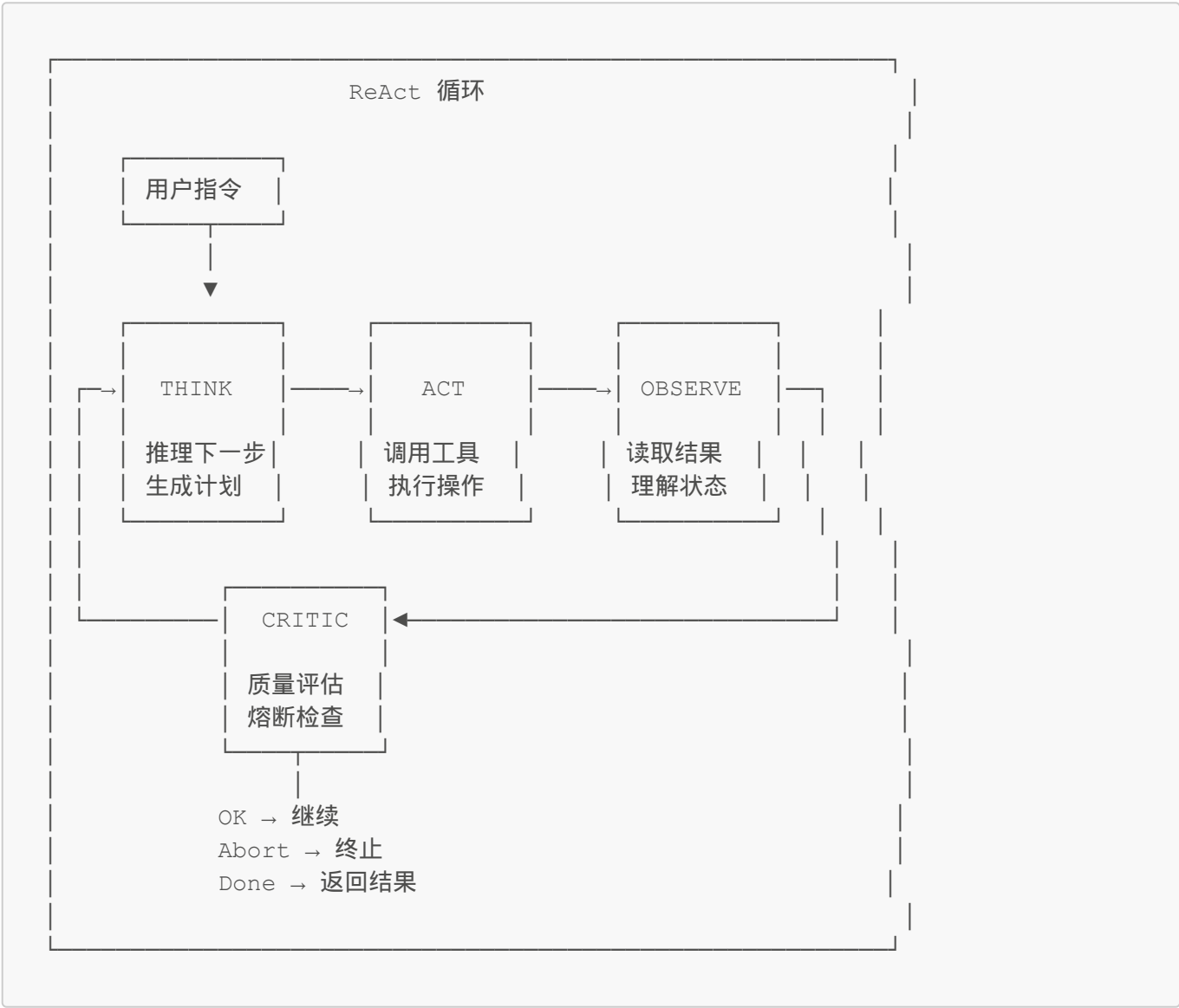




2. Agent 循环详解

2.1 ReAct 循环模型

sgClaw 采用 ReAct（Reasoning + Acting）循环模型，这是当前 AI Agent 领域最成熟的执行范式。核心思想：让 LLM 交替进行推理（Reasoning）和行动（Acting），每次行动后观察结果，再决定下一步。



2.2 System Prompt 模板

Agent 的行为由 System Prompt 定义。以下是 sgClaw 的 System Prompt 模板结构：

你是 sgClaw，一个运行在 SuperRPA 浏览器中的 AI 助手。你的任务是帮助用户在企业业务系统中完成自动化操作。

```
## 身份与角色
- 你运行在国家电网"业数融合一平台"的 SuperRPA 浏览器中
- 你可以操控浏览器中打开的业务系统页面 (ERP、OA、财务、HR 等)
- 你的每个操作都会被审计记录, trace_id: {{trace_id}}

## 可用工具
你有一个核心工具 `browser_action`, 支持以下操作:
{{#each tools}}
### {{this.name}}
{{this.description}}
参数: {{this.parameters}}
{{/each}}

## 可用技能
以下是预置的业务技能脚本, 当任务匹配时优先使用:
{{#each skills}}
- **{{this.name}}** (v{{this.version}}): {{this.description}}
  适用域名: {{this.domains}}
{{/each}}

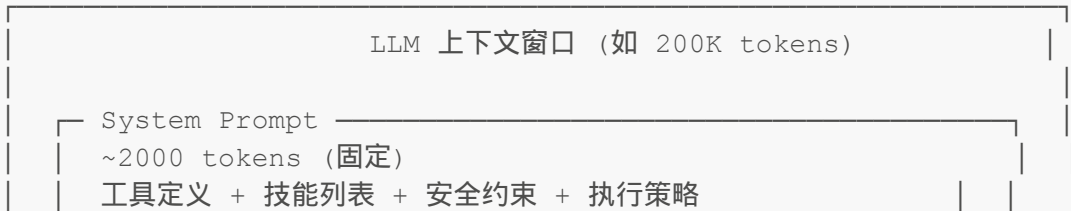
## 安全约束
1. 只能操作以下域名的页面: {{allowed_domains}}
2. 每个 browser_action 调用必须指定 expected_domain
3. 禁止尝试执行 eval、executeJsInPage 等操作
4. 涉及登录、登出、清空存储的操作会请求用户确认
5. 不要尝试读取或猜测用户密码

## 执行策略
1. 先观察当前页面状态 (getAomSnapshot)
2. 制定执行计划, 分步骤完成
3. 每步操作后检查结果
4. 遇到异常时尝试恢复 (最多重试 3 次)
5. 任务完成后提供简明的执行摘要

## 输出格式
- 思考过程用自然语言描述
- 操作通过 tool_call 执行
- 最终结果用中文文本总结
```

2.3 消息历史管理策略

Agent 的 LLM 调用需要发送对话历史 (messages)，但上下文窗口有限。sgClaw 采用分层记忆策略管理消息历史：





Token 预算分配:

区域	Token 预算	策略
System Prompt	~2000	固定, 编译时确定
长期记忆检索	~1000	按语义相关度排序取 top-3
短期对话记忆	剩余空间	最近步骤完整保留, 早期步骤压缩
用户指令	~100	原文保留
LLM 生成预留	~4096	max_tokens 参数

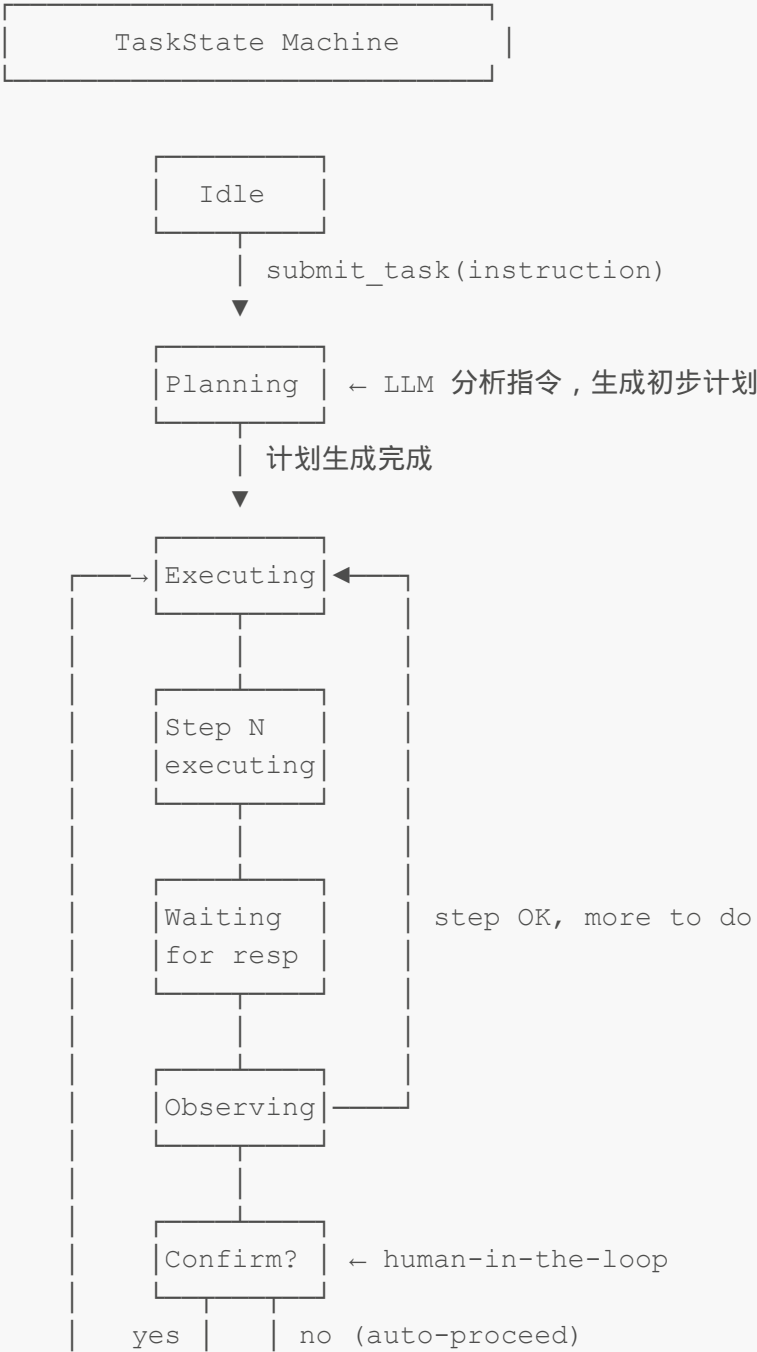
短期记忆截断算法:

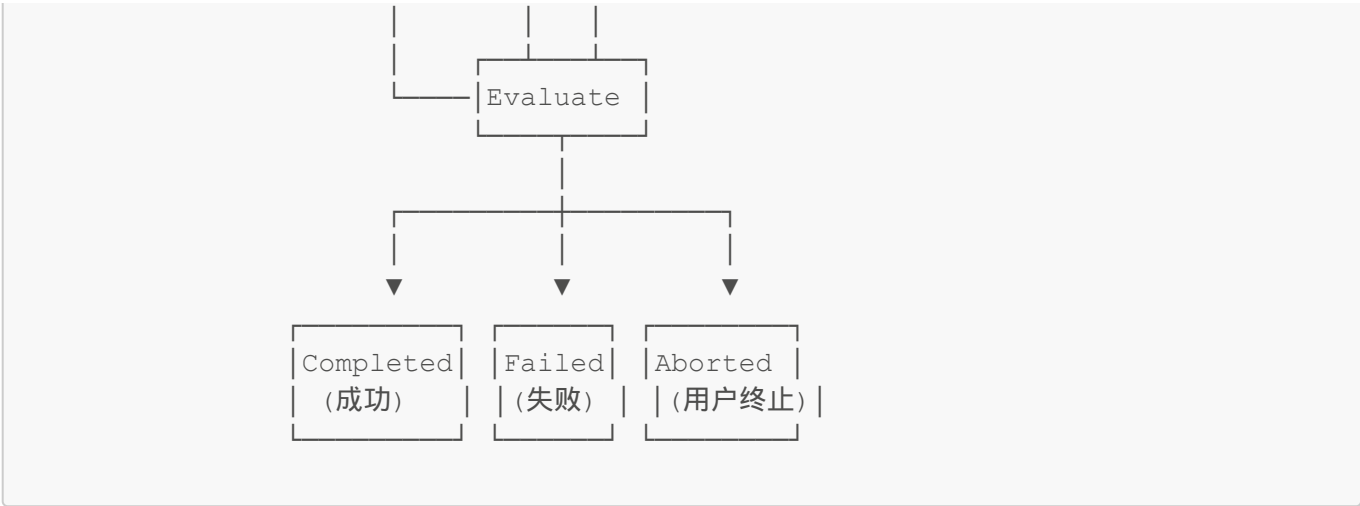
```
truncate_messages(messages, token_budget):  
    // 1. 保留最近 5 步完整记录  
    recent = messages[-5:]  
    recent_tokens = count_tokens(recent)  
  
    // 2. 如果最近 5 步已超预算, 减少保留数
```

```
while recent_tokens > token_budget * 0.8 and len(recent) > 2:
    recent = recent[-len(recent)+1:]
    recent_tokens = count_tokens(recent)

// 3. 对更早的步骤生成压缩摘要
remaining_budget = token_budget - recent_tokens
older = messages[:-len(recent)]
if older:
    summary = compress_steps(older, remaining_budget)
    return [summary_message] + recent
else:
    return recent
```

2.4 任务生命周期状态机





3. Skill 体系

3.1 Skill 定义格式

每个 Skill 是一个 JavaScript 文件，包含元数据头和执行函数。

完整 Skill 文件结构：

```
/**
 * @skill  erp-monthly-report
 * @version 1.0.0
 * @description 从ERP系统导出月度财务报表。支持按部门、科目筛选，
 *              自动处理分页数据，合并导出为完整报表。
 * @domains  erp.example.com, erp-test.example.com
 * @author   sgClaw Team
 * @params {
 *   "type": "object",
 *   "required": ["month"],
 *   "properties": {
 *     "month": {
 *       "type": "string",
 *       "pattern": "^\\d{4}-\\d{2}$",
 *       "description": "报表月份 (如 2026-03)"
 *     },
 *     "department": {
 *       "type": "string",
 *       "description": "部门名称 (可选，不填则导出全部)"
 *     },
 *     "format": {
 *       "type": "string",
 *       "enum": ["xlsx", "csv", "pdf"],
 *       "default": "xlsx",
 *       "description": "导出格式"
 *     }
 *   }
 * }
 */
```

```
/**
 * 技能执行入口
 * @param {object} params - 输入参数 (符合上方 @params 定义)
 * @param {function} browserAction - BrowserAction 调用函数
 * @returns {object} 执行结果 { success: boolean, data?: any, error?: string }
 */
async function execute(params, browserAction) {
    const { month, department, format = 'xlsx' } = params;

    try {
        // Step 1: 导航到 ERP 报表页面
        await browserAction('navigate',
            'https://erp.example.com/report/finance');
        await browserAction('waitForSelector', '.report-filter', 5000);

        // Step 2: 设置筛选条件
        await browserAction('click', '#month-picker');
        await browserAction('type', '#month-input', month);

        if (department) {
            await browserAction('click', '#dept-selector');
            await browserAction('type', '#dept-search', department);
            await browserAction('click', `.${dept-option[data-
name="${department}"]`);
        }

        // Step 3: 选择导出格式
        await browserAction('select', '#export-format', format);

        // Step 4: 点击导出
        await browserAction('click', '#export-btn');

        // Step 5: 等待导出完成
        await browserAction('waitForSelector', '.export-success', 30000);

        // Step 6: 获取结果信息
        const resultText = await browserAction('getText', '.export-
result');

        return {
            success: true,
            data: {
                message: resultText,
                month: month,
                department: department || '全部',
                format: format
            }
        };
    } catch (error) {
        return {
            success: false,
```

```
        error: `导出报表失败: ${error.message}`
      };
    }
  }
}
```

3.2 Skill 仓库结构

```
sgclaw-skills/
├── registry.json          # 技能清单 (签名 + 哈希索引)
├── builtin/              # 内置技能 (随产品交付)
│   ├── erp-monthly-report.js    # ERP 月度报表导出
│   ├── erp-anomaly-check.js    # ERP 异常交易检查
│   ├── oa-approval.js          # OA 审批单处理
│   ├── oa-meeting-schedule.js  # OA 会议日程管理
│   ├── finance-compliance.js   # 财务合规线索提报
│   ├── finance-reconciliation.js # 财务对账
│   ├── hr-social-insurance.js  # 人力社保申报
│   ├── hr-salary-check.js      # 薪酬数据核验
│   ├── legal-contract-monitor.js # 合同履约监测
│   └── cross-system-sync.js     # 跨系统数据同步
├── custom/               # 用户自定义技能
│   └── (用户创建的 .js 文件)
└── keys/
    └── skill_verify.pub      # Ed25519 公钥 (校验签名)
```

registry.json 格式:

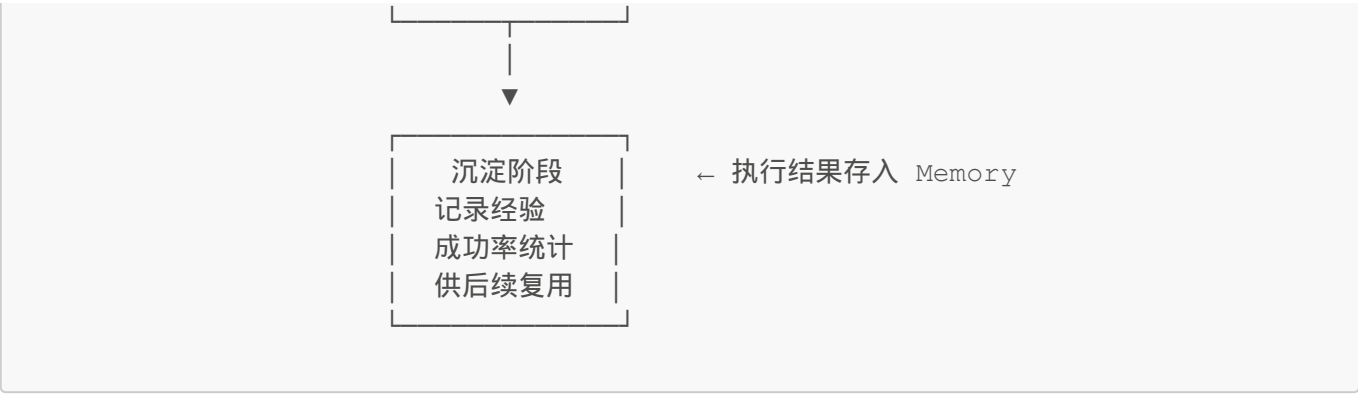
```
{
  "version": "1.0",
  "updated_at": "2026-03-03T00:00:00Z",
  "skills": [
    {
      "name": "erp-monthly-report",
      "file": "builtin/erp-monthly-report.js",
      "version": "1.0.0",
      "hash": "sha256:a1b2c3d4e5f6...",
      "signature": "ed25519:x7y8z9...",
      "enabled": true
    },
    {
      "name": "oa-approval",
      "file": "builtin/oa-approval.js",
      "version": "1.2.0",
      "hash": "sha256:f6e5d4c3b2a1...",
      "signature": "ed25519:z9y8x7...",
      "enabled": true
    }
  ]
}
```

```
]
}
```

3.3 Skill 生命周期



← Agent ReAct 循环中 LLM 选择使用



3.4 从现有 JS 场景代码迁移到 Skill

SuperRPA 已有大量 agent-vue 中的 JS 场景代码（49 个文件，208+ 个调用点）。这些代码可以迁移为 sgClaw Skill，实现复用。

迁移策略：

阶段	内容	说明
Phase 1: 直接复用	现有 JS 代码通过 BrowserAction API 调用	sgClaw 发送 pipe 命令 → Browser 执行 → JS SDK 代码无需改动
Phase 2: 技能化封装	将高频场景代码提取为 Skill	添加元数据头、参数 schema、错误处理
Phase 3: LLM 增强	Skill 作为 Agent 的工具选项	LLM 根据用户指令自动选择合适的 Skill

迁移示例：将 agent-vue 中的 OA 审批代码迁移为 Skill

现有代码（agent-vue 中）：

```
// agent-vue/src/scenes/oa-approval.js
async function approveAll() {
  await
  sgBrowser.page.navigate('https://oa.example.com/approval/pending');
  await sgBrowser.page.waitForSelector('.approval-list');
  const items = await sgBrowser.page.getText('.approval-list .item');
  for (const item of items) {
    await sgBrowser.input.click(`.item[data-id="${item.id}"] .approve-
btn`);
    await sgBrowser.page.waitForSelector('.confirm-dialog');
    await sgBrowser.input.click('.confirm-dialog .ok-btn');
  }
}
```

迁移后的 Skill：

```

/**
 * @skill oa-approval
 * @version 1.0.0
 * @description OA系统待审批单据批量处理。支持查看待审批列表、
 *              批量审批、批量驳回、添加审批意见。
 * @domains oa.example.com, oa-test.example.com
 * @params {
 *   "type": "object",
 *   "required": ["action"],
 *   "properties": {
 *     "action": { "enum": ["list", "approve_all", "reject", "approve_one"]
 *   },
 *   "item_id": { "type": "string", "description": "单据ID
 *               (approve_one/reject 时需要)" },
 *   "opinion": { "type": "string", "description": "审批意见 (可选)" }
 * }
 */
async function execute(params, browserAction) {
  const { action, item_id, opinion } = params;

  // 导航到审批页面
  await browserAction('navigate',
    'https://oa.example.com/approval/pending');
  await browserAction('waitForSelector', '.approval-list', 5000);

  switch (action) {
    case 'list': {
      const text = await browserAction('getText', '.approval-list');
      return { success: true, data: { items: text } };
    }
    case 'approve_all': {
      // 获取所有待审批项
      const snapshot = await browserAction('getAomSnapshot',
        '.approval-list');
      let count = 0;

      // 逐个审批 (带错误处理)
      for (const item of snapshot) {
        try {
          await browserAction('click', `.item[data-
            id="${item.name}"] .approve-btn`);
          await browserAction('waitForSelector', '.confirm-
            dialog', 3000);

          if (opinion) {
            await browserAction('type', '.opinion-input',
              opinion);
          }
          await browserAction('click', '.confirm-dialog .ok-
            btn');
          await browserAction('waitForSelector', '.success-
            toast', 3000);
          count++;
        }
      }
    }
  }
}

```

```
        } catch (e) {
            // 单条失败不影响整体
            continue;
        }
    }

    return { success: true, data: { approved: count, total:
snapshot.length } };
}
// ... 其他 action
}
}
```

迁移改动点总结：

改动项	说明
添加元数据头	@skill, @version, @description, @domains, @params
函数签名统一	async function execute(params, browserAction)
API 调用方式	sgBrowser.page.navigate(url) → browserAction('navigate', url)
错误处理	添加 try/catch，返回统一格式 { success, data, error }
参数化	硬编码值改为 params 输入

3.5 Skill 执行沙箱

Skill 的 JS 脚本需要在受限环境中执行，防止恶意代码突破安全边界。

沙箱约束：

能力	是否允许	说明
browserAction()	允许	唯一的外部交互方式，受 MAC 策略约束
console.log/error	允许	输出到 Agent 日志
JSON.parse/stringify	允许	数据处理必需
Promise / async-await	允许	异步控制流必需
setTimeout / setInterval	允许 (受限)	最大延迟 30s，用于等待
fetch / XMLHttpRequest	禁止	网络请求必须通过 browserAction
require / import	禁止	不允许加载外部模块
process / child_process	禁止	不允许访问系统进程
fs / path	禁止	不允许访问文件系统
eval / Function()	禁止	不允许动态代码执行

实现方案：

sgClaw 使用轻量级 JS 引擎（如 Boa 或 embedded V8 via Rusty_V8）运行 Skill 脚本，在创建执行上下文时仅注入允许的全局对象：

```
// Skill 沙箱执行（概念代码）
fn execute_skill(script: &str, params: Value, browser_tool:
&BrowserPipeTool) -> Result<Value> {
    let mut context = JsContext::new();

    // 仅注入允许的全局对象
    context.register_global("browserAction", |args| {
        // 代理到 BrowserPipeTool.execute()
        browser_tool.execute(args)
    });
    context.register_global("console", ConsoleProxy::new());
    context.register_global("JSON", JsonGlobal::new());
    context.register_global("Promise", PromiseGlobal::new());

    // 禁止所有其他全局对象
    // (JsContext 默认不提供 Node.js / Browser API)

    // 执行技能脚本
    context.eval(script)?;

    // 调用 execute 函数
    let result = context.call("execute", &[params, browser_action_ref])?;

    Ok(result)
}
```

4. 感知层数据格式

4.1 AOM（Accessibility Object Model）快照

AOM 快照是 Agent 理解页面状态的核心数据源。每次浏览器操作后，响应中附带当前页面的 AOM 快照，供 Agent 的 OBSERVE 阶段使用。

AOM 快照格式：

```
{
  "aom_snapshot": [
    {
      "role": "navigation",
      "name": "主导航栏",
      "bounds": [0, 0, 1920, 60],
      "children": [
        { "role": "link", "name": "首页", "bounds": [20, 10, 60, 40] },
        { "role": "link", "name": "财务报表", "bounds": [100, 10, 80, 40],
"focused": true },
        { "role": "link", "name": "系统设置", "bounds": [200, 10, 80, 40] }
      ]
    }
  ]
}
```



```
]
},
{
  "role": "main",
  "name": "报表筛选区",
  "bounds": [0, 60, 1920, 200],
  "children": [
    {
      "role": "combobox",
      "name": "月份选择",
      "value": "2026-03",
      "bounds": [20, 80, 200, 40],
      "selector": "#month-picker"
    },
    {
      "role": "combobox",
      "name": "部门筛选",
      "value": "",
      "bounds": [240, 80, 200, 40],
      "selector": "#dept-selector"
    },
    {
      "role": "button",
      "name": "导出报表",
      "bounds": [460, 80, 100, 40],
      "selector": "#export-btn",
      "disabled": false
    }
  ]
},
{
  "role": "table",
  "name": "报表数据",
  "bounds": [0, 260, 1920, 600],
  "row_count": 25,
  "children": [
    {
      "role": "row",
      "name": "表头",
      "children": [
        { "role": "columnheader", "name": "科目编号" },
        { "role": "columnheader", "name": "科目名称" },
        { "role": "columnheader", "name": "借方金额" },
        { "role": "columnheader", "name": "贷方金额" }
      ]
    }
  ]
}
]
```

字段	类型	说明
role	string	ARIA 角色 (button, link, textbox, table, etc.)
name	string	元素的可访问名称 (label text, aria-label, etc.)
bounds	[x, y, w, h]	元素的视口坐标和尺寸 (px)
value	string	元素当前值 (input, select 等)
selector	string	CSS 选择器 (供后续操作使用)
focused	boolean	是否获得焦点
disabled	boolean	是否禁用
checked	boolean	是否选中 (checkbox, radio)
children	array	子元素列表
row_count	number	表格行数 (仅 table 角色)

AOM 快照与 LLM 的关系：

AOM 快照被格式化为结构化文本后，作为 observation 的一部分发送给 LLM：

操作结果： 点击"财务报表"链接成功。

当前页面状态：

- 导航栏： 首页 | [财务报表] (当前) | 系统设置
- 报表筛选区：
 - 月份选择： 2026-03
 - 部门筛选： (未选择)
 - [导出报表] 按钮 (可用)
- 报表数据： 25行数据已加载
 - 表头： 科目编号 | 科目名称 | 借方金额 | 贷方金额

4.2 SoM (Set-of-Mark) 标注

在需要视觉辅助时，sgClaw 可以请求带 SoM 标注的页面截图。SoM 在页面截图上为每个 可交互元素叠加数字标签，便于 LLM 通过编号引用元素。

请求方式：

```
{
  "seq": 10,
  "type": "command",
  "action": "pageScreenshot",
  "params": {
    "full_page": false,
    "som_overlay": true
  },
}
```

```
"security": { "expected_domain": "erp.example.com", "hmac": "..."}
}
```

响应:

```
{
  "seq": 10,
  "type": "response",
  "success": true,
  "data": {
    "image_base64": "/9j/4AAQSkZJRg...",
    "som_labels": [
      { "id": 1, "selector": "#month-picker", "name": "月份选择", "bounds":
[20, 80, 200, 40] },
      { "id": 2, "selector": "#dept-selector", "name": "部门筛选", "bounds":
[240, 80, 200, 40] },
      { "id": 3, "selector": "#export-btn", "name": "导出报表", "bounds":
[460, 80, 100, 40] }
    ]
  }
}
```

LLM 使用 SoM 的方式:

LLM 看到截图后可以引用标签编号:

我看到页面上有以下可交互元素:

- [1] 月份选择 - 当前值 2026-03
- [2] 部门筛选 - 未选择
- [3] 导出报表按钮

我需要点击 [3] 导出报表按钮来导出数据。

SoM 使用策略:

场景	使用方式	优先级
正常操作	AOM 快照 (文本)	默认, token 开销低
AOM 不可用或不完整	SoM 截图	兜底方案
复杂视觉布局	AOM + SoM 结合	特殊场景
表格/图表分析	SoM 截图	视觉信息丰富时

4.3 感知数据选择策略

Agent 在 OBSERVE 阶段根据当前状态自动选择感知方式:



5. LLM 集成

5.1 Provider 适配层

sgClaw 通过 ZeroClaw 的 Provider trait 抽象对接不同的 LLM 服务。每个 Provider 实现 HTTP 调用、streaming 解析、错误重试等逻辑。

Provider 统一行为要求：

行为	要求	说明
超时	连接 10s，首 token 30s，总体 120s	防止挂起
重试	最多 3 次，指数退避 (1s, 2s, 4s)	仅重试 5xx 和网络错误
Streaming	必须支持	用于实时日志显示
Tool-use	必须支持	Agent ReAct 的核心能力
Token 计量	每次调用后上报 usage	审计和成本控制
错误分类	区分可重试/不可重试错误	401/403 不重试

5.2 Claude Provider 实现要点

```
pub struct ClaudeProvider {  
    client: reqwest::Client,  
    api_key: String,  
    model: String,  
    base_url: String,  
}
```

```
impl ClaudeProvider {
    const DEFAULT_BASE_URL: &'static str = "https://api.anthropic.com";
    const API_VERSION: &'static str = "2023-06-01";

    /// 将 sgClaw 的 Message 格式转换为 Claude API 格式
    fn convert_messages(messages: &[Message]) -> Vec<ClaudeMessage> {
        // Message::User → { role: "user", content: [...] }
        // Message::Assistant → { role: "assistant", content: [...] }
        // Message::Tool → { role: "user", content: [{ type: "tool_result",
        ... }] }
    }

    /// 将 sgClaw 的 ToolDefinition 转换为 Claude tool 格式
    fn convert_tools(tools: &[ToolDefinition]) -> Vec<ClaudeTool> {
        // ToolDefinition → { name, description, input_schema }
    }
}
```

5.3 输出约束

LLM 的输出必须遵循以下约束：

Tool-use 输出格式 (Claude API):

```
{
  "role": "assistant",
  "content": [
    {
      "type": "thinking",
      "thinking": "用户要导出本月合规报表。我需要先导航到 ERP 系统的报表模块..."
    },
    {
      "type": "tool_use",
      "id": "toolu_01xyz",
      "name": "browser_action",
      "input": {
        "action": "navigate",
        "params": {
          "url": "https://erp.example.com/report/compliance"
        },
        "expected_domain": "erp.example.com"
      }
    }
  ]
}
```

输出校验规则：

LLM 输出校验：

- 1. 是否为合法 JSON?
 - 否：请求 LLM 重新生成（最多 2 次）
- 2. tool_call 格式是否正确?
 - name 必须是 "browser_action" 或已注册的 MCP 工具
 - input 必须包含 action 字段
 - action 必须在 Action 枚举中
- 3. 参数是否通过 JSON Schema 校验?
 - 否：将校验错误信息反馈给 LLM，请求修正
- 4. 是否包含 expected_domain?
 - 否：自动从上下文推断（当前页面域名）
- 5. 通过所有校验
 - 交给 BrowserPipeTool 执行

5.4 System Prompt 中的工具描述

提供给 LLM 的工具描述需要精确、简洁，帮助 LLM 正确选择操作：

```
{
  "name": "browser_action",
  "description": "在浏览器中执行操作。可以点击元素、输入文本、导航页面、获取页面内容等。每次调用需要指定 action 类型和对应参数。",
  "input_schema": {
    "type": "object",
    "required": ["action", "expected_domain"],
    "properties": {
      "action": {
        "type": "string",
        "enum": ["click", "type", "navigate", "getText", "getHtml", "waitForSelector", "pageScreenshot", "select", "scrollTo", "getAomSnapshot", "storageSet", "storageGet", "zombieSpawn", "zombieKill"],
        "description": "要执行的操作类型"
      },
      "params": {
        "type": "object",
        "description": "操作参数，根据 action 类型不同而不同"
      },
      "expected_domain": {
        "type": "string",
        "description": "操作目标页面的域名（安全校验用）"
      }
    }
  }
}
```

6. 记忆与自进化

6.1 记忆分层架构



6.2 记忆读写流程

写入流程（任务执行过程中）：



- └─ 保存成功步骤序列 (type: SkillExperience)
- └─ 更新技能执行统计 (skill_executions 表)

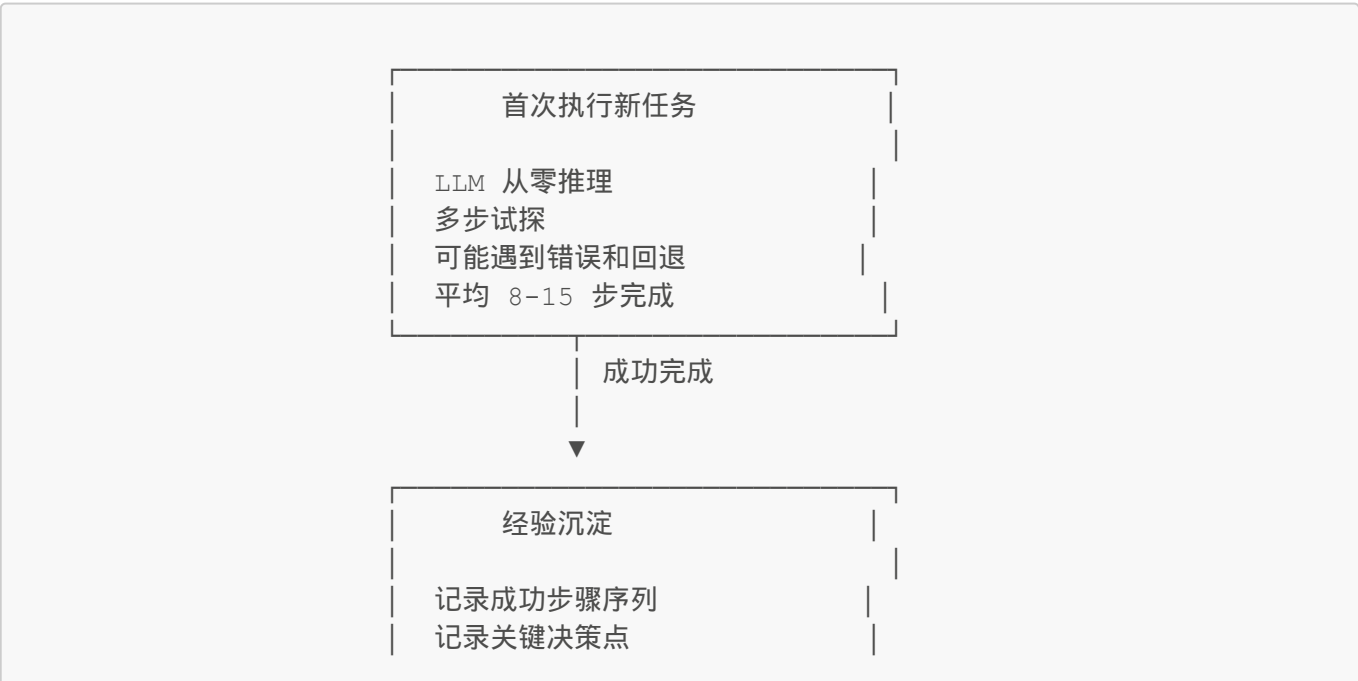
读取流程（新任务开始时）：

```
execute_task(instruction):  
└─ 1. 清空短期记忆（新任务）  
└─ 2. 从长期记忆检索相关经验  
    └─ 语义搜索: embedding(instruction) vs memory_entries  
    └─ 取 top-3 相关条目  
    └─ 格式化为 "历史经验" 上下文  
└─ 3. 检查是否有精确匹配的 Skill 经验  
    └─ 查询 skill_executions 表  
    └─ 找到成功执行记录?  
        → 作为推荐步骤加入 system_prompt  
    └─ 未找到?  
        → 不影响执行  
└─ 4. 组装初始上下文  
    └─ system_prompt + 长期记忆检索结果  
    └─ 用户指令  
    └─ 开始 ReAct 循环
```

6.3 自进化学习机制

sgClaw 的自进化核心思想：每次成功执行的任务都是一个学习样本。通过记录和检索这些样本，Agent 在后续遇到相似任务时可以更快、更准确地完成。

自进化循环：





经验记录格式:

```
{
  "id": "exp-2026-03-03-001",
  "type": "task_result",
  "content": "任务: 导出本月ERP合规报表\n\n执行路径:\n1. navigate →\nerp.example.com/report\n2. click → #month-picker → 设置为2026-03\n3. click →\n#compliance-tab\n4. click → #export-btn\n5. waitForSelector → .export-\nsuccess\n\n结果: 成功导出, 文件名 compliance-2026-03.xlsx",
  "metadata": {
    "task_instruction": "导出本月合规报表",
    "steps": 5,
    "duration_ms": 12500,
    "domains_visited": ["erp.example.com"],
    "skills_used": [],
    "success": true
  },
  "embedding": [0.12, -0.34, 0.56, ...],
  "created_at": "2026-03-03T10:30:00Z",
  "session_id": "trace-abc123"
}
```

6.4 向量检索实现

长期记忆的语义检索使用简单的余弦相似度计算。在数据量较小时（< 10000 条），线性扫描足够快（< 10ms）。

嵌入向量生成:

文本 → 嵌入方式:

├ 在线模型可用时:

├ 调用 LLM Provider 的 embedding API

├ Claude: 不提供独立 embedding API, 使用 summarize + hash

├ OpenAI: text-embedding-3-small (1536 维)

└ Ollama: nomic-embed-text (384 维)

├ 离线/无 embedding API 时:

├ 简单 TF-IDF 向量 + 关键词匹配

├ 中文分词 (jieba-rs)

├ 计算词频向量

└ 余弦相似度排序

检索优化策略:

策略	说明	适用场景
类型过滤	先按 entry_type 过滤，减少扫描量	总是使用
时间窗口	优先检索最近 30 天的记忆	默认策略
域名过滤	按当前任务涉及的域名过滤	域名已知时
混合排序	0.7 * 向量相似度 + 0.3 * 时间衰减	综合排序

7. 审计与追溯

7.1 Trace ID 体系

每次 Agent 会话分配唯一的 `trace_id`，贯穿所有模块和组件，用于事后审计追溯。

trace_id 格式:

sgclaw-{date}-{random}

示例: sgclaw-20260303-a1b2c3d4

trace_id 传播路径:

SgClawProcessHost::Start()

├ 生成 trace_id

├ → init message: { trace_id: "sgclaw-20260303-a1b2c3d4" }

├ → sgClaw Runtime: 所有日志附带 trace_id

└ → LLM 调用记录

- └→ Pipe 命令记录
 - └→ Memory 条目关联
- └→ PipeListener: 所有 pipe 消息日志附带 trace_id
- └→ CommandRouter: 操作审计日志附带 trace_id

7.2 审计日志格式

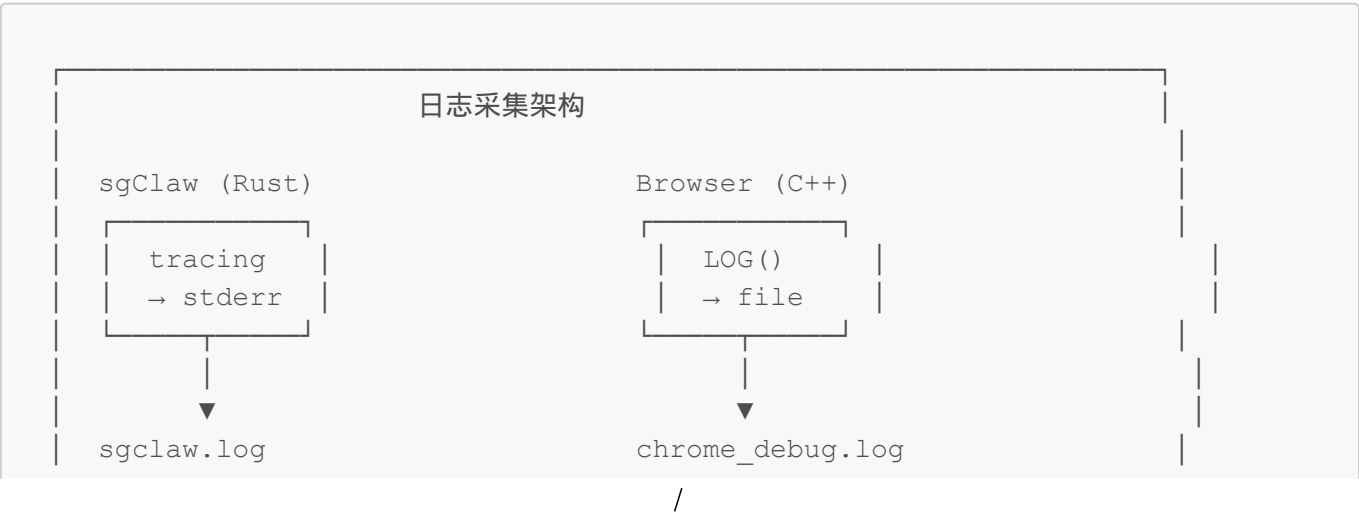
sgClaw 端日志 (输出到 stderr, 结构化 JSON):

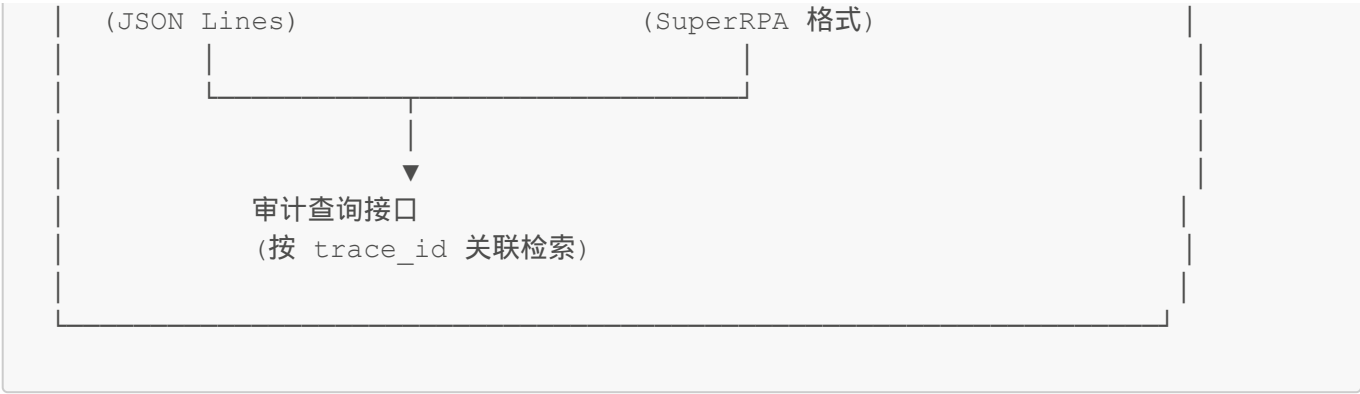
```
{
  "timestamp": "2026-03-03T10:23:05.123Z",
  "level": "INFO",
  "trace_id": "sgclaw-20260303-a1b2c3d4",
  "module": "agent::runtime",
  "event": "step_completed",
  "data": {
    "step": 3,
    "action": "click",
    "selector": "#export-btn",
    "domain": "erp.example.com",
    "success": true,
    "duration_ms": 245
  }
}
```

Browser 端日志 (SuperRPA 已有日志系统):

```
[2026-03-03 10:23:05.123] [sgclaw] [trace:sgclaw-20260303-a1b2c3d4]
PIPE_CMD seq=3 action=click selector=#export-btn domain=erp.example.com
MAC_CHECK: ALLOW
CMD_EXEC: OK (245ms)
```

7.3 日志采集与存储





日志保留策略：

日志类型	保留时长	存储位置
sgClaw 操作日志	90 天	本地文件 + 可选远程上传
Pipe 通信记录	30 天	本地文件
LLM 调用记录	90 天	本地文件 (含 token 用量)
长期记忆 (SQLite)	永久	本地数据库
浏览器审计日志	按已有策略	SuperRPA 日志系统

文档结束。本文档为 sgClaw L3 层数据流与 Skill 体系参考。Skill 开发者应重点关注 第 3 节（Skill 体系）和 第 4 节（感知层数据格式），高级开发者应关注第 2 节（Agent 循环）和 第 6 节（记忆与自进化）。