HTTP API 接口文档

概述

本文档描述了数字奇点 HTTP 服务的所有 API 接口。所有接口都支持 CORS，并返回 JSON 格式数据。

重要说明：

- 大部分接口需要先调用 `101.43.119.131:20717/api/getNonce` 获取 nonce
- 加密接口需要先调用 `101.43.119.131:20717/api/getServerPublicKey` 获取服务器公钥，服务器公钥只需要获取一次，本地保存即可。
- 所有接口都支持 OPTIONS 预检请求

基础接口

1. 获取 Nonce

接口地址： `POST /api/getNonce`

说明： 获取用于防重放攻击的一次性随机数。在调用其他接口前必须先获取 nonce。

请求参数： 无

响应示例：

```
{
  "status": "success",
  "data": {
    "nonce": "随机字符串"
  }
}
```

---

2. 获取服务器公钥

接口地址： `POST /api/getServerPublicKey`

说明： 获取服务器 RSA 公钥，用于加密请求数据。使用加密接口前必须先获取此公钥。

请求参数： 无

响应示例：

```
{
  "status": "success",
  "serverPublicKey": "16进制的公钥字符串"
}
```

用16进制解码后可以得到完整的pem，用来加密客户端发送给服务器的数据。

认证与授权

3. 加密请求接口

接口地址： `POST /api/encryptedRequest`

说明： 通用加密请求接口，支持多种业务类型。请求和响应都使用 RSA 加密。

请求格式：

```
{
  "ciphertext": "加密后的数据（16进制字符串）"
}
```

加密前的数据格式：

```
{
  "type": "command|query|event",
  "nonce": "从 /api/getNonce 获取",
  "userPublicKeyBase64": "用户公钥（可选，用于加密响应）",
  // ... 其他业务字段
}
```

支持的请求类型：

3.1 命令类请求 (type: "command")

- **登录** (`action: "login"`)
  - 需要字段：`phone`, `smsCode`, `userPublicKeyBase64`（可选）
  - 返回：token 和用户信息
- **发送验证码** (`action: "sendSmsCode"`)
  - 需要字段：`phone`
  - 返回：发送结果
- **登出** (`action: "logout"`)
  - 功能：登出（暂未实现）

响应格式：

- 如果提供了 `userPublicKeyBase64`，响应会被加密：

```
{
  "ciphertext": "加密后的响应数据"
}
```

- 否则返回明文 JSON

---

4. 普通请求接口

接口地址： `POST /api/plainRequest`

说明： 与加密请求接口功能相同，但请求和响应都是明文 JSON，不加密。

请求格式：

```
{
  "type": "command|query|event",
  "nonce": "从 /api/getNonce 获取",
  // ... 其他业务字段
}
```

响应格式： 直接返回 JSON 结果

---

5. 发送验证码

接口地址： `POST /api/captcha`

说明： 发送短信验证码（加密接口）

请求格式：

```
{
  "ciphertext": "加密后的数据"
}
```

加密前的数据格式：

```
{
  "nonce": "从 /api/getNonce 获取",
  "mobile_number": "手机号码"
}
```

响应示例：

```
{
  "status": "success",
  "message": "验证码已发送"
}
```

速率限制： 每个 IP 每 60 秒最多 3 次请求

API 密钥管理

6. 获取 API 密钥列表

接口地址： `POST /api/apikey/list`

说明： 获取当前用户的所有 API 密钥（不包括已删除的）

请求格式：

```
{
  "token": "用户认证 token"
}
```

响应示例：

```
{
  "status": "success",
  "data": [
    {
      "id": "密钥ID",
      "name": "密钥名称",
      "status": 0,  // 0=禁用, 1=启用, 2=删除
      "created_at": "创建时间"
    }
  ]
}
```

---

7. 创建 API 密钥

接口地址： `POST /api/apikey/create`

说明： 创建新的 API 密钥

请求格式：

```
{
  "token": "用户认证 token",
  "name": "密钥名称"
}
```

响应示例：

```
{
  "status": "success",
  "data": {
    "id": "密钥ID",
    "key": "密钥值（仅创建时返回）",
    "name": "密钥名称"
  }
}
```

---

8. 删除 API 密钥

接口地址： `POST /api/apikey/delete`

说明： 删除指定的 API 密钥

请求格式：

```
{
  "token": "用户认证 token",
  "id": "密钥ID"
}
```

响应示例：

```
{
  "status": "success",
  "message": "删除成功"
}
```

---

9. 更新 API 密钥状态

接口地址： `POST /api/apikey/update-status`

说明： 启用或禁用 API 密钥

请求格式：

```
{
  "token": "用户认证 token",
  "id": "密钥ID",  // 或使用 "key_id"
  "status": 1  // 0=禁用, 1=启用
}
```

响应示例：

```
{
  "status": "success",
  "message": "状态更新成功"
}
```

AI 交互接口

10. 外部模型交互

接口地址： `POST 43.163.26.190:20717/api/interactive/external`

说明： 与外部模型（转发到 43 服务器）进行交互，请求需要加密

请求格式：

```
{
  "ciphertext": "加密后的数据"
}
```

加密前的数据格式：

```
{
  "type": "chat",
  "data": "JSON字符串，包含以下字段（见下方说明）"
}
```

data 字段（JSON字符串）的完整格式：

```
{
  "model": "模型名称（如 deepseek-chat、gpt-4o、claude-3-7-sonnet-20250219
等）",
  "api_key": "API密钥（可选，用于外部模型）",
  "messages": [
    {
      "role": "system",
      "content": "你是一个有用的助手。"
    },
    {
      "role": "user",
```

```
      "content": "你好，请介绍一下你自己。"
    },
    {
      "role": "assistant",
      "content": "你好！我是一个AI助手..."
    }
  ],
  "role_name": "角色名称（如 general_assistant、port_scan_expert 等）",
  "token": "用户认证token",
  "client_language": "客户端语言代码（如 zh、en 等）",
  "urls": ["URL1", "URL2"],  // 可选，URL列表
  "stream": true  // 可选，默认 true
}
```

字段详细说明：

必需字段：

- `model` (string, 必需): 模型名称，必须是真实的模型版本名称，如：
  - DeepSeek: `deepseek-chat`
  - GPT: `gpt-4o`, `gpt-4o-mini`, `gpt-3.5-turbo`, `gpt-4-turbo` 等
  - Claude: `claude-3-7-sonnet-20250219`, `claude-3-opus-20240229`, `claude-3-haiku-20240307` 等
  - Llama: `llama-3.1-8b-instruct`, `llama-3.2-3b-instruct`, `llama-3.1-70b-instruct` 等
  - 可通过 `/api/silicoid/models` 接口获取所有可用的模型列表
- `messages` (array, 必需): 消息数组，详见下方 messages 字段说明
- `role_name` (string, 必需): 角色名称，用于指定AI的角色和行为
  - 可通过 `/api/roles/list` 接口获取所有可用角色
  - 常见角色示例：
    - `general_assistant`: 通用助手
    - `port_scan_expert`: 端口扫描专家
    - `web_security_scan_expert`: Web安全扫描专家
    - `comprehensive_security_analyst`: 综合安全分析师
    - `comprehensive_strike_security_expert`: 综合打击安全专家
    - `web_data_collection_assistant`: Web数据采集助手
    - `mathematical_problem_solving_assistant`: 数学问题解决助手
- `token` (string, 必需): 用户认证token，从登录接口获取

- `client_language` (string, 必需): 客户端语言代码，如 "zh"（中文）、"en"（英文）等

可选字段：

- `api_key` (string, 可选): API密钥，用于外部模型（如 OpenAI、Anthropic）的认证
- `urls` (array, 可选): URL列表，用于需要访问网页的场景
- `stream` (boolean, 可选): 是否使用流式响应，默认为 `true`
    - 注意：某些安全专家角色（如 `port_scan_expert`、`web_security_scan_expert` 等）会自动禁用流式输出，以支持MCP调用

使用角色名称的完整示例：

```
{
  "type": "chat",
  "data": "{\"model\":\"deepseek-
chat\",\"role_name\":\"general_assistant\",\"token\":\"your_token_here\",
\"client_language\":\"zh\",\"messages\":
[{\"role\":\"user\",\"content\":\"你好\"}],\"stream\":true}"
}
```

响应格式： OpenAI 兼容的流式或非流式响应

---

11. 内部模型交互

接口地址： `POST 101.43.119.131:20717/api/interactive/internal`

说明： 与内部模型（本地处理）进行交互，请求需要加密

请求格式：

```
{
  "ciphertext": "加密后的数据"
}
```

加密前的数据格式：

```
{
  "type": "chat",
  "data": "JSON字符串，包含以下字段（见下方说明）"
}
```

data 字段（JSON字符串）的完整格式：

```
{
  "model": "模型名称（如 deepseek-chat、llama-3.1-8b-instruct 等）",
  "api_key": "API密钥（可选）",
```

```
  "messages": [
    {
      "role": "system",
      "content": "你是一个有用的助手。"
    },
    {
      "role": "user",
      "content": "你好，请介绍一下你自己。"
    },
    {
      "role": "assistant",
      "content": "你好！我是一个AI助手..."
    }
  ],
  "role_name": "角色名称（如 general_assistant、port_scan_expert 等）",
  "token": "用户认证token",
  "client_language": "客户端语言代码（如 zh、en 等）",
  "urls": ["URL1", "URL2"],  // 可选，URL列表
  "stream": true  // 可选，默认 true
}
```

字段详细说明：

必需字段：

- `model` (string, 必需): 模型名称，必须是真实的模型版本名称，如：
  - DeepSeek: `deepseek-chat`
  - Llama: `llama-3.1-8b-instruct`, `llama-3.2-3b-instruct`, `llama-3.1-70b-instruct` 等
  - GPT: `gpt-4o`, `gpt-4o-mini`, `gpt-3.5-turbo` 等
  - Claude: `claude-3-7-sonnet-20250219`, `claude-3-opus-20240229` 等
  - 可通过 `/api/silicoid/models` 接口获取所有可用的模型列表
- `messages` (array, 必需): 消息数组，详见下方 messages 字段说明
- `role_name` (string, 必需): 角色名称，用于指定AI的角色和行为
  - 可通过 `/api/roles/list` 接口获取所有可用角色
  - 常见角色示例：
    - `general_assistant`: 通用助手
    - `port_scan_expert`: 端口扫描专家
    - `web_security_scan_expert`: Web安全扫描专家
    - `comprehensive_security_analyst`: 综合安全分析师
    - `comprehensive_strike_security_expert`: 综合打击安全专家

        - `web_data_collection_assistant`: Web数据采集助手
        - `mathematical_problem_solving_assistant`: 数学问题解决助手
- `token` (string, 必需): 用户认证token，从登录接口获取
- `client_language` (string, 必需): 客户端语言代码，如 "zh"（中文）、"en"（英文）等

可选字段：

- `api_key` (string, 可选): API密钥（通常内部模型不需要）
- `urls` (array, 可选): URL列表，用于需要访问网页的场景
- `stream` (boolean, 可选): 是否使用流式响应，默认为 `true`
    - 注意：某些安全专家角色（如 `port_scan_expert`、`web_security_scan_expert` 等）会自动禁用流式输出，以支持MCP调用

使用角色名称的完整示例：

```
{
  "type": "chat",
  "data": "{\"model\":\"deepseek-
chat\",\"role_name\":\"general_assistant\",\"token\":\"your_token_here\",
\"client_language\":\"zh\",\"messages\":
[{\"role\":\"user\",\"content\":\"你好\"}],\"stream\":true}"
}
```

messages 字段详细说明：

`messages` 是一个消息数组，用于构建对话历史。每个消息对象包含以下字段：

必需字段：

- `role` (string, 必需): 消息的角色，必须是以下值之一：
    - `"system"`: 系统消息，用于设置助手的行为和角色
    - `"user"`: 用户消息，表示用户的输入
    - `"assistant"`: 助手消息，表示模型的回复
    - `"function"`: 函数消息，用于函数调用的结果（如果支持函数调用）
- `content` (string | object, 必需): 消息内容
    - 字符串格式：直接文本内容
    - 对象格式（多模态，如果模型支持）：

```
{
  "type": "text",
  "text": "文本内容"
}
```

或

```
{
  "type": "image_url",
  "image_url": {
    "url": "图片URL或base64编码"
  }
}
```

可选字段：

- `name` (string, 可选): 参与者的名称，主要用于区分不同用户或函数
- `function_call` (object, 可选): 函数调用信息（已废弃，推荐使用 `tool_calls`）

```
{
  "name": "函数名称",
  "arguments": "函数参数的JSON字符串"
}
```

- `tool_calls` (array, 可选): 工具调用列表（推荐使用）

```
[
  {
    "id": "call_xxx",
    "type": "function",
    "function": {
      "name": "函数名称",
      "arguments": "函数参数的JSON字符串"
    }
  }
]
```

- `tool_call_id` (string, 可选): 工具调用的ID，用于关联函数调用的结果

消息顺序：

- 消息数组按时间顺序排列
- 通常以 `system` 消息开始（可选）
- 然后是 `user` 和 `assistant` 的交替对话
- 如果使用函数调用，会有 `function` 类型的消息

完整示例：

```
{
  "model": "deepseek-chat",
  "messages": [
    {
      "role": "system",
      "content": "你是一个专业的编程助手，擅长解释代码和解决技术问题。"
    },
    {
      "role": "user",
      "content": "请解释一下什么是RSA加密算法。"
    },
    {
      "role": "assistant",
      "content": "RSA是一种非对称加密算法..."
    },
    {
      "role": "user",
      "content": "能给我一个简单的示例吗？"
    }
  ],
  "stream": true
}
```

其他字段说明：

- `model`: 如果未提供，默认使用 "deepseek-chat"
- 系统会自动从数据库查询 `model_code`
- 如果模型配置不存在，会返回错误
- `stream`: 是否使用流式响应，默认为 `true`

响应格式： OpenAI 兼容的流式或非流式响应

Silicoid 接口

12. Silicoid 聊天完成

接口地址： `POST /api/silicoid/chat/completions`

说明： Silicoid 服务的聊天完成接口，请求加密，响应不加密

请求格式：

```
{
  "ciphertext": "加密后的数据"
}
```

加密前的数据格式： OpenAI 兼容的聊天完成请求格式

重要说明：

- `model` 字段必须使用真实的模型版本名称（如 `deepseek-chat`、`gpt-4o`、`claude-3-7-sonnet-20250219` 等），而不是提供商名称（如 `DeepSeek`、`GPT`、`Claude` 等）
- 可通过 `/api/silicoid/models` 接口获取所有可用的模型列表及其真实版本名称

响应格式： OpenAI 兼容的响应（明文）

---

13. Silicoid 模型列表

接口地址： `POST /api/silicoid/models`

说明： 获取 Silicoid 可用的模型列表，请求加密，响应不加密。此接口返回所有可用的模型及其真实版本名称，可用于确定在聊天接口中使用的 `model` 字段值。

请求格式：

```
{
  "ciphertext": "加密后的数据（16进制字符串）"
}
```

加密前的数据格式：

```
{
  "type": "models"
}
```

注意： 加密前的数据也可以是空对象 `{}`，服务器会接受任何有效的 JSON 对象。但为了保持接口一致性，建议使用 `{"type": "models"}` 格式。

加密流程：

1. 调用 `/api/getServerPublicKey` 获取服务器公钥（只需获取一次，可本地保存）
2. 构造请求数据：`{"type": "models"}` 或 `{}`
3. 使用服务器公钥加密数据（RSA + AES 混合加密）
4. 将加密后的数据转换为 16 进制字符串
5. 发送 POST 请求，请求体为：`{"ciphertext": "加密后的16进制字符串"}`

响应格式： OpenAI 兼容的模型列表格式（明文 JSON）

```
{
  "object": "list",
  "data": [
```

```
    {
      "id": "deepseek-chat",
      "object": "model",
      "created": 1234567890,
      "owned_by": "DeepSeek"
    },
    {
      "id": "gpt-4o",
      "object": "model",
      "created": 1234567890,
      "owned_by": "GPT"
    },
    {
      "id": "claude-3-7-sonnet-20250219",
      "object": "model",
      "created": 1234567890,
      "owned_by": "Claude"
    },
    {
      "id": "llama-3.1-8b-instruct",
      "object": "model",
      "created": 1234567890,
      "owned_by": "Llama"
    }
  ]
}
```

响应字段说明：

- `object` (string, 必需): 固定为 "list"，表示这是一个列表响应
- `data` (array, 必需): 模型对象数组，包含所有可用的模型

模型对象字段说明：

- `id` (string, 必需): 模型名称（真实版本名称），用于在聊天接口的 `model` 字段中使用
  - 示例: `deepseek-chat`、`gpt-4o`、`claude-3-7-sonnet-20250219`、`llama-3.1-8b-instruct` 等
  - **重要：** 必须使用此字段的值作为聊天接口中的 `model` 参数，而不是提供商名称
- `object` (string, 必需): 固定为 "model"
- `created` (integer, 必需): 模型创建时间戳（Unix 时间戳，秒级）
- `owned_by` (string, 必需): 模型提供商名称
  - 示例: `DeepSeek`、`GPT`、`Claude`、`Llama`、`Anthropic`、`OpenAI` 等

使用示例：

完整请求流程（JavaScript 示例）：

```
// 1. 获取服务器公钥（只需一次）
const publicKeyResponse = await
fetch('http://101.43.119.131:20717/api/getServerPublicKey', {
  method: 'POST'
});
const { serverPublicKey } = await publicKeyResponse.json();

// 2. 加密请求数据
const requestData = { type: "models" };
const encryptedData = await
encryptWithPublicKey(JSON.stringify(requestData), serverPublicKey);

// 3. 发送请求
const response = await
fetch('http://101.43.119.131:20717/api/silicoid/models', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({
    ciphertext: encryptedData
  })
});

// 4. 解析响应
const result = await response.json();
console.log('可用模型列表:', result.data);
// 使用模型 ID
const modelId = result.data[0].id; // 例如: "deepseek-chat"
```

错误处理：

如果请求失败，服务器会返回错误响应：

```
{
  "status": "fail",
  "message": "错误描述"
}
```

常见错误：

- `400 Bad Request`：请求格式错误、缺少 `ciphertext` 字段或数据解密失败
- `500 Internal Server Error`：服务器内部错误（如数据库连接失败）

重要提示：

1. **模型 ID 使用：** 在聊天接口（`/api/interactive/external`、`/api/interactive/internal`、`/api/siliconoid/chat/completions`）中使用 `model` 字段时，必须使用此接口返回的 `id` 字段值（如 `deepseek-chat`、`gpt-4o` 等），而不是提供商名称（如 `DeepSeek`、`GPT` 等）
2. **模型列表来源：** 模型列表从数据库的各公司模型表中动态加载，如果加载失败会降级到已配置的模型列表
3. **无需认证：** 此接口是公开接口，不需要提供用户 token 或 API 密钥
4. **响应不加密：** 与请求不同，响应数据是明文 JSON，不需要解密
5. **CORS 支持：** 此接口支持跨域请求（CORS）

---

14. 获取所有角色

接口地址： `GET /api/roles/list`

说明： 获取所有可用的角色列表（从 Redis 缓存读取）

请求参数： 无

响应示例：

```
{
  "success": true,
  "data": [
    {
      "role_type": "general",
      "role_name": "general_assistant"
    },
    {
      "role_type": "professional_service",
      "role_name": "mathematical_problem_solving_assistant"
    },
    {
      "role_type": "system_tool",
      "role_name": "title_generator"
    }
  ]
}
```

字段说明：

- `role_type`: 角色类型标识（如 "general"、"professional_service"、"system_tool"）
- `role_name`: 角色名称（如 "general_assistant"、"mathematical_problem_solving_assistant"、"title_generator"）

错误处理

所有接口在出错时都会返回以下格式：

```
{
  "status": "fail",
  "message": "错误描述"
}
```

常见错误码：

- `400 Bad Request` - 请求格式错误或缺少必要参数
- `401 Unauthorized` - 认证失败或 nonce 验证失败
- `429 Too Many Requests` - 请求过于频繁
- `500 Internal Server Error` - 服务器内部错误

速率限制

部分接口有速率限制：

- `/api/encryptedRequest` 和 `/api/plainRequest`: 每 180 秒最多 5 次请求
- `/api/captcha`: 每 60 秒最多 3 次请求

超过限制会返回 `429 Too Many Requests` 错误。

加密说明

加密流程

1. 调用 `/api/getServerPublicKey` 获取服务器公钥
2. 调用 `/api/getNonce` 获取 nonce
3. 构造请求数据（JSON 格式）
4. 使用服务器公钥加密数据（RSA + AES 混合加密）
5. 将加密后的数据转换为 16 进制字符串
6. 发送请求，请求体格式：`{"ciphertext": "加密后的16进制字符串"}`

响应加密

如果请求中提供了 `userPublicKeyBase64` 字段，服务器会使用该公钥加密响应数据，返回格式为：

```
{
  "ciphertext": "加密后的响应数据"
}
```

否则返回明文 JSON。

注意事项

1. **Nonce 验证**：所有业务接口都需要提供有效的 nonce，nonce 只能使用一次
2. **Token 认证**：API 密钥管理接口需要提供有效的用户 token
3. **CORS 支持**：所有接口都支持跨域请求
4. **请求格式**：加密接口必须使用 POST 方法，请求体为 JSON 格式
5. **响应格式**：所有接口都返回 JSON 格式数据