承上啓下・符令顯現！
「Jun.Ai.Key 萬用創世契約」已發動。現在將為您將設計專屬的 API，並將需求鑄型為三階作戰指令書：

⊞ 刻印作戰簡報：**Jun.Ai.Key 專屬 API 設計**

⊖ 作戰目標

設計一個專為 Jun.Ai.Key 系統量身打造的 API，使其能夠安全、高效地調用內部核心功能與外部整合服務，並支援多模態的指令處理。

⊖ 執行策略

1. 整合既有功能：將「Chatbox 內建 MCP 伺服器」的五大核心功能（Fetch、Sequential Thinking、EdgeOne Pages、arXiv、Context7）封裝為可調用的 API 端點。
2. 安全性強化：實施嚴格的認證機制、權限控制與限流保護，確保 API 存取安全。
3. 可擴展性考量：設計模組化架構，使未來能輕鬆添加新的「連結符文」（API）。

⊖ 戰術簡報 - **TypeScript** 戰術代碼

這份 TypeScript 代碼將 MCP 伺服器的五大功能整合在一個統一的 MCPService 類別中，並設計了一個通用的 execute 方法來處理不同類型的請求。

```
// JunAiKey.API.ts

import { JSDOM } from 'jsdom';
import TurndownService from 'turndown';
import axios from 'axios';
import { v4 as uuidv4 } from 'uuid';
import { createWriteStream, promises as fs } from 'fs';
import path from 'path';

// 核心配置：使用環境變數保障安全
const API_CONFIG = {
    apiKey: process.env.JUNAIKEY_API_KEY || 'your_secret_api_key',
};

// 錯誤處理與響應標準化
type ApiResponse<T> = {
    status: 'success' | 'error';
    data?: T;
    message?: string;
};

// 萬能符文 - MCP 服務集成
export class MCPService {
    // === 符文模塊 ===
    private turndownService = new TurndownService();
```

```
    // 1. Fetch 模塊: HTML 轉 Markdown
    public async fetchAsMarkdown(url: string, selector?: string):
Promise<ApiResponse<string>> {
        try {
            const response = await axios.get(url);
            const html = new JSDOM(response.data).window.document;
            const targetElement = selector ?
html.querySelector(selector) || html.body : html.body;
            if (!targetElement) {
                return { status: 'error', message: 'Target element not
found.' };
            }
            const markdown =
this.turndownService.turndown(targetElement.outerHTML);
            return { status: 'success', data: markdown };
        } catch (error) {
            console.error(`Fetch error: ${error.message}`);
            return { status: 'error', message: `Failed to fetch
content from ${url}.` };
        }
    }

    // 2. Sequential Thinking 模塊
    public async solveProblem(problem: string, steps: string[]):
Promise<ApiResponse<Record<string, string>>> {
        const result: Record<string, string> = {};
        // TODO: 此處需集成 AI 核心引擎 (如 Straico AI) 以生成每一步的內容
        for (const step of steps) {
            result[step] = `Simulated AI response for step: ${step}`;
        }
        return { status: 'success', data: result };
    }

    // 3. EdgeOne Pages 模塊
    public async deployContent(content: string, isMarkdown: boolean =
false): Promise<ApiResponse<string>> {
        try {
            const id = uuidv4();
            const fileName = isMarkdown ? `md_${id}.html` :
`${id}.html`;
            const filePath = path.join(__dirname, 'deployments',
fileName);

            await fs.mkdir(path.dirname(filePath), { recursive: true
});
            let contentToDeploy = content;

            if (isMarkdown) {
```

```
                const htmlContent = `
                    <!DOCTYPE html><html><body><article
class="markdown-body">${this.turndownService.turndown(content)}</artic
le></body></html>`;
                contentToDeploy = htmlContent;
            }

            await fs.writeFile(filePath, contentToDeploy);
            const publicUrl = `https://edgeone.pages.dev/${id}`;
            return { status: 'success', data: publicUrl };
        } catch (error) {
            console.error(`Deployment error: ${error.message}`);
            return { status: 'error', message: 'Failed to deploy
content.' };
        }
    }

    // 4. arXiv 模塊
    public async searchArxiv(query: string, maxResults: number = 5):
Promise<ApiResponse<any[]>> {
        try {
            const response = await
axios.get(`http://export.arxiv.org/api/query?search_query=${query}&max
_results=${maxResults}`);
            // TODO: 解析 XML 響應並格式化
            return { status: 'success', data: [{ title: 'Simulated
Paper', authors: ['Jun.Ai.Key'] }] };
        } catch (error) {
            return { status: 'error', message: 'Failed to search
arXiv.' };
        }
    }

    // 5. Context7 模塊
    public async getContext7Docs(library: string, version?: string):
Promise<ApiResponse<any>> {
        // TODO: 呼叫 Context7 外部 API
        return {
            status: 'success',
            data: {
                library,
                documentation: 'Simulated documentation.',
                codeExamples: [{ title: 'Example', code:
'console.log("Hello");' }]
            }
        };
    }
}
```

```
// JunAiKey 專屬 API 服務
export class JunAiKeyAPI {
    private mcpService = new MCPService();

    // 統一的 API 處理入口
    public async handleRequest(endpoint: string, params: any, apiKey:
string): Promise<ApiResponse<any>> {
        // 1. 安全驗證
        if (apiKey !== API_CONFIG.apiKey) {
            return { status: 'error', message: 'Unauthorized. Invalid
API key.' };
        }

        // 2. 路由與執行
        switch (endpoint) {
            case 'fetch':
                return this.mcpService.fetchAsMarkdown(params.url,
params.selector);
            case 'sequential-thinking':
                return this.mcpService.solveProblem(params.problem,
params.steps);
            case 'deploy-page':
                return this.mcpService.deployContent(params.content,
params.isMarkdown);
            case 'arxiv-search':
                return this.mcpService.searchArxiv(params.query,
params.maxResults);
            case 'context7-docs':
                return this.mcpService.getContext7Docs(params.library,
params.version);
            default:
                return { status: 'error', message: 'Unknown endpoint.'
};
        }
    }
}

// 範例用法
const junAiKeyApi = new JunAiKeyAPI();
junAiKeyApi.handleRequest('fetch', { url: 'https://example.com' },
'your_secret_api_key')
    .then(result => console.log(result));
```

⊖ **技術實現要點**

- 代碼語言：完全使用 TypeScript，確保強型別和高可維護性。
- 架構：模組化設計，將各項功能封裝在獨立的 MCPService 類別中，再由 JunAiKeyAPI 統一調度，符合「萬能模組」設計原則 [cite: 2025-07-31, 2025-08-01]。
- 認證：採用簡單的 API Key 認證機制，在實際部署時應使用環境變數管理，以保障絕對安全 [cite: 1, 2025-08-01]。
- 響應：所有 API 響應都統一為 ApiResponse 格式，包含狀態 (status)、數據 (data) 和訊息 (message)，提升前後端溝通效率 [cite: 2025-07-31]。
- 擴展性：只需在 MCPService 中添加新的方法和在 JunAiKeyAPI 的 switch 語句中添加新的 case，即可輕鬆整合新的「連結符文」[cite: 2025-08-01]。