# 程序鉴别材料（前30页）

## 文件1: worker.js（前30行）

```
// 缓存存储
const CACHE_NAME = 'sales-proxy-cache-v2';
const CACHE_TTL = 10 * 60 * 1000; // 10分钟缓存

export default {
  async fetch(request, env) {
    // 检查是否为API请求
    const url = new URL(request.url);

    // 对于GET请求且不是API请求，尝试使用缓存
    if (request.method === 'GET' && !url.pathname.startsWith('/api/'))
{
      const cachedResponse = await getFromCache(request);
      if (cachedResponse) {
        return cachedResponse;
      }
    }

    // 处理运营策略模块的单独路由（需要在通用API路由之前处理）
    if (url.pathname === '/api/traffic-operation' && request.method
=== 'POST') {
      return await handleTrafficOperationRequest(request, env);
    }

    // 处理风控与售后模块的单独路由
    if (url.pathname === '/api/risk-control-and-after-sales' &&
request.method === 'POST') {
      return await handleRiskControlRequest(request, env);
    }

    // 处理数据化运营指标模块的单独路由
    if (url.pathname === '/api/data-operation-dashboard' &&
request.method === 'POST') {
      return await handleDataOperationRequest(request, env);
    }

    // 处理代销合作模式模块的单独路由
    if (url.pathname === '/api/cooperation-model' && request.method
=== 'POST') {
      return await handleCooperationModelRequest(request, env);
    }

    // 处理 API 模块路由
    if (url.pathname.startsWith('/api/')) {
      const moduleName = url.pathname.substring(5); // 移除 '/api/' 前
缀

      // 对于 GET 请求，直接处理特定模块
      if (request.method === 'GET') {
        switch(moduleName) {
          case 'drug-selection-strategy':
            return handleSelectionStrategyRequest(request, env);
          default:
            return new Response(JSON.stringify({ error: 'Invalid
module name or method not allowed', module: moduleName }), {
              status: 400,
              headers: { 'Content-Type': 'application/json' }
            });
        }
      }

      // 对于 POST 请求，重构请求体以匹配模块名称
      if (request.method === 'POST') {
        let body = {};
        try {
          body = await request.json();
        } catch (e) {
          // 如果不是 JSON 格式，使用空对象
        }
        const newBody = { ...body, module_name: moduleName };

        // 创建新的请求对象，但是使用原始请求的其他属性
        const newRequest = new Request(request.url, {
          method: 'POST',
          headers: request.headers,
          body: JSON.stringify(newBody)
        });

        return await handleRequest(newRequest, env);
      }
    }

    // 处理 POST 请求（非 API）
    if (request.method === 'POST') {
      return await handleRequest(request, env);
    }

    // 处理静态页面请求 (GET 请求)
    try {
      const response = await env.ASSETS.fetch(request);
      // 添加缓存控制头以避免 304 问题
      const newHeaders = new Headers(response.headers);
      newHeaders.set('Cache-Control', 'public, max-age=600, stale-
while-revalidate=300'); // 10 分钟缓存，5 分钟宽限
      newHeaders.set('Expires', new Date(Date.now() +
600000).toUTCString());

      // 如果是 GET 请求，缓存响应
      if (request.method === 'GET') {
        await putInCache(request, response);
      }

      return new Response(response.body, {
        status: response.status,
        statusText: response.statusText,
        headers: newHeaders
      });
    } catch (e) {
      console.error('Static asset fetch failed:', e);
      return new Response('Static asset fetch failed: ' + e.message,
{ status: 500 });
    }
  }
}

// 模拟区块链存证函数
async function saveToBlockchain(terms) {
  // 这里应该调用实际的区块链存证 API
  // 模拟返回存证哈希值
  return {
    term_id: `term_${Date.now()}`,
    contract_status: "draft",
    signed_urls: {
      brand_signature: "https://sign.url/brand",
      affiliate_signature: "https://sign.url/affiliate"
    },
    blockchain_proof: "0x123abc..."
  };
}

// 缓存函数
async function getFromCache(request) {
  try {
    const cache = await caches.open(CACHE_NAME);
    const cachedResponse = await cache.match(request);

    if (cachedResponse) {
      // 检查缓存是否过期
      const cacheTime = cachedResponse.headers.get('x-cache-time');
      if (cacheTime && (Date.now() - parseInt(cacheTime)) < CACHE_TTL)
{
        // 更新缓存的最后访问时间（用于 LRU 策略）
        const headers = new Headers(cachedResponse.headers);
        headers.set('x-cache-last-access', Date.now().toString());
        return new Response(cachedResponse.body, {
          status: cachedResponse.status,
          statusText: cachedResponse.statusText,
          headers: headers
        });
      } else {
        // 缓存过期，删除
        await cache.delete(request);
      }
    }
  } catch (e) {
    console.error('Cache read error:', e);
  }

  return null;
}

async function putInCache(request, response) {
  try {
    // 只缓存成功的 GET 请求
    if (request.method === 'GET' && response.status === 200) {
      const cache = await caches.open(CACHE_NAME);

      // 克隆响应并添加缓存时间戳
      const clonedResponse = response.clone();
      const headers = new Headers(clonedResponse.headers);
      headers.set('x-cache-time', Date.now().toString());
      headers.set('x-cache-last-access', Date.now().toString());

      const responseToCache = new Response(clonedResponse.body, {
        status: clonedResponse.status,
        statusText: clonedResponse.statusText,
        headers: headers
      });

      await cache.put(request, responseToCache);
    }
  } catch (e) {
    console.error('Cache write error:', e);
  }
}
```

```
// 模拟权限校验函数
async function verifyParty(term_id, party) {
  // 这里应该调用实际的权限校验 API
  // 模拟返回校验结果
  return true;
}

// 模拟仲裁服务调用函数
async function triggerArbitration(details) {
  // 这里应该调用实际的仲裁服务 API
  // 模拟返回仲裁 ID
  return `arbitration_${Date.now()}`;
}

// 模拟数据查询函数
async function queryData(term_id, data_type, time_range) {
  // 这里应该调用实际的数据查询 API
  // 模拟返回数据
  if (data_type === "sales") {
    return {
      total: 150000,
      avg_daily: 5000,
      top_region: "广东"
    };
  }
  return {};
}

// 药品代销风控与售后综合 API 处理函数
async function handleRiskControlRequest(request, env) {
  // 检查请求方法
  if (request.method !== 'POST') {
    return new Response(JSON.stringify({
      error: 'Method not allowed',
      allowed_methods: ['POST'],
      received_method: request.method
    }), {
      status: 405,
      headers: { 'Content-Type': 'application/json' }
    });
  }

  // 检查内容类型
  const contentType = request.headers.get('content-type');
  if (!contentType || !contentType.includes('application/json')) {
    return new Response(JSON.stringify({
      error: 'Unsupported content type',
      expected: 'application/json',
      received: contentType
    }), {
      status: 406,
      headers: { 'Content-Type': 'application/json' }
    });
  }

  let body;
  try {
    body = await request.json();
  } catch (error) {
    return new Response(JSON.stringify({
      error: 'Invalid JSON body',
      details: error.message
    }), {
      status: 400,
      headers: { 'Content-Type': 'application/json' }
    });
  }

  const module = body.module_name;

  // 添加调试日志
  // console.log('Received module name:', module);
  // console.log('Request body:', JSON.stringify(body));

  switch(module) {
    case 'tencent_cloud_medical_content_audit':
      return handleContentAudit(body, env);
    case 'compliant_pharmacist_transfer':
      return handlePharmacistTransfer(body, env);
    case 'drug_registration_wechat_notification':
      return handleWechatNotification(body, env);
    case 'return_refund_auto_review':
      return handleReturnReview(body, env);
    default:
      return new Response(JSON.stringify({ error: 'Invalid module
name', module: module }), {
        status: 400,
        headers: { 'Content-Type': 'application/json' }
      });
  }
}

// 药品代销短视频流量运营综合 API 处理函数
async function handleTrafficOperationRequest(request, env) {
  // 检查请求方法
  if (request.method !== 'POST') {
    return new Response(JSON.stringify({
      error: 'Method not allowed',
      allowed_methods: ['POST'],
      received_method: request.method
    }), {
      status: 405,
      headers: { 'Content-Type': 'application/json' }
    });
  }

  // 检查内容类型
  const contentType = request.headers.get('content-type');
  if (!contentType || !contentType.includes('application/json')) {
    return new Response(JSON.stringify({
      error: 'Unsupported content type',
      expected: 'application/json',
      received: contentType
    }), {
      status: 406,
      headers: { 'Content-Type': 'application/json' }
    });
  }

  let body;
  try {
    body = await request.json();
  } catch (error) {
    return new Response(JSON.stringify({
      error: 'Invalid JSON body',
      details: error.message
    }), {
      status: 400,
      headers: { 'Content-Type': 'application/json' }
    });
  }

  const module = body.module_name;

  // 添加调试日志
  // console.log('Received module name:', module);
  // console.log('Request body:', JSON.stringify(body));

  switch(module) {
    case 'chanmama_ad_placement':
      return handleChanmamaAdPlacement(body, env);
    case 'jietiao_smart_clipping':
      return handleJietiaoSmartClipping(body, env);
    case 'influencer_recommendation':
      return handleInfluencerRecommendation(body, env);
    case 'publish_schedule_webhook':
      return handlePublishScheduleWebhook(body, env);
    default:
      return new Response(JSON.stringify({ error: 'Invalid module
name', module: module }), {
        status: 400,
        headers: { 'Content-Type': 'application/json' }
    });
  }
}
```

# 程序鉴别材料（后 30 页）

## 文件 1: worker.js (后 30 行)

```
  };

  return templates[type] || templates["保健品"];
}
// 辅助函数（示例）
function getSuggestions(type) {
  switch(type) {
    case "保健品":
      return [
        "增加临床数据引用",
        "替换更具体的痛点"
      ];
    case "OTC 药品":
      return [
        "强化医生/药师背书",
        "添加用药禁忌提示"
      ];
    case "中药饮片":
      return [
        "增加传统医学理论支撑",
        "添加适宜人群说明"
      ];
    default:
      return [
        "可增加用户评价",
        "可添加使用场景"
      ];
  }
}
```