AI绘本创作系统

企业后端集成指南 V3.1

V3.1 | 2026-04-03 | 新增 Session Token 认证 + dataVersion 同步机制

文档版本	V3.1
发布日期	2026-04-03
适用场景	企业只有管理后台无C端，乐读派提供C端H5+Android
核心能力	Webhook回调同步 + STS直传 + A6角色提取 + A3图片创作 + A20配音
同步机制	Webhook主通道 + B3批量拉取兜底 + B2单品补偿 + dataVersion门卫
认证方式	Session Token（H5生产推荐）+ HMAC-SHA256（开发/Android）

目录

一、方案概述

1.1 适用场景

本方案面向只有管理后台、没有自建C端（H5/App）的企业。企业的核心需求是AI绘本创作能力，但不希望投入C端开发资源。

特征	说明
企业现状	有管理后台（CMS/OA/LMS），无面向终端用户的H5或App
用户触达	由乐读派提供C端H5页面和Android客户端，企业嵌入或跳转
数据回流	用户在乐读派C端完成创作后，作品数据通过Webhook实时推送到企业后端
企业后端角色	接收Webhook回调、存储作品数据、管理用户-作品关系、提供管理后台展示

1.2 核心流程

整个集成围绕三个阶段展开：

核心优势：企业零C端开发成本。乐读派提供完整的C端体验（H5+Android），企业只需实现一个Webhook接收接口，即可获得所有创作数据。

二、架构与职责划分

2.1 职责对照表

职责领域	乐读派	企业
C端界面	提供H5页面+Android客户端 企业嵌入或跳转使用	无需开发C端 仅需提供入口链接
用户创作	完整创作流程：画作上传、 角色提取、故事生成、配音	无需介入 用户在乐读派C端操作
实时进度	WebSocket推送给C端用户 创作过程中的进度反馈	无需关注 由乐读派C端处理
作品回传	Webhook主动推送到企业 包含完整作品数据	实现Webhook接收接口 验签+持久化
数据存储	OSS存储AI生成的图片/音频 链接长期有效	本地DB存储作品元数据 关联用户和业务
管理后台	提供机构管理、额度管理、 画风配置等平台级管理	作品审核、用户管理、 统计分析等业务管理
额度管理	扣减创作额度 按机构/周期统计	监控剩余额度 按需充值

2.2 数据流向示意

数据从用户创作到企业入库的完整流向：

步骤	动作	数据方向
1	用户通过企业入口进入乐读派C端	企业 -> 乐读派C端
2	用户完成创作（A6+A3+A20）	乐读派C端 <-> 乐读派API
3	创作完成，触发Webhook	乐读派API -> 企业后端
4	企业后端接收并持久化	企业后端 -> 企业DB
5	企业管理后台展示	企业DB -> 企业管理后台

重要：企业后端不需要代理调用A6/A3/A20等创作接口。这些接口由乐读派C端直接调用。企业后端需要实现的接口：1) 令牌交换调用（1次API调用）； 2) Webhook接收。

三、C端H5接入认证（会话令牌）

乐读派C端H5的API认证支持两种模式。企业生产环境必须使用 Session Token 模式，确保 appSecret 不会暴露在浏览器端。

3.1 认证双模式说明

模式1: HMAC签名（开发调试/直连模式）

H5直接用orgId + appSecret签名
appSecret存在浏览器localStorage（仅限开发）
适合本地调试和快速验证，不可用于生产

模式2: Session Token（企业生产环境，推荐）

企业后端调 POST /api/v1/auth/session 换令牌
appSecret只在企业服务端，不到达浏览器
H5用 Authorization: Bearer sess_xxx 调API
令牌有效期2小时，过期重新换取

3.2 安全对比

对比维度	HMAC模式	Session Token模式
appSecret位置	浏览器localStorage	仅企业服务端
安全性	低（可被F12查看）	高（不到达浏览器）
适用场景	开发调试	生产环境
令牌有效期	无过期	2小时
暴力破解防护	无	5次错误/10分钟锁定
密钥泄露风险	高（前端可见）	极低（服务端保管）

重要：HMAC模式下appSecret存储在浏览器端，任何用户都可以通过F12开发者工具查看。这意味着您的密钥将完全暴露。生产环境必须使用Session Token模式！

3.3 Session Token 流程图

Session Token 认证涉及三方交互：企业后端、乐读派后端、用户浏览器。

Step 1: 企业后端换取令牌

```
企业后端 --> POST /api/v1/auth/session --> 乐读派后端
请求: { "orgId": "ORG001", "appSecret": "企业密钥", "phone": "13800001111" }

乐读派后端 --> 响应 --> 企业后端
响应: { "code": 200, "data": {
```

```
    "sessionToken": "sess_a3f8c912e4b7d1056c89a2e4f7b3d1a2",
    "expiresIn": 7200
} }
```

Step 2: 企业后端重定向用户到H5

```
企业后端 --> 302重定向 --> 用户浏览器
Location: https://h5.leai.com/?token=sess_xxx&orgId;=ORG001□=13800001111

注意：token通过URL参数传递，H5页面接收后存入内存（非localStorage）
```

Step 3: H5使用令牌调用API

```
用户浏览器(H5) --> Authorization: Bearer sess_xxx --> 乐读派后端
（所有后续API调用都携带此Header，无需HMAC签名）

令牌过期（2小时后）：
乐读派后端 --> { "code": 20009, "message": "会话已过期" } --> H5
H5 --> 跳转回企业入口 --> 重新换取令牌
```

3.4 令牌交换接口详情

项目	内容
方法	POST
路径	/api/v1/auth/session
认证	无需HMAC签名，直接传appSecret
安全	5次密钥错误/10分钟锁定（暴力破解防护）

请求体：

```
{ "orgId": "ORG001", "appSecret": "企业密钥", "phone": "13800001111" }
```

响应（成功）：

```
{ "code": 200, "data": {
    "sessionToken": "sess_a3f8c912e4b7d1056c89a2e4f7b3d1a2",
    "expiresIn": 7200
} }
```

获取token后，重定向用户到乐读派H5：

```
https://h5.leai.com/?token=sess_xxx&orgId;=ORG001□=13800001111
```

令牌有效期2小时，过期后需重新换取。企业建议在用户每次点击"开始创作"时换取新令牌。

3.5 Session Token 错误码

错误码	常量名	说明	处理建议
20009	SESSION_EXPIRED	会话已过期	重新调用令牌交换接口换取新token

错误码	常量名	说明	处理建议
20010	SESSION_INVALID	会话凭证无效	检查token格式，重新换取
20002	ACCOUNT_LOCKED	密钥错误锁定	5次密钥错误后锁定10分钟， 检查appSecret是否正确

3.6 企业代码示例

Java Spring Boot 示例（令牌交换+重定向）

```
@RestController
@RequestMapping("/creation")
public class CreationEntryController {

  @Value("${leai.api-url}") private String leaiUrl;
  @Value("${leai.h5-url}")  private String h5Url;
  @Value("${leai.org-id}")  private String orgId;
  @Value("${leai.app-secret}") private String appSecret;

  private final RestTemplate restTemplate;

  public CreationEntryController(RestTemplateBuilder builder) {
    this.restTemplate = builder
        .setConnectTimeout(Duration.ofSeconds(5))
        .setReadTimeout(Duration.ofSeconds(10)).build();
  }

  /** 用户点击"开始创作"时调用 */
  @GetMapping("/start")
  public void startCreation(@RequestParam String phone,
      HttpServletResponse response) throws IOException {
    // 1. 换取Session Token
    Map<String, String> body = new HashMap<>();
    body.put("orgId", orgId);
    body.put("appSecret", appSecret);
    body.put("phone", phone);

    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    HttpEntity<Map<String, String>> req =
        new HttpEntity<>(body, headers);

    ResponseEntity<Map> resp = restTemplate.postForEntity(
        leaiUrl + "/api/v1/auth/session", req, Map.class);

    Map data = (Map) resp.getBody().get("data");
    String token = data.get("sessionToken").toString();

    // 2. 302重定向用户到乐读派H5
    String redirectUrl = h5Url + "/?token=" + token
        + "&orgId=" + orgId + "&phone=" + phone;
    response.sendRedirect(redirectUrl);
  }
}
```

Python 示例（Flask）

```
import os, requests
from flask import Flask, redirect, request

app = Flask(__name__)
LEAI_URL = "https://api.leai.com"
H5_URL = "https://h5.leai.com"
ORG_ID = "ORG001"
APP_SECRET = os.environ["LEAI_APP_SECRET"]  # 从环境变量读取

@app.route("/create")
def start_creation():
  phone = request.args["phone"]
  # 1. 换取令牌
  resp = requests.post(f"{LEAI_URL}/api/v1/auth/session", json={
      "orgId": ORG_ID, "appSecret": APP_SECRET, "phone": phone
  })
  token = resp.json()["data"]["sessionToken"]
  # 2. 重定向到H5
  return redirect(f"{H5_URL}/?token={token}&orgId;={ORG_ID}□={phone}")
```

Node.js 示例（Express）

```
const express = require("express");
const axios = require("axios");
const app = express();

const LEAI_URL = "https://api.leai.com";
const H5_URL = "https://h5.leai.com";
const ORG_ID = "ORG001";
const APP_SECRET = process.env.LEAI_APP_SECRET;

app.get("/create", async (req, res) => {
  const { phone } = req.query;
  // 1. 换取令牌
  const { data } = await axios.post(
    `${LEAI_URL}/api/v1/auth/session`,
    { orgId: ORG_ID, appSecret: APP_SECRET, phone }
  );
  const token = data.data.sessionToken;
  // 2. 重定向到H5
  res.redirect(`${H5_URL}/?token=${token}&orgId;=${ORG_ID}□=${phone}`);
});
```

四、Webhook 回调详解

Webhook
是本方案的核心同步机制。乐读派在作品生命周期的关键节点主动POST回调到企业配置的URL。

4.1 事件类型

事件类型	触发时机	频率	说明
work.created	A3创建作品时	1次/作品	作品开始创作，返回workId
work.processing	创作进度变化时	多次/作品	进度更新（10%/30%/70%等）
work.completed	创作完成时	1次/作品	包含完整pageList（核心事件）
work.failed	创作失败时	1次/作品	包含失败原因
work.updated	作品信息变更时	按需	标题/标签等元数据更新
work.audio_updated	配音完成时	按需	新增/更新音频URL
work.deleted	作品被删除时	1次/作品	软删除通知
extract.completed	A6角色提取完成	1次/提取	角色列表数据
quota.warning	额度不足预警	按需	剩余额度低于阈值

4.2 统一 Payload 结构

所有Webhook回调共享统一的JSON结构，通过 event 字段区分事件类型：

HTTP Headers 携带签名信息，请求体为纯 JSON：

```
// HTTP Headers:
// X-Webhook-Id: evt_1903686714382888960
// X-Webhook-Event: work.completed
// X-Webhook-Timestamp: 1711234567890
// X-Webhook-Signature: HMAC-SHA256=a3f8c2d1e5b7...

// 请求体 (JSON):
{
  "id": "evt_1903686714382888960",
  "event": "work.completed",
  "created_at": 1711234567890,
  "data": {
    // 事件相关的业务数据（见各事件详情）
  }
}
```

字段	类型	说明
id	String	事件唯一ID（与Header X-Webhook-Id一致，用于幂等去重）
event	String	事件类型（如 work.completed）

字段	类型	说明
created_at	Long	事件触发时间（毫秒时间戳）
data	Object	事件数据体（不同事件结构不同）

签名在 X-Webhook-Signature Header 中传递，不在 JSON body 中。验签时使用请求body原文，不要先解析再序列化。

4.3 work.completed 事件详情（核心）

这是企业最需要关注的事件。创作完成后推送完整的作品数据，包含封面和所有分镜页。


```
{
  "event": "work.completed",
  "event_id": "evt_1903686714382889001",
  "timestamp": 1711234567890,
  "org_id": "ORG001",
  "data": {
    "work_id": "1903686714382889000",
    "data_version": 3,
    "title": "小璃的森林冒险",
    "status": "COMPLETED",
    "completion_step": 1,
    "tags": ["冒险", "成长", "友谊"],
    "style": "style_cartoon",
    "phone": "13800001111",
    "pages": 6,
    "page_list": [
      {
        "page_num": 0,
        "text": "小璃的森林冒险",
        "image_url": "https://oss.../cover.png",
        "audio_url": null
      },
      {
        "page_num": 1,
        "text": "阳光明媚的早晨，小璃走出家门...",
        "image_url": "https://oss.../page_1.png",
        "audio_url": null
      }
    ],
    "created_at": "2026-04-02T10:30:00Z",
    "completed_at": "2026-04-02T10:32:15Z"
  },
  "signature": "a3f8c2d1e5b7..."
}
```


page_num=0 为封面页，page_num>=1 为故事正文页。image_url 为 PNG 格式（2560x1440），存储在阿里云 OSS，链接长期有效。completion_step=1 表示纯图文完成，=2 表示含配音。data_version 为递增版本号，企业同步时必须做“远程 > 本地”判断（详见第六章）。

4.4 work.updated 事件详情

当作品的元数据（标题、标签等）发生变更时触发。data 中仅包含变更的字段。


```
{
  "event": "work.updated",
  "event_id": "evt_1903686714382889010",
  "timestamp": 1711234600000,
  "org_id": "ORG001",
  "data": {
    "work_id": "1903686714382889000",
    "changed_fields": {
      "title": "小璃的奇妙森林之旅",
      "tags": ["冒险", "成长", "友谊", "森林"]
    }
  },
  "signature": "b4e9d3f2a6c8..."
}
```


changed_fields 仅包含本次修改的字段，未变更字段不会出现。企业后端按字段增量更新本地DB即可。

4.5 work.audio_updated 事件详情

当用户在乐读派C端完成AI配音后触发，推送已配音页面的音频URL。


```
{
  "event": "work.audio_updated",
  "event_id": "evt_1903686714382889020",
  "timestamp": 1711234700000,
  "org_id": "ORG001",
  "data": {
    "work_id": "1903686714382889000",
    "data_version": 5,
    "completion_step": 2,
    "audio_pages": [
      { "page_num": 0, "audio_url": "https://oss.../page_0.mp3" },
      { "page_num": 1, "audio_url": "https://oss.../page_1.mp3" },
      { "page_num": 2, "audio_url": "https://oss.../page_2.mp3" },
      { "page_num": 3, "audio_url": "https://oss.../page_3.mp3" },
      { "page_num": 4, "audio_url": "https://oss.../page_4.mp3" },
      { "page_num": 5, "audio_url": "https://oss.../page_5.mp3" }
    ],
    "total_voiced": 6
  },
  "signature": "c5f0e4g3b7d9..."
}
```


企业后端收到后，按 page_num 匹配更新本地DB中对应页面的 audio_url。completion_step 从 1 更新为 2，表示该作品已含配音。

4.6 签名验证

企业后端必须验证每个Webhook回调的签名，防止伪造请求。签名算法与HMAC认证一致。

签名生成规则

```
签名体 = "{X-Webhook-Id}.{X-Webhook-Timestamp}.{请求body原文}"
签名值 = HMAC-SHA256(签名体, app_secret).toHex()

Header: X-Webhook-Signature: HMAC-SHA256={签名值}

注意：请求body原文是完整的JSON字符串，不做任何重排
```

Java Spring Boot 验签示例（完整 Controller）

```
@RestController
@Slf4j
public class WebhookController {

  @Value("${leai.app-secret}") private String appSecret;
  private final WorkSyncService workSyncService;
  // 幂等缓存（生产建议用Redis: SETNX + TTL 24h）
  private final Set<String> processedEvents =
      Collections.newSetFromMap(
          new ConcurrentHashMap<>());

  @PostMapping("/webhook/leai")
  public ResponseEntity<String> handleWebhook(
      @RequestBody String rawBody,
      @RequestHeader("X-Webhook-Id") String webhookId,
      @RequestHeader("X-Webhook-Timestamp") String ts,
      @RequestHeader("X-Webhook-Signature") String sig) {
    // 1. 时间窗口校验（5分钟）
    if (System.currentTimeMillis() - Long.parseLong(ts)
        > 300_000L) {
      return ResponseEntity.status(401).body("expired");
    }
    // 2. 幂等去重
    if (processedEvents.contains(webhookId)) {
      return ResponseEntity.ok("duplicate");
    }
    // 3. HMAC-SHA256 验签
    String signData = webhookId + "." + ts + "." + rawBody;
    if (!verifyHmac(signData, sig)) {
      return ResponseEntity.status(401).body("bad sig");
    }
    // 4. 解析事件并分发处理
    JSONObject payload = JSON.parseObject(rawBody);
    String event = payload.getString("event");
    JSONObject data = payload.getJSONObject("data");
    workSyncService.onReceiveSync(event, data);
    processedEvents.add(webhookId);
    return ResponseEntity.ok("ok");
  }

  private boolean verifyHmac(String data, String header) {
    try {
      Mac mac = Mac.getInstance("HmacSHA256");
      mac.init(new SecretKeySpec(
          appSecret.getBytes("UTF-8"), "HmacSHA256"));
      byte[] hash = mac.doFinal(
          data.getBytes("UTF-8"));
      // 转十六进制
```

```
      StringBuilder sb = new StringBuilder();
      for (byte b : hash) {
        sb.append(String.format("%02x", b));
      }
      String expected = "HMAC-SHA256=" + sb.toString();
      return MessageDigest.isEqual(
          expected.getBytes(), header.getBytes());
    } catch (Exception e) {
      log.error("HMAC verify failed", e);
      return false;
    }
  }
}
```

4.7 重试策略

如果企业Webhook接口未返回 HTTP 200，乐读派将按以下策略重试：

重试次数	间隔	累计等待	说明
第1次	10秒	10秒	立即重试，处理瞬时故障
第2次	30秒	40秒	短间隔重试
第3次	2分钟	2分40秒	中间隔
第4次	10分钟	12分40秒	较长间隔
第5次	30分钟	42分40秒	最终重试，失败后放弃

首次发送 + 5次重试 = 共6次尝试。第5次重试后仍失败则标记为FAILED。

重要：6次重试全部失败后，该事件将标记为投递失败。企业可通过B3接口主动拉取遗漏数据。建议企业监控Webhook接口可用性，确保99.9%以上的成功率。

幂等要求：由于重试机制，企业可能收到重复的Webhook回调。请使用 event_id 做幂等去重，避免重复处理同一事件。

五、作品状态机

作品有两个关键状态维度：status（创作状态）和
completionStep（完成子状态）。两者组合决定作品当前所处的精确阶段。

5.1 status 状态值

状态值	含义	说明
PENDING	排队等待	A3提交后进入队列，等待处理
PROCESSING	创作中	AI正在生成故事/图片/配音
COMPLETED	创作完成	至少图文已完成，可查看作品
FAILED	创作失败	AI处理异常，需重新创作

5.2 completionStep 完成子状态

值	含义	说明
0	未完成	仍在创作中（PENDING/PROCESSING/FAILED）
1	纯图文完成	图片+文字已就绪，无配音
2	含配音完成	图片+文字+AI配音全部就绪

5.3 完整状态判断表

企业后端根据以下组合判断作品状态，决定如何处理：

status	completionStep	含义	Webhook事件	建议操作
PENDING	0	排队中	work.created	创建本地记录， 标记为待处理
PROCESSING	0	创作中	work.processing	更新进度信息， 无需其他操作
COMPLETED	1	图文完成（无配音）	work.completed	持久化完整pageList， 作品可展示
COMPLETED	2	含配音完成	work.audio_updated	更新audioUrl， 作品含配音可播放
FAILED	0	创作失败	work.failed	记录失败原因， 通知用户可重试

状态流转路径

```
PENDING(0) --> PROCESSING(0) --> COMPLETED(1) --> COMPLETED(2)
                  |                                (用户配音后)
                  +----------> FAILED(0)
```

正常流程: PENDING -> PROCESSING -> COMPLETED(step=1)
配音追加: COMPLETED(step=1) -> COMPLETED(step=2)
异常流程: PROCESSING -> FAILED

COMPLETED(step=1) 是最常见的终态。用户可能在创作完成后的任意时间追加配音，此时 completionStep 从1变为2，触发 work.audio_updated 事件。企业无需主动等待配音完成，纯图文作品即可展示。

六、企业同步方案

根据可靠性需求，企业可选择以下三种同步方案。推荐场景2作为生产环境方案。

场景1：B3 + B2 全量同步（最简单）

企业后端定时调用 B3 拉取所有作品变更，再用 B2 获取单个作品详情。不依赖Webhook，适合快速验证阶段。

步骤	接口	频率	说明
1	B3 批量查询	每5分钟	拉取指定时间范围内的所有作品变更 （支持按 updatedAfter 增量查询）
2	B2 详情查询	按需	对B3返回的每个workId调B2获取 完整pageList和状态
3	入库	-	持久化到企业DB

优点：实现最简单，无需暴露公网接口。缺点：延迟最高（最长5分钟），对我方API有轮询压力。

Java Spring Boot 定时拉取示例（场景1/场景3兜底）

```
@Component @Slf4j
public class WorkSyncScheduler {
  @Autowired private LeAiApiClient apiClient;
  @Autowired private WorkSyncService syncService;
  @Value("${leai.org-id}") private String orgId;

  /** 每30分钟兜底扫描最近1小时变更 */
  @Scheduled(fixedRate = 30 * 60 * 1000)
  public void syncFromB3() {
    String since = Instant.now()
        .minus(1, ChronoUnit.HOURS)
        .toString(); // ISO 8601
    JSONArray records =
        apiClient.batchQueryWorks(orgId, since);
    for (int i = 0; i < records.size(); i++) {
      JSONObject r = records.getJSONObject(i);
      // dataVersion 比较在 syncService 内完成
      syncService.onReceiveSync(
          "b3.sync", r);
    }
    log.info("B3 sync done, {} records",
        records.size());
  }
}
```

场景2：Webhook 主通道（推荐）

以Webhook回调作为唯一数据来源，实时接收所有事件。适合对实时性要求高的生产环境。

步骤	触发方式	说明
1	Webhook回调	乐读派主动推送 work.completed 事件， 包含完整pageList数据

2	验签+去重	验证HMAC签名，event_id幂等去重
3	入库	解析data，持久化到企业DB。 关联用户（通过phone字段）
4	确认	返回HTTP 200，乐读派停止重试

优点：实时性最好（秒级），无轮询开销。缺点：需要企业暴露公网HTTPS接口。

场景3：降级处理 — Webhook + B3兜底 + B2补偿（最可靠）

以Webhook为主通道，B3定时兜底扫描遗漏，B2单品补偿。三重保障确保数据最终一致。

层级	方式	频率	说明
主通道	Webhook回调	实时	接收所有事件，秒级同步。 覆盖99%以上的正常场景
兜底扫描	B3批量拉取	每30分钟	扫描最近1小时的变更， 补齐Webhook丢失的事件
单品补偿	B2详情查询	按需	用户在企业管理后台查看某作品时， 实时调B2刷新（缓存5分钟）

优点：最高可靠性，数据绝不丢失。缺点：实现复杂度较高（三层同步逻辑）。

6.4 三字段同步机制（必须遵守）

核心规则：乐读派所有同步数据均携带以下三个字段。企业后端处理任何同步来源（Webhook / B2 / B3）时，都必须基于这三个字段做决策。违反此规则将导致数据错乱！

字段	类型	语义	决策用途
dataVersion	INT	数据版本号，单调递增。 每次乐读派侧数据变更 +1	判断“要不要同步” 大的覆盖小的， 相等或更小则跳过
status	STRING	作品状态 (PENDING/PROCESSING/ COMPLETED/FAILED)	判断“什么方向” 决定走创建/更新/完成/ 失败处理分支
completionStep	INT	完成子步骤 0=未完成, 1=纯图文, 2=含配音	判断“完成后到哪步” 在COMPLETED状态下 区分图文/配音阶段

dataVersion 是乐读派内部维护的全局递增版本号。企业不得自行修改此值，只需在同步时做“远程 > 本地”的判断即可。首次入库时本地 dataVersion 初始化为 0。

6.5 企业同步伪代码

无论数据来自 Webhook 回调、B2 单品查询还是 B3 批量拉取，企业后端均应遵循以下统一处理逻辑：

```
@Service
@Slf4j
public class WorkSyncService {

  @Autowired private WorkRecordMapper workRecordMapper;
```

```
/** Webhook/B2/B3 统一入口 */
@Transactional
public void onReceiveSync(String event,
    JSONObject remote) {
  String workId = remote.getString("work_id");
  int remoteVer = remote.getIntValue("data_version");

  // Step 1: dataVersion 门卫
  WorkRecord local =
      workRecordMapper.selectById(workId);
  if (local != null
      && remoteVer <= local.getDataVersion()) {
    log.info("skip: remote.v={} <= local.v={}",
        remoteVer, local.getDataVersion());
    return; // 旧版本或重复，直接丢弃
  }

  // Step 2: 覆盖同步字段
  if (local == null) {
    local = new WorkRecord();
    local.setWorkId(workId);
  }
  local.setDataVersion(remoteVer);
  local.setStatus(remote.getString("status"));
  local.setCompletionStep(
      remote.getIntValue("completion_step"));
  local.setTitle(remote.getString("title"));
  local.setTags(remote.getString("tags"));
  if (remote.containsKey("page_list")) {
    local.setPageList(
        remote.getString("page_list"));
  } // B3摘要无此字段,不覆盖
  local.setPhone(remote.getString("phone"));
  local.setSyncedAt(new Date());

  // Step 3: 按status+completionStep路由业务
  String status = local.getStatus();
  if ("COMPLETED".equals(status)) {
    if (local.getCompletionStep() == 1) {
      // 纯图文完成 -> 作品可展示
      notifyAdmin("新作品就绪", local);
    } else if (local.getCompletionStep() == 2) {
      // 含配音完成 -> 更新音频URL
      mergeAudioUrls(local,
          remote.getJSONArray("audio_pages"));
    }
  } else if ("FAILED".equals(status)) {
    local.setFailReason(
        remote.getString("fail_reason"));
  }
  // PENDING/PROCESSING: 仅更新进度

  workRecordMapper.insertOrUpdate(local);
 }
}
```

关键点：Step 1 的 dataVersion 门卫是最重要的防线。它确保无论 Webhook、B2、B3 以任何顺序到达，最终数据永远是最新版本。企业不需要关心消息到达顺序。

6.6 企业DB设计推荐

企业本地DB建议将字段分为两组：乐读派同步字段（由同步逻辑覆盖写入，企业不得修改）和企业扩展字段（乐读派不碰，企业自行管理）。

一、乐读派同步字段（只被覆盖，企业不改）

字段名	类型	说明	数据来源
work_id	VARCHAR(32) PK	乐读派作品ID（主键）	Webhook / B2 / B3
data_version	INT NOT NULL DEFAULT 0	数据版本号（门卫字段） 远程>本地才覆盖	Webhook / B2 / B3 每次同步必带
status	VARCHAR(20)	PENDING / PROCESSING / COMPLETED / FAILED	Webhook / B2 / B3
completion_step	INT	0=未完成, 1=纯图文, 2=含配音	Webhook / B2 / B3
title	VARCHAR(100)	绘本标题	work.completed / work.updated
tags	JSON	标签数组	work.completed / work.updated
page_list	JSON / MEDIUMTEXT	完整页面数据（图/文/音频URL）	work.completed / B2
phone	VARCHAR(20)	创作用户手机号	work.completed
webhook_event_id	VARCHAR(64)	最后处理的事件ID（幂等）	每次Webhook回调
synced_at	DATETIME	最后同步时间	每次同步时写入

二、企业扩展字段（乐读派不碰）

字段名	类型	说明
local_user_id	BIGINT	企业侧用户ID，通过phone关联
review_status	VARCHAR(20)	审核状态（企业自定义审核流程）
reviewed_by	VARCHAR(50)	审核人
reviewed_at	DATETIME	审核时间
display_order	INT	展示排序权重
is_featured	TINYINT(1)	是否精选/推荐
category_id	BIGINT	企业自定义分类
remark	TEXT	企业内部备注

设计原则：同步字段与扩展字段严格隔离。同步逻辑只写入第一组字段，企业业务只写入第二组字段。这样即使同步覆盖发生，企业的审核状态、分类等业务数据也不会丢失。page_list 建议使用 JSON 类型（MySQL 5.7+），方便后续查询单页数据。

三、MySQL 建表语句参考

```
-- MySQL 5.7+ / 8.0
CREATE TABLE `ent_work_record` (
```

```
  `work_id`          VARCHAR(32)  NOT NULL COMMENT "乐读派作品ID",
  `data_version`     INT          NOT NULL DEFAULT 0 COMMENT "数据版本(门卫)",
  `status`           VARCHAR(20)  DEFAULT NULL COMMENT "PENDING/PROCESSING/COMPLETED/FAILED",
  `completion_step`  INT          DEFAULT 0 COMMENT "0未完成 1纯图文 2含配音",
  `title`            VARCHAR(100) DEFAULT NULL COMMENT "绘本标题",
  `tags`             JSON         DEFAULT NULL COMMENT "标签数组",
  `page_list`        JSON         DEFAULT NULL COMMENT "页面数据(图/文/音频URL)",
  `phone`            VARCHAR(20)  DEFAULT NULL COMMENT "创作用户手机号",
  `fail_reason`      VARCHAR(500) DEFAULT NULL COMMENT "失败原因",
  `webhook_event_id` VARCHAR(64)  DEFAULT NULL COMMENT "最后事件ID(幂等)",
  `synced_at`        DATETIME     DEFAULT NULL COMMENT "最后同步时间",
  -- 企业扩展字段
  `local_user_id`    BIGINT       DEFAULT NULL COMMENT "企业侧用户ID",
  `review_status`    VARCHAR(20)  DEFAULT NULL COMMENT "审核状态",
  `reviewed_by`      VARCHAR(50)  DEFAULT NULL COMMENT "审核人",
  `reviewed_at`      DATETIME     DEFAULT NULL COMMENT "审核时间",
  `display_order`    INT          DEFAULT 0    COMMENT "展示排序",
  `is_featured`      TINYINT(1)   DEFAULT 0    COMMENT "是否精选",
  `category_id`      BIGINT       DEFAULT NULL COMMENT "企业自定义分类",
  `remark`           TEXT         DEFAULT NULL COMMENT "企业备注",
  `created_at`       DATETIME     DEFAULT CURRENT_TIMESTAMP,
  `updated_at`       DATETIME     DEFAULT CURRENT_TIMESTAMP
                                  ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`work_id`),
  INDEX `idx_phone` (`phone`),
  INDEX `idx_status` (`status`),
  INDEX `idx_synced` (`synced_at`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
  COMMENT="乐读派作品同步表";
```

6.7 严禁事项

以下操作将导致严重的数据一致性问题，企业在集成时必须避免！

严禁操作	后果	正确做法
不存储 dataVersion	无法判断数据新旧，Webhook乱序或重复推送时数据被错误覆盖 ——> 数据错乱	必须持久化 data_version 字段， 每次同步前做 remote > local 判断
修改乐读派同步字段后不调 C1（保存作品）	本地修改不会同步回乐读派，下次Webhook/B3拉取时被覆盖 ——> 数据丢失	如需修改标题等字段，必须通过 C1接口同步回乐读派， 或只修改企业扩展字段
不做 B3 对账（仅依赖Webhook）	网络抖动/接口超时导致Webhook丢失，6次重试后数据永久遗漏 ——> 漏同步	至少每30分钟跑一次B3兜底扫描， 对比 dataVersion 补齐遗漏。 推荐场景3（三层同步）
直接修改 data_version（手动+1或设为大值）	后续真实同步全部被门卫拦截，相当于永久断开同步 ——> 同步中断	data_version 由乐读派维护， 企业只读不写。同步逻辑用远程值覆盖本地值

以上四条是数据一致性的生命线。集成测试时请逐一验证。如有疑问请联系乐读派技术支持。

七、接口参考

以下为V3.1方案涉及的全部接口汇总。注意：A6/A3/A20等创作接口由乐读派C端直接调用，企业后端通常只需关注Webhook接收、令牌交换和B2/B3查询接口。V3.1变更：B2不再公开，需认证或shareToken。

7.1 接口汇总表

编号	接口名称	方法	路径	调用方	认证	说明
AUTH	会话令牌交换	POST	/api/v1/auth/session	企业后端	密钥	用appSecret换取H5会话令牌(见第三章)
STS	STS临时凭证	POST	/api/v1/oss/sts-token	乐读派C端	Token	获取OSS直传凭证
A6	角色提取	POST	/api/v1/creation/extract-original	乐读派C端	Token	AI识别画作中的角色
A3	图片故事创作	POST	/api/v1/creation/image-story	乐读派C端	Token	异步创建AI绘本
B2	作品详情查询	GET	/api/v1/query/work/{workId}	企业后端	HMAC / Token/ shareToken	查询作品完整数据 (V3.1:不再公开)
B3	批量作品查询	GET	/api/v1/query/works	企业后端	HMAC	按时间范围批量查询
C1	编辑绘本信息	PUT	/api/v1/update/work/{workId}	乐读派C端	Token	编辑标题/作者/简介等
A20	AI配音	POST	/api/v1/creation/voice	乐读派C端	Token	TTS语音合成
B1	额度校验	POST	/api/v1/query/validate	企业后端	HMAC	检查剩余创作额度

认证列说明：密钥=请求体传appSecret；Token=H5会话令牌(Bearer sess xxx)；HMAC=HMAC-SHA256签名头

Android客户端仍使用HMAC-SHA256签名认证（密钥编译在APK中），H5端统一使用会话令牌。

7.2 企业后端常用接口详情

B2 作品详情查询

项目	内容
方法	GET
路径	/api/v1/query/work/{workId}
认证	认证用户: HMAC签名或Bearer Token; 分享链接: shareToken参数; 无认证无shareToken返回403

项目	内容
用途	查询单个作品的完整数据（状态、pageList、进度）

V3.1变更：B2接口不再完全公开。认证用户(HMAC签名或Bearer Token)可正常访问；分享链接通过shareToken参数获得只读访问；无认证且无shareToken将返回403拒绝。

```
// 响应示例 (COMPLETED)
{
  "code": 200,
  "data": {
    "workId": "1903686714382889000",
    "dataVersion": 3,
    "status": "COMPLETED", "completionStep": 1,
    "title": "小璃的冒险", "tags": ["冒险","成长"],
    "progress": 100,
    "pageList": [
      { "pageNum": 0, "text": "小璃的冒险",
        "imageUrl": "https://oss.../page_0.png", "audioUrl": null },
      { "pageNum": 1, "text": "阳光明媚的早晨...",
        "imageUrl": "https://oss.../page_1.png", "audioUrl": null }
    ]
  }
}
```

B3 批量作品查询

项目	内容
方法	GET
路径	/api/v1/query/works
认证	HMAC 签名
用途	按时间范围批量查询作品列表（用于兜底同步）

查询参数

参数	类型	必填	说明
orgId	String	是	机构ID
updatedAfter	String	否	起始时间（ISO 8601格式）， 如 2026-04-01T00:00:00Z
updatedBefore	String	否	截止时间（ISO 8601格式）
status	String	否	状态过滤（COMPLETED/FAILED等）
page	Integer	否	页码（默认1）
size	Integer	否	每页数量（默认20，最大100）

```
// 响应示例
{
  "code": 200,
  "data": {
    "total": 42,
```

```
  "page": 1,
  "size": 20,
  "records": [
   { "workId": "...", "dataVersion": 3, "status": "COMPLETED",
     "completionStep": 1, "title": "...", "phone": "138...",
     "updatedAt": "2026-04-02T10:32:15Z" },
   ...
  ]
 }
}
```

B3 返回的是作品摘要列表（不含 pageList）。如需完整数据，对每个 workId 调 B2 获取。建议兜底扫描时，仅对 status 或 completionStep 与本地不一致的作品调 B2。

B1 额度校验

项目	内容
方法	POST
路径	/api/v1/query/validate
认证	HMAC 签名
用途	检查机构剩余创作额度

```
// 请求
{ "orgId": "ORG001", "phone": "13800001111", "apiType": "A3" }

// 响应
{ "valid": true, "quotaRemaining": 42, "orgName": "测试机构" }
```

7.3 HMAC-SHA256 签名认证

B3、B1等需要认证的接口使用HMAC签名。签名方式与Webhook验签使用相同的 app_secret。

Header	值	说明
X-App-Key	orgId	机构标识（平台分配）
X-Timestamp	毫秒时间戳	如 1711234567890
X-Nonce	随机字符串	防重放（UUID或随机串）
X-Signature	签名结果	HmacSHA256 十六进制

```
签名算法:
1. 收集所有 query 参数 + nonce + timestamp
2. 按 key 字母序排列 (TreeMap)
3. 拼接: key1=val1&key2=val2&nonce=xxx&timestamp=yyy
4. 签名: HmacSHA256(拼接字符串, appSecret).toHex()

注意: POST JSON body 不参与签名计算!
```

时间窗口：签名有效期 5 分钟。超时将返回 HMAC_EXPIRED 错误。Nonce 5 分钟内不可重复使用。

Java Spring Boot HMAC 签名工具类

```
@Component
public class HmacSignUtil {
  @Value("${leai.org-id}") private String orgId;
  @Value("${leai.app-secret}") private String appSecret;

  /** 为 GET 请求生成带签名的 HttpHeaders */
  public HttpHeaders sign(
      Map<String, String> queryParams) {
    String nonce = UUID.randomUUID().toString();
    String ts = String.valueOf(
        System.currentTimeMillis());
    // 1. TreeMap 保证字母序
    TreeMap<String, String> sorted =
        new TreeMap<>(queryParams);
    sorted.put("nonce", nonce);
    sorted.put("timestamp", ts);
    // 2. 拼接
    StringBuilder sb = new StringBuilder();
    sorted.forEach((k, v) -> {
      if (sb.length() > 0) sb.append("&");
      sb.append(k).append("=").append(v);
    });
    // 3. HmacSHA256
    String signature = hmacSha256(sb.toString());
    // 4. 设置Header
    HttpHeaders headers = new HttpHeaders();
    headers.set("X-App-Key", orgId);
    headers.set("X-Timestamp", ts);
    headers.set("X-Nonce", nonce);
    headers.set("X-Signature", signature);
    return headers;
  }

  private String hmacSha256(String data) {
    try {
      Mac mac = Mac.getInstance("HmacSHA256");
      mac.init(new SecretKeySpec(
          appSecret.getBytes("UTF-8"),
          "HmacSHA256"));
      byte[] hash = mac.doFinal(
          data.getBytes("UTF-8"));
      StringBuilder hex = new StringBuilder();
      for (byte b : hash)
        hex.append(String.format("%02x", b));
      return hex.toString();
    } catch (Exception e) {
      throw new RuntimeException(e);
    }
  }
}
```

Java Spring Boot 调用 B2/B3 示例

```
@Service
public class LeAiApiClient {
  @Value("${leai.api-url}") private String apiUrl;
```

```
  private final RestTemplate rest;
  private final HmacSignUtil hmac;

  /** B2: 查询单个作品详情（无需签名） */
  public JSONObject getWorkDetail(String workId) {
    String url = apiUrl
        + "/api/v1/query/work/" + workId;
    String json = rest.getForObject(
        url, String.class);
    return JSON.parseObject(json)
        .getJSONObject("data");
  }

  /** B3: 批量拉取作品列表（需HMAC签名） */
  public JSONArray batchQueryWorks(
      String orgId, String updatedAfter) {
    Map<String,String> params = new HashMap<>();
    params.put("orgId", orgId);
    params.put("updatedAfter", updatedAfter);
    params.put("status", "COMPLETED");
    params.put("size", "100");

    HttpHeaders headers = hmac.sign(params);
    HttpEntity<?> entity =
        new HttpEntity<>(headers);

    String url = apiUrl
        + "/api/v1/query/works?orgId=" + orgId
        + "&updatedAfter=" + updatedAfter
        + "&status=COMPLETED&size=100";
    ResponseEntity<String> resp =
        rest.exchange(url, HttpMethod.GET,
            entity, String.class);
    return JSON.parseObject(resp.getBody())
        .getJSONObject("data")
        .getJSONArray("records");
  }
}
```

B2 为公开接口无需签名，可直接 GET 调用。B3 需要 HMAC 签名头。建议将 HmacSignUtil 和 LeAiApiClient 注册为 Spring Bean，通过 @Autowired 注入使用。

八、附录

8.1 错误码参考

错误码	常量名	含义	处理建议
200	SUCCESS	成功	-
10001	PARAM_ERROR	参数校验失败	检查必填字段和格式
10003	FORBIDDEN	无权访问	检查 orgId 与资源归属
10006	TOO_MANY_REQUESTS	请求过于频繁	降低调用频率，等待限流窗口过期
20002	ACCOUNT_LOCKED	账号已锁定	令牌交换连续5次密钥错误，等待10分钟
20006	HMAC_INVALID	签名验证失败	检查 appSecret 和签名算法
20007	HMAC_EXPIRED	请求超时(>5分钟)	检查服务器时间同步
20008	HMAC_REPLAY	Nonce 重复	使用唯一随机 Nonce
20010	SESSION_INVALID	会话令牌无效或过期	重新换取 sessionToken
30001	ORG_NOT_FOUND	机构不存在	检查 orgId
30002	ORG_NOT_AUTHORIZED	机构未授权	联系平台开通
30003	QUOTA_EXCEEDED	创作额度不足	联系平台充值
30005	WORK_NOT_FOUND	作品不存在	检查 workId
30012	QUOTA_VIDEO_EXCEEDED	视频额度不足	联系平台充值
30013	CONTENT_REJECTED	内容包含不适合儿童的元素	修改输入内容后重试
30014	A6_LIMIT_EXCEEDED	A6用户日限超出	等待下个周期
50001	FILE_TOO_LARGE	文件超过10MB	压缩后重试

8.2 常见问题 (FAQ)

Q1: Webhook接收接口必须是HTTPS吗?

生产环境强烈建议HTTPS。开发/测试阶段可使用HTTP，但正式上线前必须切换到HTTPS。可使用 ngrok 等工具在开发阶段暴露本地HTTP接口。

Q2: 用户在乐读派C端创作完成后，企业多久能收到数据?

Webhook回调通常在创作完成后1-3秒内发出。如果企业接口正常响应200，整个同步过程在秒级完成。

Q3: Webhook回调失败了怎么办?

乐读派会按退避策略重试5次（10s/30s/2m/10m/30m，最长约43分钟）。如果全部失败，建议企业使用B3接口主动拉取。同时检查企业Webhook接口的可用性和响应速度（建议3秒内返回200）。

Q4: 企业需要实现哪些接口?

最少只需1个：Webhook接收接口（POST /webhook/leai）。如果采用场景3（降级方案），还需实现B3定时拉取的定时任务。

Q5: 如何关联乐读派作品与企业用户?

Webhook回调的data中包含phone（用户手机号）字段。企业通过手机号关联到自有用户体系。建议在企业侧建立 work_id <-> user_id 的映射表。

Q6: OSS图片/音频链接会过期吗?

不会。所有生成的资源存储在阿里云OSS，链接长期有效（除非作品被删除触发清理）。企业可直接使用这些URL在管理后台展示，无需下载到本地。

Q7: 能否自定义Webhook回调URL?

可以。在乐读派管理后台的机构配置中设置 webhook_url 字段。支持按机构配置不同的回调地址。修改后立即生效。

Q8: completionStep从1变2时，会重新推送完整pageList吗?

不会。work.audio_updated 事件只推送 audio_pages（配音页面列表），不会重新推送图片和文字。企业后端按 page_num 匹配更新本地的 audio_url 字段即可。