软件自动升级与版本控制系统开发设计文档

v0.1

角色	姓名	日期
编写	待填写	待填写
校对	待填写	待填写
校正	待填写	待填写

修订记录

版本	日期	修订章节	修订内容	修订人
v0.1	2026-06-23	技术选型	明确客户端与服务端技术栈边界。	待填写
v0.1	2026-06-23	总体架构	梳理客户端、服务端和对象存储交互关系。	待填写
v0.1	2026-06-23	客户端目录结构	将目录树整理为正式表格。	待填写
v0.1	2026-06-23	本地配置文件设计	补充 app_config、client_identity、version_policy、local_state 字段说明。	待填写
v0.1	2026-06-23	Manifest 设计	补充全量清单、签名和文件完整性规则。	待填写
v0.1	2026-06-23	启动票据设计	明确票据用途、校验规则和安全边界。	待填写
v0.1	2026-06-23	客户端启动逻辑	新增 Visio 流程图和步骤表。	待填写
v0.1	2026-06-23	REST API 设计	明确设备身份接口非用户登录，补充管理员登录仅用于后台。	待填写
v0.1	2026-06-23	数据库设计	补充应用、通道、版本、策略、设备、授权和审计日志表。	待填写
v0.1	2026-06-23	对象存储目录设计	将对象存储目录整理为正式表格。	待填写
v0.1	2026-06-23	Hash 校验策略	明确更新可增量、校验必须全量。	待填写

v0.1	2026-06-23	签名策略	补充签名算法、key_id 和稳定序列化要求。	待填写
v0.1	2026-06-23	限流策略	将限流规则整理为表格。	待填写
v0.1	2026-06-23	回滚机制	补充事务状态机和失败恢复规则。	待填写
v0.1	2026-06-23	Demo 开发里程碑	按阶段整理开发计划。	待填写
v0.1	2026-06-23	Demo 验收用例	补充升级、回滚、禁用、预览版、离线等验收用例。	待填写
v0.1	2026-06-23	后续扩展方向	补充 License、灰度、多平台、日志上传等扩展方向。	待填写
v0.1	2026-06-23	设计范围与登录边界	明确本期默认可用且不做普通用户登录校验；管理员登录仅用于后台。	待填写
v0.1	2026-06-23	本期范围	明确普通用户登录本期不做，默认可用；后续可扩展权限和授权校验。	待填写

目录

1 设计范围与登录边界

本设计文档用于说明软件自动升级与版本控制系统第一版 Demo 的技术方案。设计范围包括客户端启动器、主程序启动校验、独立升级器、版本策略、manifest、签名校验、下载授权、离线运行、回滚机制、管理后台和基础服务端接口。

本期普通客户端用户默认可以使用软件，不需要登录账号。Launcher 启动检查、MainApp 启动校验、Updater 下载与替换、离线启动、离线更新包导入、版本禁用和降级控制均不做普通用户登录校验，只使用设备身份、版本策略、manifest、启动票据和签名校验。

本期客户端侧不实现普通用户登录。所有客户端请求使用设备身份凭证进行识别，设备身份凭证不代表用户账号，也不产生用户会话。

后续版本可以增加普通用户登录能力，用于用户权限校验、用户授权状态判断，以及判断该用户是否可以使用该软件。该能力属于后续扩展，不纳入本期 Demo 必做范围。

后续如增加普通用户登录，建议放在 Launcher 启动检查阶段或 MainApp 首次进入业务前，用于用户级授权状态判断。该能力不替代设备身份、版本策略、manifest、启动票据和签名校验。

管理后台管理员登录属于本期范围，仅用于保护后台版本发布、策略修改、设备管理、离线包生成和日志查看等管理操作。该登录不是普通客户端用户登录，普通客户端用户不调用管理后台管理员登录接口，也不持有后台访问令牌。

2 技术选型

2.1 客户端技术栈

客户端采用 Qt + C++ 开发。

客户端程序包括：

a) Launcher.exe

软件启动器。

b) MainApp.exe

主业务程序，用于 Demo 演示主软件启动、版本显示和 DLL 加载。

c) Updater.exe

独立升级器，负责下载、校验、替换、回滚和重启。

d) demo_plugin.dll

Demo 插件 DLL，用于验证 DLL 替换能力。

e) UpdateHelper.exe

可选辅助程序，用于后续处理 Updater 自身更新。

客户端推荐使用 Qt 模块：

a) QNetworkAccessManager：HTTP API 请求和文件下载。

b) QJsonDocument：JSON 解析。

c) QFile、QDir：文件操作。

d) QCryptographicHash：SHA-256 hash 计算。

e) QProcess：启动和关闭外部进程。

f) QTimer：定时检查版本。

g) QLockFile：防止重复启动升级器。

h) QSaveFile：安全写入本地配置文件。

i) QSettings：本地配置管理。

j) QTemporaryDir：临时目录管理。

2.2 服务端技术栈

Demo 阶段建议使用：

a) Python FastAPI。

b) SQLite。

c) MinIO。

d) Redis 可选，用于限流。

正式阶段可选：

a) Python FastAPI + PostgreSQL + Redis + MinIO / OSS。

b) Go + PostgreSQL + Redis + MinIO / OSS。

接口采用 HTTP REST JSON 方式约定。

2.3 文件存储

Demo 阶段使用 MinIO。

正式阶段可使用：

a) 阿里云 OSS。

b) 腾讯云 COS。

c) 华为云 OBS。

d) AWS S3。

e) 私有化 MinIO。

f) 企业内网对象存储。

版本文件不建议直接存入数据库。数据库只存文件元数据、hash、大小、存储路径和版本关系。

3 总体架构

系统总体架构如下：

客户端：

a) Launcher.exe

b) MainApp.exe

c) Updater.exe

d) 本地身份凭证

e) 本地版本策略缓存

f) 本地 manifest 缓存

g) staging 下载目录

h) backup 备份目录

i) logs 日志目录

服务端：

a) API 服务。

b) 管理后台。

c) 数据库。

d) 对象存储。

e) Redis 限流模块。

f) 签名服务模块。

交互流程：

a) 客户端请求服务端 API。

b) 服务端验证设备身份。

c) 服务端返回版本策略。

d) 客户端根据策略判断是否允许运行或升级。

e) 需要升级时，客户端请求下载链接。

f) 服务端生成短期对象存储下载 URL。

g) 客户端下载文件并校验。

h) 客户端替换文件并上报结果。

4 客户端目录结构

建议客户端目录如下：

目录或文件	用途
DemoApp/	客户端安装根目录
Launcher.exe	默认启动入口，负责版本准入检查和启动票据生成
MainApp.exe	主业务程序

Updater.exe	独立升级程序
UpdateHelper.exe	可选辅助程序，用于处理 Updater 自身更新
plugins/demo_plugin.dll	Demo 插件 DLL，用于验证 DLL 替换能力
bin/	运行依赖目录
config/app_config.json	客户端基础配置
config/client_identity.dat	本地设备身份凭证
config/version_policy.dat	本地版本策略缓存
config/local_state.json	本地运行状态和防回滚状态
update/manifest_cache/	manifest 缓存目录
update/staging/	更新文件下载临时目录
update/backup/	升级前备份目录
update/logs/	升级日志目录
data/	用户数据目录，不纳入 hash 校验
cache/	缓存目录
logs/	运行日志目录

说明：

a) config/client_identity.dat：本地设备身份凭证。

b) config/version_policy.dat：本地版本策略。

c) update/manifest_cache：缓存 manifest。

d) update/staging：下载新文件的临时目录。

e) update/backup：升级前备份目录。

f) update/logs：升级日志。

g) plugins：插件 DLL 目录，需要进行白名单检查。

h) data：用户数据目录，不纳入 hash 校验。

i) logs：运行日志目录，不纳入 hash 校验。

5 本地配置文件设计

5.1 app_config.json

字段	示例值
app_id	marsco_demo_app
app_name	Marsco Demo App
channel	stable
api_base_url	https://api.example.com/api/v1
platform	windows
arch	x64
current_version	1.0.0

5.2 client_identity.dat

该文件由服务端签发，客户端只验证，不自行伪造。

字段	示例值
client_id	client_xxx
device_id	device_xxx
license_id	license_xxx
app_id	marsco_demo_app
channel	stable
issued_at	2026-06-23T00:00:00Z
valid_until	2026-12-31T00:00:00Z
signature	base64_signature

5.3 version_policy.dat

字段	示例值
app_id	marsco_demo_app
client_id	client_xxx

policy_seq	1024
channel	stable
current_version	1.0.0
allow_run	True
force_update	False
allow_rollback	False
offline_allowed	True
issued_at	2026-06-23T00:00:00Z
valid_until	2026-07-23T00:00:00Z
latest_version	1.0.2
min_supported_version	1.0.0
message	当前版本允许使用
signature	base64_signature

5.4 local_state.json

用于记录本地状态，防止策略回滚和系统时间回拨。

字段	示例值
max_policy_seq	1024
last_success_run_at	2026-06-23T10:00:00Z
last_online_verified_at	2026-06-23T09:50:00Z
last_success_version	1.0.0

离线时间和策略回滚判断规则：

a) `max_policy_seq` 只能递增，不能被旧策略覆盖。

b) 新读取的 `version_policy.policy_seq` 小于 `max_policy_seq` 时，客户端应拒绝启动。

c) 当前系统时间早于 `last_success_run_at` 时，视为疑似时间回拨。

d) 当前系统时间早于 last_online_verified_at 时，视为疑似时间回拨。

e) 离线启动时必须同时满足：策略签名正确、policy_seq 未倒退、offline_allowed 为 true、当前时间未超过 valid_until、未检测到明显时间回拨。

f) 联网校验成功后更新 last_online_verified_at 和本地缓存策略。

g) 每次成功启动后更新 last_success_run_at。

6 Manifest 设计

manifest 必须是当前版本的全量受控文件清单。

字段	示例值
app_id	marsco_demo_app
version	1.0.2
channel	stable
platform	windows
arch	x64
manifest_seq	3001
created_at	2026-06-23T00:00:00Z
files	[{"path": "Launcher.exe", "sha256": "aaa", "size": 123456, "executable": true}, {"path": "MainApp.exe", "sha256": "bbb", "size": 223456, "executable": true}, {"path": "Updater.exe", "sha256": "ccc", "size": 323456, "executable": true}, {"path": "plugins/demo_plugin.dll", "sha256": "ddd", "size": 423456, "executable": false}]
signature	base64_signature

规则：

a) manifest 中列出的文件必须存在。

b) manifest 中列出的文件 hash 必须一致。

c) 受控目录下未在 manifest 中出现的 exe、dll 文件应视为异常。

d) manifest 必须由服务端私钥签名。

e) 客户端使用内置公钥验证 manifest。

f) manifest 应支持版本号、通道、平台、架构。

7 启动票据设计

Launcher 启动 MainApp 时生成短期启动票据。

推荐方式：

a) Launcher 生成 ticket 文件。

b) ticket 文件存放到系统临时目录。

c) Launcher 启动 MainApp，并传入 ticket 文件路径。

d) MainApp 读取并验证 ticket。

e) MainApp 验证成功后删除 ticket 文件。

启动参数示例：

项目	示例
启动参数	MainApp.exe --ticket-file=C:\Users\xxx\AppData\Local\Temp\marsco_ticket_xxx.json

ticket 内容：

字段	示例值
payload	{"app_id": "marsco_demo_app", "client_id": "client_xxx", "device_id": "device_xxx", "version": "1.0.0", "nonce": "uuid_xxx", "issued_at": "2026-06-23T10:00:00Z", "expires_at": "2026-06-23T10:01:00Z"}
signature	base64_signature

校验规则：

a) 签名正确。

b) 未过期。

c) app_id 一致。

d) client_id 一致。

e) device_id 一致。

f) version 一致。

g) nonce 未重复使用。

Demo 阶段可由 Launcher 生成本地签名票据。正式阶段需要区分启动票据和服务端版本策略，真正的版本准入仍以服务端签名策略为准。

启动票据安全边界：

a) 启动票据只用于防止普通用户直接双击 MainApp 绕过 Launcher。

b) 启动票据不能替代 `version_policy`、`manifest` 和 `client_identity` 的服务端签名校验。

c) 客户端不得保存服务端私钥。

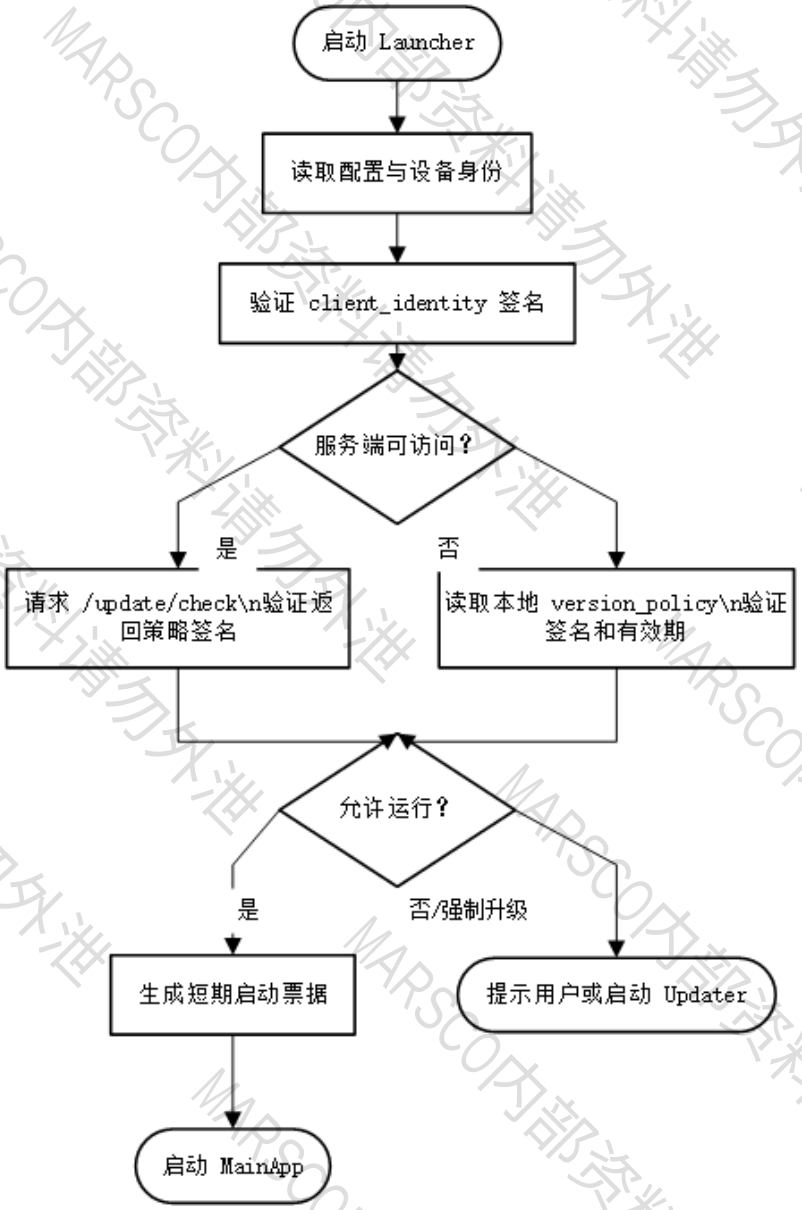
d) Demo 阶段的本地票据签名密钥只用于本机进程间启动校验，不用于签发版本策略、manifest 或设备身份。

e) 即使启动票据校验通过，MainApp 仍必须验证本地版本策略签名、有效期、版本号和核心文件 hash。

8 客户端启动逻辑

8.1 Launcher 启动逻辑

图 1 Launcher 启动准入流程



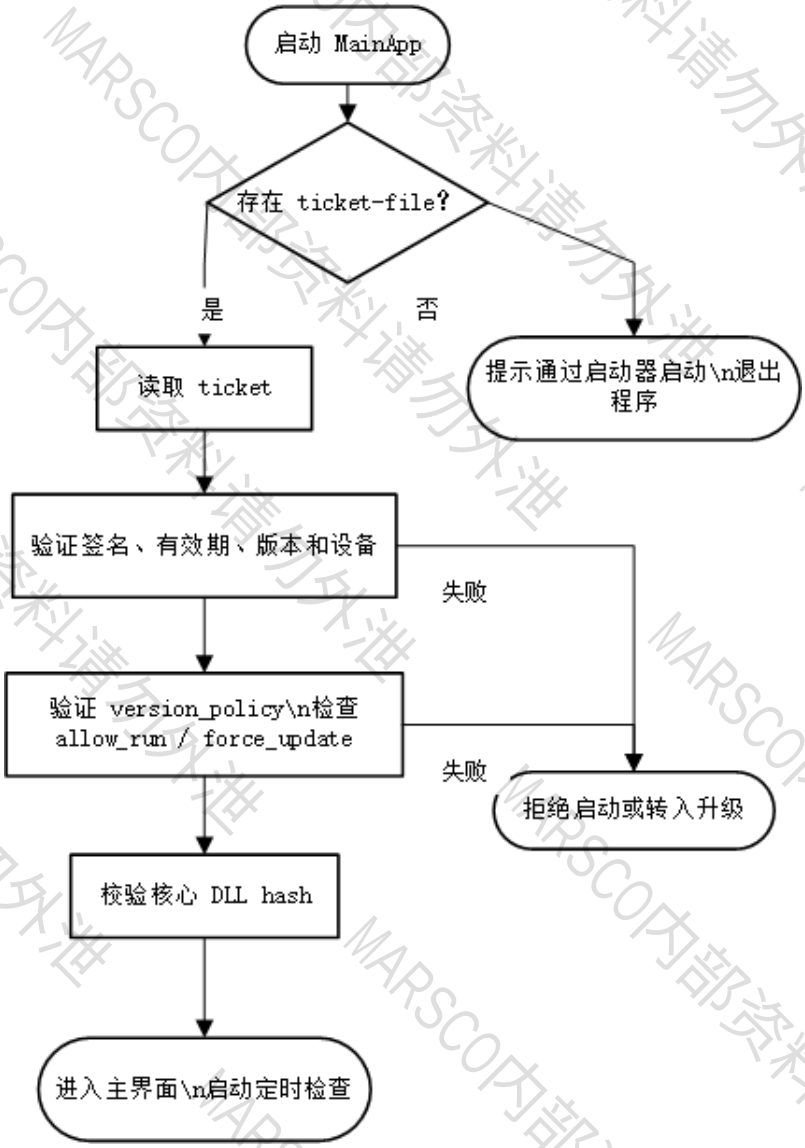
流程步骤：

步骤	判断或处理	结果
1	启动 Launcher，读取 app_config 和 client_identity	获得本地配置和设备身份
2	验证 client_identity 签名	身份无效时拒绝继续

3	尝试请求 /update/check	联网成功时进入在线策略校验
4	验证服务端返回策略签名，更新 version_policy	获得最新运行策略
5	联网失败时读取本地 version_policy	进入离线策略校验
6	验证本地策略签名、valid_until、offline_allowed	离线策略无效时拒绝启动
7	检查本地核心文件 hash	核心文件异常时进入更新或提示
8	判断是否允许运行	允许时生成启动票据并启动 MainApp
9	不允许运行或强制升级	启动 Updater 或提示用户更新

8.2 MainApp 启动逻辑

图 2 MainApp 启动校验流程



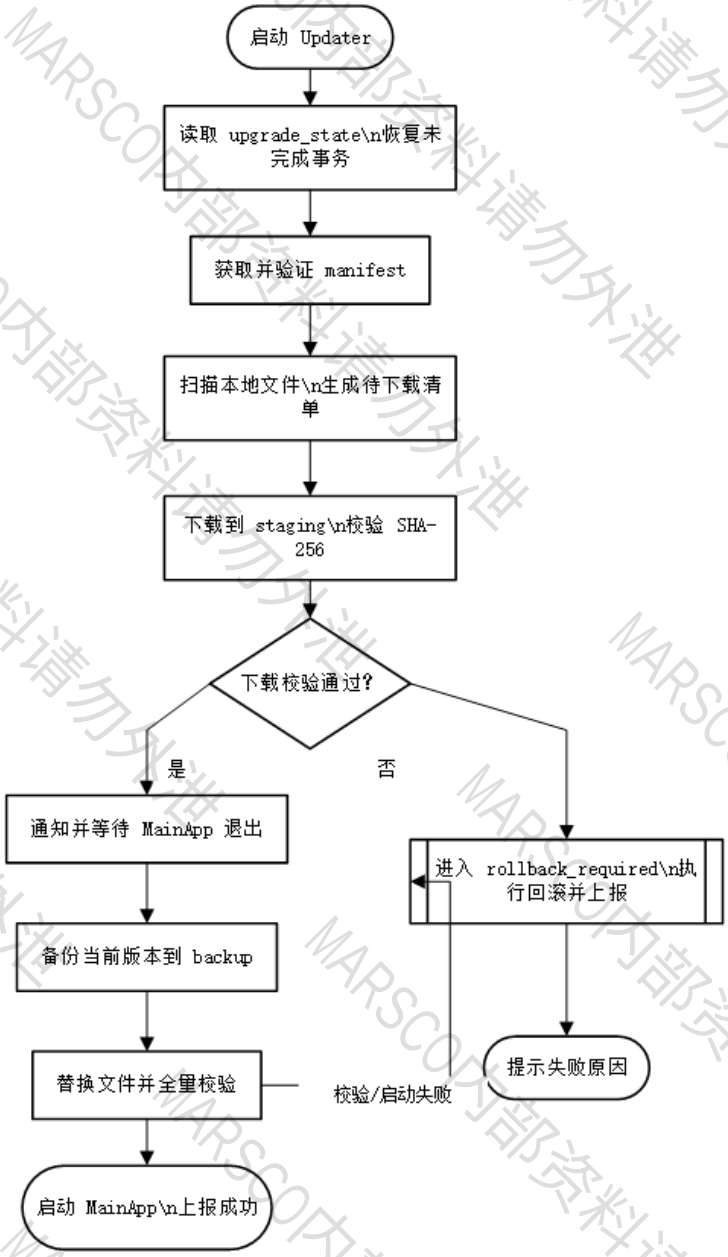
流程步骤：

步骤	判断或处理	结果
1	启动 MainApp，检查启动参数中是否存在 ticket-file	无票据时提示通过启动器启动并退出
2	读取 ticket	获得启动票据内容
3	验证 ticket 签名和有效期	票据无效时拒绝启动

4	验证 version_policy	策略无效时拒绝启动
5	验证当前版本是否允许运行	禁用版本或过期策略进入拦截
6	验证核心 DLL hash	核心文件异常时拒绝启动
7	校验通过	进入主界面
8	主界面运行期间	启动定时器，定期检查更新

8.3 Updater 升级逻辑

图 3 Updater 升级与回滚流程



流程步骤：

步骤	判断或处理	结果
1	启动 Updater，读取升级参数	获得目标版本和升级上下文

2	请求 /update/check，获取目标版本 manifest	获得版本清单
3	验证 manifest 签名	签名无效时停止升级
4	扫描本地文件并对比 manifest 全量 hash	生成待下载文件列表
5	请求临时下载 URL	获得短期下载地址
6	下载文件到 staging 并校验 hash	下载文件可用于替换
7	通知 MainApp 退出并等待完全退出	进入文件替换窗口
8	备份旧文件到 backup	形成可回滚状态
9	替换新文件并执行安装后全量校验	校验通过后启动 MainApp
10	上报升级成功	服务端记录升级结果
11	任一步骤失败	执行回滚并上报失败原因

9 REST API 设计

9.1 检查更新接口

接口：

方法	路径
POST	/api/v1/update/check

请求：

字段	示例值
app_id	marsco_demo_app
client_id	client_xxx
device_id	device_xxx

license_id	license_xxx
current_version	1.0.0
target_version	
channel	stable
platform	windows
arch	x64
operation	check

返回：

字段	示例值
allow_run	True
action	optional_update
message	发现新版本
current_version	1.0.0
latest_version	1.0.2
channel	stable
force_update	False
allow_rollback	False
policy_seq	1025
policy_valid_until	2026-07-23T00:00:00Z
manifest_id	manifest_xxx
manifest_url	https://download.example.com/xxx
signature	base64_signature

action 可选值：

字段/取值
allow

optional_update
force_update
blocked
rollback_allowed
rollback_denied
license_invalid
device_invalid
policy_expired

operation 可选值：

字段/取值
check
upgrade
downgrade

当 operation 为 downgrade 时，客户端必须传入 target_version。服务端需要判断目标版本是否存在、是否属于当前通道、是否允许降级、是否满足最低支持版本和数据兼容规则。

降级检查返回示例：

字段	示例值
allow_run	True
action	rollback_allowed
message	允许降级到 1.0.0
current_version	1.0.2
latest_version	1.0.2
target_version	1.0.0
channel	stable

force_update	False
allow_rollback	True
policy_seq	1030
policy_valid_until	2026-07-23T00:00:00Z
manifest_id	manifest_1_0_0
manifest_url	https://download.example.com/manifest-1.0.0.json
signature	base64_signature

9.2 获取下载链接接口

接口：

方法	路径
POST	/api/v1/update/download-url

请求：

字段	示例值
app_id	marsco_demo_app
client_id	client_xxx
device_id	device_xxx
license_id	license_xxx
version	1.0.2
channel	stable
files	["MainApp.exe", "plugins/demo_plugin.dll"]

返回：

字段	示例值
expires_in	600
files	[{"path": "MainApp.exe", "url": "https://oss.example.com/temp-url-1", "sha256":

	"bbb", "size": 223456}, {"path": "plugins/demo_plugin.dll", "url": "https://oss.example.com/temp-url-2", "sha256": "ddd", "size": 423456}]

9.3 升级结果上报接口

接口：

方法	路径
POST	/api/v1/update/report

请求：

字段	示例值
app_id	marsco_demo_app
client_id	client_xxx
device_id	device_xxx
license_id	license_xxx
from_version	1.0.0
to_version	1.0.2
channel	stable
result	success
error_code	
message	upgrade success
started_at	2026-06-23T10:00:00Z
finished_at	2026-06-23T10:05:00Z

result 可选：

字段/取值
success

download_failed
hash_failed
signature_failed
replace_failed
rollback_success
rollback_failed
permission_failed
disk_space_not_enough
mainapp_restart_failed

9.4 设备身份签发接口（非用户登录）

该接口用于首次签发或刷新设备身份凭证，不是客户端用户登录接口。

Demo 阶段可不暴露该接口，由测试数据或预置文件生成 `client_identity.dat`。正式阶段如需启用，也应定位为设备激活/身份签发能力，而不是用户名密码登录。

接口：

方法	路径
POST	/api/v1/device/activate

请求：

字段	示例值
app_id	marsco_demo_app
activation_code	XXXX-XXXX-XXXX
device_fingerprint	device_hash_xxx
platform	windows
arch	x64

返回：

字段	示例值
client_id	client_xxx
device_id	device_xxx
license_id	license_xxx
channel	stable
identity_token	{"valid_until": "2026-12-31T00:00:00Z", "signature": "base64_signature"}

Demo 阶段可以简化为预置 client_id，不做复杂激活流程。

9.5 管理后台管理员登录接口（非本期普通客户端用户登录）

该接口仅用于管理后台管理员登录，与本期普通客户端用户无关。普通客户端用户在本期客户端不登录，不调用该接口，也不持有该接口返回的后台访问令牌。

后续如增加普通用户登录，应单独设计客户端用户登录接口、用户授权模型和用户权限校验逻辑，不复用管理后台管理员登录接口。

Demo 阶段可使用单管理员账号。正式阶段可扩展为多管理员、角色权限、双因素认证或企业单点登录。

接口：

方法	路径
POST	/api/v1/admin/login

请求：

字段	示例值
username	admin
password	password

返回：

字段	示例值
access_token	jwt_or_session_token
token_type	Bearer
expires_in	7200

要求：

a) 密码不得明文存储。

b) 登录失败应记录审计日志。

c) 管理后台接口必须校验管理员 token。

d) Demo 阶段可不实现复杂角色权限，但必须区分未登录和已登录管理员；该状态只存在于管理后台，不存在于本期普通客户端。

9.6 管理后台接口摘要

第一版 Demo 至少需要以下后台接口：

方法	路径
POST	/api/v1/admin/apps
POST	/api/v1/admin/channels
POST	/api/v1/admin/versions
POST	/api/v1/admin/versions/{version_id}/files
POST	/api/v1/admin/versions/{version_id}/publish
POST	/api/v1/admin/policies
POST	/api/v1/admin/policies/{policy_id}/force-update
POST	/api/v1/admin/policies/{policy_id}/block
POST	/api/v1/admin/policies/{policy_id}/rollback
GET	/api/v1/admin/download-logs
GET	/api/v1/admin/upgrade-logs
GET	/api/v1/admin/audit-logs

后台接口规则：

a) 所有后台接口都需要管理员 token。

b) 发布版本时生成 manifest、计算文件 hash 并写入版本文件表。

c) 修改强制升级、禁用、降级策略时必须生成新的 `policy_seq`。

d) 每次关键操作必须写入管理员操作记录。

e) Demo 阶段可使用简化管理页面或 API 调试页面完成后台操作。

10 数据库设计

10.1 应用表 apps

字段：

字段/取值
id
app_id
app_name
description
created_at
updated_at

10.2 版本通道表 channels

字段：

字段/取值
id
app_id
channel_code
channel_name
enabled
created_at

updated_at

示例：

字段/取值
stable / 正式版
preview / 预览版
internal / 内测版
customer_a / 客户专版

10.3 版本表 versions

字段：

字段/取值
id
app_id
channel_code
version
version_name
status
is_latest
release_notes
offline_allowed
offline_valid_days
allow_rollback
rollback_targets
force_update
created_at
published_at

updated_at

status 可选：

字段/取值
draft
published
deprecated
force_update
blocked
archived

说明：

a) `allow_rollback` 表示该版本是否允许作为降级目标。

b) `rollback_targets` 可为空；为空时表示不允许主动降级到其他版本。

c) 如果需要精确控制降级目标，`rollback_targets` 记录允许降级到的版本列表，或拆分为独立版本关系表。

10.4 版本文件表 version_files

字段：

字段/取值
id
version_id
relative_path
storage_key
sha256
size
file_type
is_required

created_at

10.5 版本策略表 version_policies

字段：

字段/取值
id
app_id
channel_code
version
policy_seq
allow_run
force_update
allow_rollback
rollback_targets
offline_allowed
valid_until
min_supported_version
latest_version
message
created_at
updated_at

说明：

a) 每次策略变更必须递增 `policy_seq`。

b) 客户端记录已见过的最大 `policy_seq`，检测到倒退时拒绝使用。

c) `allow_rollback` 表示当前策略是否允许降级。

d) `rollback_targets` 用于限制允许降级的目标版本；未命中目标版本时返回 `rollback_denied`。

10.6 设备表 devices

字段：

字段/取值
id
client_id
device_id
license_id
app_id
channel_code
status
first_seen_at
last_seen_at
created_at
updated_at

status 可选：

字段/取值
active
disabled
suspicious
expired

10.7 授权表 licenses

字段：

字段/取值
id
license_id
customer_name
app_id
channel_code
max_devices
valid_until
status
created_at
updated_at

10.8 下载日志表 download_logs

字段：

字段/取值
id
client_id
device_id
license_id
app_id
version
channel_code
file_path
file_size
ip
user_agent

result
created_at

10.9 升级日志表 upgrade_logs

字段：

字段/取值
id
client_id
device_id
license_id
app_id
channel_code
from_version
to_version
result
error_code
message
started_at
finished_at
created_at

10.10 管理员表 admin_users

字段：

字段/取值
id
username

password_hash
status
last_login_at
created_at
updated_at

status 可选：

字段/取值
active
disabled

规则：

a) 密码只保存 hash，不保存明文。

b) Demo 阶段可预置一个管理员账号。

c) 正式阶段可扩展角色、权限、双因素认证和单点登录。

10.11 管理员操作日志表 admin_audit_logs

字段：

字段/取值
id
admin_user_id
action
target_type
target_id
before_data
after_data
ip
user_agent

created_at

需要记录的关键操作：

a) 登录成功和登录失败。

b) 创建或发布版本。

c) 上传版本文件。

d) 修改强制升级策略。

e) 禁用或恢复版本。

f) 修改降级策略。

g) 生成离线更新包。

h) 禁用或恢复设备。

11 对象存储目录设计

建议对象存储目录如下：

对象路径	内容
updates/{app_id}/	应用根目录
updates/{app_id}/{channel}/	版本通道目录，例如 stable、preview
updates/{app_id}/{channel}/{version}/	单个版本目录
updates/{app_id}/{channel}/{version}/manifest.json	当前版本 manifest
updates/{app_id}/{channel}/{version}/manifest.sig	manifest 签名
updates/{app_id}/{channel}/{version}/files/	版本文件目录
updates/{app_id}/{channel}/{version}/files/Launcher.exe	Launcher 可执行文件
updates/{app_id}/{channel}/{version}/files/MainApp.exe	主程序可执行文件
updates/{app_id}/{channel}/{version}/files/Updater.exe	升级器可执行文件
updates/{app_id}/{channel}/{version}/files/plugins/demo_plugin.dll	Demo 插件 DLL
updates/{app_id}/{channel}/{version}/offline/	离线更新包目录

updates/{app_id}/{channel}/{version}/offline/{package}.upd	离线更新包

规则：

a) 已发布版本目录不可覆盖。

b) 如果要修复，应发布新版本号。

c) manifest 与 files 必须保持一致。

d) 离线包由同一套 manifest 生成。

e) 对象存储 bucket 正式环境应设置为私有。

12 Hash 校验策略

系统采用 SHA-256 作为文件 hash 算法。

规则：

a) 服务端发布时计算所有受控文件 hash。

b) manifest 包含全量受控文件 hash。

c) 客户端升级前扫描本地文件。

d) 客户端只下载缺失或 hash 不一致的文件。

e) 下载完成后校验下载文件 hash。

f) 文件替换完成后执行全量 hash 校验。

g) 启动时执行核心文件 hash 校验。

h) 插件目录执行白名单检查。

结论：

更新可以增量，校验必须全量。

13 签名策略

签名采用非对称签名机制。Demo 阶段建议使用 Ed25519；如果现有技术栈不方便支持 Ed25519，可使用 RSA-2048 / SHA-256，但同一套 Demo 中必须固定一种算法。

规则：

a) 服务端保存私钥。

b) 客户端内置公钥。

c) version_policy 必须签名。

d) manifest 必须签名。

e) client_identity 必须签名。

f) 离线更新包必须签名。

g) 客户端验证签名失败时拒绝启动或拒绝安装。

客户端不得保存服务端私钥。

签名数据规则：

a) 被签名对象必须包含 `signature_alg` 和 `key_id`。

b) 计算签名前必须移除 `signature` 字段。

c) JSON 签名载荷必须使用稳定序列化规则：字段按字典序排序、无多余空白、UTF-8 编码。

d) 时间字段统一使用 UTC ISO-8601 格式。

e) 客户端根据 `key_id` 选择内置公钥进行验签。

f) 公钥轮换时，客户端可同时内置当前公钥和上一个公钥。

g) 签名失败返回或记录 `signature_failed`。

version_policy 签名字段至少包括：

字段/取值
app_id
client_id
policy_seq
channel
current_version
allow_run
force_update

allow_rollback
rollback_targets
offline_allowed
issued_at
valid_until
latest_version
min_supported_version
message
signature_alg
key_id

manifest 签名字段至少包括：

字段/取值
app_id
version
channel
platform
arch
manifest_seq
created_at
files
signature_alg
key_id

client_identity 签名字段至少包括：

字段/取值
client_id

device_id
license_id
app_id
channel
issued_at
valid_until
signature_alg
key_id

14 限流策略

服务端需要支持多维度限流。

建议限流维度：

a) client_id。
b) device_id。
c) license_id。
d) IP。
e) app_id。
f) version。
g) file_path。

建议限流规则：

维度	限制
client_id	10 分钟内最多请求 10 次下载链接
device_id	1 小时内最多下载 3 次完整包
license_id	24 小时内最多下载指定额度
IP	10 分钟内最多请求 200 次 API

超限后返回：

字段	示例值
code	RATE_LIMITED
message	请求过于频繁，请稍后再试。

15 回滚机制

升级失败时必须支持回滚。

流程：

a) 下载文件到 staging。

b) 校验 hash。

c) 备份当前版本到 backup。

d) 替换文件。

e) 替换完成后执行全量校验。

f) 启动 MainApp 验证。

g) 如果失败，使用 backup 回滚。

h) 回滚后再次校验旧版本文件。

i) 上报回滚结果。

升级事务状态文件：

Updater 应在 `update/` 目录下维护事务状态文件，例如 `update/upgrade_state.json`。

字段：

字段	示例值
transaction_id	uuid_xxx
from_version	1.0.0
to_version	1.0.1
status	prepared
manifest_id	manifest_xxx

started_at	2026-06-23T10:00:00Z
updated_at	2026-06-23T10:01:00Z
staging_dir	update/staging/uuid_xxx
backup_dir	update/backup/uuid_xxx
error_code	
message	

事务状态：

字段/取值
idle
prepared
downloading
downloaded
verified
waiting_mainapp_exit
backed_up
replacing
replaced
post_verify
committed
rollback_required
rolling_back
rolled_back
failed

状态规则：

a) 下载前进入 prepared。

b) 下载过程中进入 downloading。

c) 下载完成且 hash 校验通过后进入 verified。

d) MainApp 完全退出后才能进入 backed_up。

e) 备份完成后才能进入 replacing。

f) 替换完成后进入 post_verify 并执行全量 hash 校验。

g) 全量校验成功、MainApp 可启动后进入 committed。

h) 替换后校验失败、启动失败或异常中断时进入 rollback_required。

i) 回滚完成并校验旧版本成功后进入 rolled_back。

j) 回滚失败进入 failed，并提示用户重新安装或联系管理员。

异常恢复规则：

a) Updater 启动时先读取 upgrade_state.json。

b) 如果状态为 downloading、downloaded 或 verified，可以清理 staging 后重新下载。

c) 如果状态为 backed_up、replacing、replaced 或 post_verify，必须优先尝试回滚。

d) 如果状态为 rollback_required 或 rolling_back，继续执行回滚。

e) 如果状态为 committed 或 rolled_back，可清理旧事务状态。

f) 清理 staging 和 backup 前必须确认事务已提交或已成功回滚。

需要处理的失败类型：

a) 下载失败。

b) hash 失败。

c) 签名失败。

d) 文件占用。

e) 权限不足。

f) 磁盘空间不足。

g) 替换失败。

h) 启动失败。

i) 回滚失败。

16 Demo 开发里程碑

16.1 第一阶段：基础框架

目标：

a) 完成 Launcher、MainApp、Updater 三个 Qt 程序。

b) MainApp 能显示当前版本。

c) MainApp 能加载 demo_plugin.dll。

d) Launcher 能启动 MainApp。

e) MainApp 直接启动时能拒绝进入。

16.2 第二阶段：服务端基础能力

目标：

a) 搭建 FastAPI 或 Go API。

b) 搭建 SQLite 数据库。

c) 搭建 MinIO。

d) 实现应用、版本、通道基础数据。

e) 实现 /update/check。

f) 实现 manifest 返回。

16.3 第三阶段：在线更新

目标：

a) Updater 下载 manifest。

b) Updater 校验 manifest 签名。

c) Updater 对比本地 hash。

d) Updater 下载差异文件。

e) Updater 替换 MainApp 和 DLL。

f) Updater 自动重启 MainApp。

16.4 第四阶段：版本策略

目标：

a) 支持强制升级。

b) 支持版本禁用。

c) 支持正式版和预览版。

d) 支持动态新增通道。

e) 支持允许降级和禁止降级。

16.5 第五阶段：离线能力

目标：

a) 本地缓存 version_policy。

b) 支持离线有效期。

c) 离线凭证未过期时允许启动。

d) 离线凭证过期时禁止启动。

e) 支持离线更新包导入。

16.6 第六阶段：可靠性与安全

目标：

a) 支持升级失败回滚。

b) 支持 policy_seq 防回滚。

c) 支持系统时间回拨检测。

d) 支持插件白名单检查。

e) 支持下载限流。

f) 支持升级日志和下载日志。

17 Demo 验收用例

a) 通过 Launcher 正常启动 MainApp。
b) 直接启动 MainApp 被拒绝。
c) MainApp 显示主程序版本和 DLL 版本。
d) 后台发布 1.0.1。
e) 客户端从 1.0.0 升级到 1.0.1。
f) 升级后 DLL 版本变化。
g) 升级后 MainApp 自动重启。
h) 手动篡改 DLL 后启动失败。
i) 手动篡改 version_policy 后启动失败。
j) 后台设置 1.0.0 强制升级后客户端无法继续使用。
k) 后台禁用 1.0.0 后客户端拒绝启动。
l) 后台开放预览版后 preview 客户端获取 preview 更新。
m) 后台关闭降级通道后客户端无法降级。
n) 离线凭证未过期时可以启动。
o) 离线凭证过期后拒绝启动。
p) 模拟升级失败后成功回滚。

18 后续扩展方向

后续可扩展：

a) 完整 License 系统。
b) 客户专版分发。
c) 灰度发布。
d) 多平台支持。
e) Windows 服务模式更新。
f) 管理员权限提权。
g) 代码签名证书。
h) 自动错误收集。

i) 客户端日志上传。

j) 增量二进制差分更新。

k) 多语言界面。

l) 企业内网镜像站点。