# TOP.E OTA 详细设计文档

## 1 文档信息

### 1.1 当前版本号

v1.0

### 1.2 作者及版权



### 1.3 适用范围

- 3D打印机设备端
- 3D打印机云端服务

### 1.4 修订记录

| 序号 | 版本号 | 作者 | 时间 | 说明 |
| --- | --- | --- | --- | --- |
| 1 | v1.0 | 戈枫 | 2025-11-24 | 初始化 |

### 1.5 文档密级

内部公开

## 2 文档目的及适用对象

### 2.1 文档目的

本文档旨在为 TOP.E 3D 打印机设备端与云端之间的 OTA（Over-The-Air）固件/软件更新系统提供详细设计说明，指导开发团队进行编码实现。本系统采用全量更新策略，通过 MQTT 协议进行云端与设备端通信，并集成 GUI、device-api、Updater 等模块，由用户决定是否下载更新包，是否立即更新版本的操作，确保安全、可靠、用户友好的更新体验。

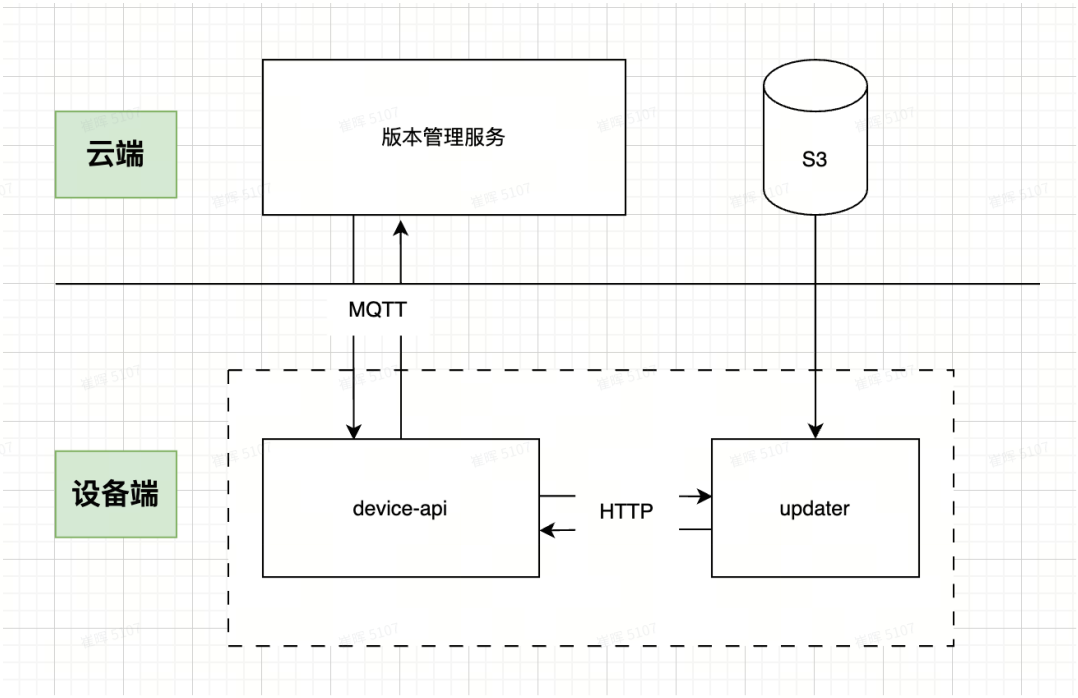
### 2.2 适用对象

1. 系统架构师
2. 前端工作师
3. 后端工程师

4. 嵌入式工作师
5. 运维工程师
6. 其他有需要的人员

# 3 系统架构及模块划分

## 3.1 系统整体架构




## 3.2 模块划分及在OTA中对应职责

| 序号 | 模块 | 应用程序 | 职责描述 |
| --- | --- | --- | --- |
| 1 | 云端 | 前端应用页面 | 用于为用户提供产品管理、版本管理、查询OTA记录的页面 |
| 2 | | 后端服务 | 用于为前端页面提供后端接口，与设备端通过 MQTT 进行通信 |
| 3 | 中间件 | OSS存储 | 用于存储版本更新包 |
| 4 | | MQTT服务 | 用于提供设备端和云端的订阅和指令 |
| 5 | 设备端 | GUI程序 | 用于提供向用户展示版本信息，用户操作检查更新、下载更新包及触发更新指令等操作 |
| 6 | | device-api | 用于保存当前版本、检查更新、与updater通信等 |
| 7 | | ai-check | 被更新服务 |

| | | | |
|---|---|---|---|
| 8 | | voice-app | 被更新服务 |
| 9 | | Moonraker | 被更新服务 |
| 10 | | klippy | 被更新服务 |
| 11 | | updater | 用于执行下载更新包、与 device-api 通信、提供下载进度、更新程序、杀死/启动程序、提供更新时的交互界面等功能 |
| 12 | | ... | |

# 4 云端设计

## 4.0 用户管理（待补充）

> 用于在当前云端的用户体系不可用于当前OTA功能时，再进行相关的处理

## 4.1 产品管理模块

### 4.1.1 模块作用

产品管理模块（Product Management Module） 是 OTA 系统云端的核心组成部分，其主要作用包括：

- 统一管理设备型号：将物理或逻辑上具有相同软硬件架构的 3D 打印机归类为一个“产品”，便于版本分发和 OTA 策略控制。
- 建立设备与固件的映射关系：确保只有适配当前产品的更新包才能下发到对应设备，防止错误刷写。
- 支持多产品线并行开发与发布：例如同时维护 M1系统打印机（printer_x1）和M2系列打印机（printer_pro_v2）。

该模块是后续版本管理和OTA 记录追踪的前提，所有 OTA 操作均需绑定到具体产品。

### 4.1.2 产品模型

产品模型包含的字段以及字段说明

| 序号 | 字段名 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- | --- |
| 1 | id | string (UUID) | 是 | 产品唯一标识（系统生成） |
| 2 | product_id | string | 是 | 产品对外 ID，如 printer_x1，用于 MQTT 别，全局唯一 |
| 3 | name | string | 是 | 产品中文/英文名称，如 “X1 桌面级 3D |
| 4 | description | string | 否 | 产品描述 |
| 5 | created_at | int | 是 | 创建时间戳 |
| 6 | created_by | string | 是 | 创建人 |
| 7 | updated_at | int | 否 | 最后更新时间 |
| 8 | updated_by | string | 否 | 最后更新人 |
| 9 | is_active | boolean | 是 | 是否启用（停用后不可创建新版本或下发 |
| 10 | deleted_at | int | 否 | 删除时间 |
| 11 | deleted_by | string | 否 | 删除人 |

> 约束：
>
> - 同一 `product_id` 不可重复

### 4.1.3 接口定义

> 约束
>
> - 所有接口遵循 RESTful 风格，返回 JSON 格式。
> - 认证方式：Bearer Token（由权限系统提供），统一在 Header 的 Authorization 中传递。
> - 若云端的用户认证体系不能支持当前 OTA 模块，则后期自行实现

#### 4.1.3.1 创建产品

1. URL：/api/v1.0/products
2. Method：POST
3. 请求头

代码块

```
Authorization: Bearer <access_token>
Content-Type: application/json
```

4. 请求体

代码块

```
# 以下为 json 格式的示例
```

```
{
  "product_id": "printer_x1",
  "name": "X1 桌面级 3D 打印机",
  "description": "面向家庭用户的 FDM 3D 打印机",
}
```

5. 响应体

代码块

```
# 以下为 json 格式的示例
{
    "code": 200,
    "msg": "成功时为 success；失败时为具体的失败原因，此时 code 不为 200",
}
```

#### 4.1.3.2 查询产品列表

1. URL：/api/v1.0/products
2. Method：GET
3. 请求头

代码块

```
Authorization: Bearer <access_token>
Content-Type: application/json
```

4. 查询参数

- page：当前页
- size：每页大小
- keyword：关键字
- order：排序，格式为 -created_at（倒序），created_by（正序）。多字段排序时以 英文逗号(,) 隔开

5. 响应体

代码块

```
# 以下为 json 格式的示例
{
    "code": 200,
    "msg": "成功时为 success；失败时为具体的失败原因，此时 code 不为 200",
    "data": [{
```

```
            // 参考 4.1.2 节
        }],
        "total": 100
    }
```

#### 4.1.3.3 更新单个产品

1. URL：/api/v1.0/product
2. Method：POST
3. 请求头

代码块

```
Authorization: Bearer <access_token>
Content-Type: application/json
```

4. 请求体

> 仅允许更新非关键字段

代码块

```
{
    "id": "系统生成的唯一标识",
    "name": "new value",
    "description": "new value",
    "is_active": true
}
```

5. 响应体

代码块

```
# 以下为 json 格式的示例
{
    "code": 200,
    "msg": "成功时为 success；失败时为具体的失败原因，此时 code 不为 200"
}
```

#### 4.1.3.4 删除单个产品

1. URL：/api/v1.0/product/delete
2. Method：POST

3. 请求头

```
Authorization: Bearer <access_token>
Content-Type: application/json
```

4. 请求体

```
{
    "id": "系统生成的唯一标识"
}
```

5. 响应体

```
# 以下为 json 格式的示例
{
    "code": 200,
    "msg": "成功时为 success；失败时为具体的失败原因，此时 code 不为 200"
}
```

## 4.2 版本管理模块

### 4.2.1 模块作用

版本管理模块（Version Management Module）是 OTA 系统云端的核心组成部分，负责对 3D 打印机固件/软件的发布版本进行全生命周期管理。其主要作用包括：

- 统一维护产品各版本信息：记录每个产品的历史版本、当前最新版本及预发布版本。
- 关联更新包资源：将版本与具体的 ZIP 更新包（含 changelog、manifest、校验信息）绑定。
- 作为 OTA 下发任务的数据源：设备触发 OTA 时，依据此模块确定目标版本和下载地址。
- 确保版本兼容性和安全性：通过版本号语义化、MD5 校验等机制保障更新可靠性。

该模块依赖于产品管理模块，所有版本必须归属于一个产品，且在新增版本时必须绑定到一个启用的产品上（`product_id`）。

### 4.2.2 版本模型

产品模型包含的字段以及字段说明

| 序号 | 字段名 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- | --- |
| 1 | id | string (UUID) | 是 | 版本唯一标识（系统生成） |
| 2 | product_id | string | 是 | 所属产品 ID（外键，关联产品管理模块） |
| 3 | version | string | 是 | 语义化版本号，格式为 v{major}.{minor}.{patch}，如 |
| 4 | name | string | 否 | 版本别名，如 “稳定版 1.2” |
| 5 | description | string | 否 | 版本描述 |
| 6 | changelog | text | 是 | Markdown 格式的变更日志内容（与 ZIP 包中的 |
| 7 | package_url | string | 是 | 更新包在OSS中的位置，格式为（bucket/path） |
| 8 | package_md5 | string | 是 | ZIP 包的 MD5 校验值（32位小写 hex） |
| 9 | package_name | string | 是 | 包文件名，符合命名规范：updater-{YYYYMMDD}-{ra |
| 10 | package_size | int | 是 | 包文件大小，以 B 以单位 |
| 11 | manifest | JSON | 是 | 待更新程序列表（与 ZIP 包中的一致） |
| 12 | is_prerelease | boolean | 是 | 是否为预发布版本（默认 false） |
| 13 | created_by | string | 是 | 创建人（用户 ID 或系统） |
| 14 | created_at | int | 是 | 创建时间 |
| 15 | updated_at | int | 否 | 最后更新时间 |
| 16 | updated_by | string | 否 | 最后更新人 |
| 17 | is_active | boolean | 是 | 是否启用（停用后不可用于 OTA） |
| 18 | deleted_at | int | 否 | 删除时间 |
| 19 | deleted_by | string | 否 | 删除人 |

### 4.2.3 接口定义

> 创建和更新版本时，file 字段中均为表单上传的文件。在列表和单个版本查询时，返回的是 OSS 桶中的位置，示例如 `tope/ota/20251124/updater-20251124-a3b9f1.zip`

#### 4.2.3.1 创建版本

1. URL：/api/v1.0/versions
2. Method：POST
3. 请求头：

代码块

```
Authorization: Bearer <access_token>
Content-Type: multipart/form-data
```

4. 请求体：

```
# 以下是 json 格式的示例
{
  "product_id": "xx-dd-3tg",
  "version": "v1.2.3",
  "name": "X1 稳定版 1.2.3",
  "description": "修复打印暂停问题，优化 GUI 响应速度",
  "changelog": "## v1.2.3\n- 修复：打印暂停后无法恢复\n- 优化：GUI 启动时间缩短30%",
  "file": "此处是 FORM 文件",
  "package_name": "updater-20251124-a3b9f1.zip",
  "manifest": {
    "version": "v1.2.3",
    "modules": [
      {"name": "gui", "src": "gui/gui_app", "dst": "/opt/printer/gui/gui_app"},
      {"name": "device-api", "src": "device-api/device-api", "dst": "/usr/bin/device-api"}
    ]
  },
  "is_prerelease": false
}
```

5. 响应体：

代码块

```
# 以下为 json 格式的示例
{
    "code": 200,
    "msg": "成功时为 success；失败时为具体的失败原因，此时 code 不为 200"
}
```

#### 4.2.3.2 查询版本列表

1. URL：/api/v1.0/versions
2. Method：GET
3. 请求头：

代码块

```
Authorization: Bearer <access_token>
Content-Type: application/json
```

4. 请求参数：

- product_id：所属产品的id
- page：当前页
- size：每页大小
- keyword：关键字
- order：排序，格式为 -created_at（倒序），created_by（正序）。多字段排序时以 英文逗号(,) 隔开

5. 响应体：

代码块

```
# 以下为 json 格式的示例
{
    "code": 200,
    "msg": "成功时为 success; 失败时为具体的失败原因，此时 code 不为 200",
    "data": [{
        // 参考 4.2.2 节
    }],
    "total": 100
}
```

#### 4.2.3.3 更新单个版本

1. URL：/api/v1.0/version
2. Method：POST
3. 请求头：

代码块

```
Authorization: Bearer <access_token>
Content-Type: application/json
```

4. 请求体：

代码块

```
# 以下是 json 格式的示例
{
  "id": "omeofnf03",
  "version": "v1.2.3",
  "name": "X1 稳定版 1.2.3",
  "description": "修复打印暂停问题，优化 GUI 响应速度",
```

```
  "changelog": "## v1.2.3\n- 修复：打印暂停后无法恢复\n- 优化：GUI 启动时间缩短30%",
  "file": "此处是 FORM 文件",
  "package_name": "updater-20251124-a3b9f1.zip",
  "manifest": {
    "version": "v1.2.3",
    "modules": [
      {"name": "gui", "src": "gui/gui_app", "dst": "/opt/printer/gui/gui_app"},
      {"name": "device-api", "src": "device-api/device-api", "dst": "/usr/bin/device-api"}
    ]
  },
  "is_prerelease": false
}
```

5. 响应体：

代码块

```
# 以下为 json 格式的示例
{
    "code": 200,
    "msg": "成功时为 success；失败时为具体的失败原因，此时 code 不为 200"
}
```

#### 4.2.3.4 删除单个版本

1. URL：/api/v1.0/version/delete
2. Method：POST
3. 请求头

代码块

```
Authorization: Bearer <access_token>
Content-Type: application/json
```

4. 请求体

代码块

```
{
    "id": "系统生成的唯一标识"
}
```

5. 响应体

```
# 以下为 json 格式的示例
{
    "code": 200,
    "msg": "成功时为 success；失败时为具体的失败原因，此时 code 不为 200"
}
```

#### 4.2.3.5 检查更新

1. 说明：

本接口由 MQTT 实现，云端OTA后端服务订阅 MQTT 的主题，在接到设备端上报的主题后，查询是否有新版本可供下载更新

2. Topic 定义：tope/<device_name>/ota/query
3. Topic 类型：Service 服务类
4. 负载：

```
{
    "command_id": "ccc",
    "product_id: "M15",
    "device_name": "M15-CT56XA",
    "version": "v1.0.0",
    "unix": 17948058604
}
```

#### 4.2.3.6 回复更新

> 待补充

## 4.3 OTA 记录模块

### 4.3.1 模块作用

OTA 记录模块（OTA Record Module） 是 OTA 系统云端的关键审计与追踪组件，负责记录每一次固件/软件更新任务的全生命周期状态。其主要作用包括：

- 追踪设备更新历史：为每个设备建立完整的 OTA 升级日志，便于问题排查与用户支持。
- 监控 OTA 任务执行状态：实时掌握更新成功率、失败原因、进度分布等关键指标。
- 支撑运维告警与分析：识别批量失败设备、版本兼容性问题等。

- 提供用户端更新状态查询依据：设备 GUI 或管理后台可基于此数据展示“上次更新时间”“更新是否成功”等信息。
- 满足合规与审计要求：保留不可篡改的更新操作记录，用于安全审计或质量追溯。

该模块与产品管理模块和版本管理模块紧密耦合，每条 OTA 记录必须关联一个有效的 `product_id` 和 `version`。

### 4.3.2 记录模型

每条 OTA 记录在数据库中以如下字段表示：

| 序号 | 字段名 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- | --- |
| | id | string (UUID) | 是 | 记录唯一标识（系统生成） |
| | device_name | string | 是 | 设备唯一 ID（如 dev_abc123xyz），全局唯一 |
| | product_id | string | 是 | 所属产品 ID（外键，关联产品管理模块） |
| | version | string | 是 | 目标更新版本（如 v1.2.3） |
| | version_id | string (UUID) | 是 | 关联的版本记录 ID（外键，关联版本管理模块） |
| | current_version | string | 是 | 设备触发更新前的当前版本 |
| | status | enum | 是 | 任务状态（见下表） |
| | trigger_type | enum | 是 | 触发方式：manual（用户手动）、auto（开机自务） |
| | package_url | string | 是 | 实际使用的更新包下载地址（快照，防止版本被 |
| | package_md5 | string | 是 | 包的 MD5 校验值（快照） |
| | progress | integer | 是 | 进度百分比（0~100） |
| | error_code | string | 否 | 错误代码（如 DOWNLOAD_FAILED, MD5_MISMATCH） |
| | error_message | string | 否 | 详细错误信息（设备上报） |
| | started_at | datetime | 否 | 任务开始时间 |
| | completed_at | datetime | 否 | 任务完成时间（成功或失败） |
| | created_at | datetime | 是 | 记录创建时间（即任务下发时间） |
| | updated_at | datetime | 是 | 最后状态更新时间 |

`status` 枚举值说明：

| 值 | 说明 |
| --- | --- |
| pending | 任务已下发，等待设备响应 |
| downloading | 设备正在下载更新包 |
| verifying | 下载完成，正在校验 MD5 |
| installing | 正在解压并部署文件 |
| rebooting | 更新完成，设备正在重启服务 |
| success | 更新成功完成 |
| failed | 更新失败（含具体 error_code） |
| cancelled | 用户取消更新（仅适用于 manual 触发） |

> 约束：
>
> - 每个 `(device_id, version)` 组合可存在多条记录（允许多次尝试）
> - 一旦状态变为 `success` / `failed` / `cancelled`，不可再修改
> - `progress` 仅在 `downloading` ~ `installing` 阶段有效

### 4.3.3 接口定义

#### 4.3.3.1 创建OTA记录

> 待补充

#### 4.3.3.2 查询OTA记录列表

1. URL：/api/v1.0/ota
2. Method：GET
3. 请求头：

代码块

```
Authorization: Bearer <access_token>
Content-Type: application/json
```

4. 请求参数

- product_id：所属产品的id
- page：当前页
- size：每页大小
- keyword：关键字，可以是版本号、设备名等

- order：排序，格式为 -created_at（倒序），created_by（正序）。多字段排序时以 英文逗号(,) 隔开

5. 响应体

代码块

```
# 以下为 json 格式的示例
{
    "code": 200,
    "msg": "成功时为 success; 失败时为具体的失败原因，此时 code 不为 200",
    "data": [{
        // 参考 4.3.2 节
    }],
    "total": 100
}
```

# 5 设备端设计

## 5.1 device-api 设计

### 5.1.1 开机检查更新设计

> 待补充

### 5.1.2 定时检查更新设计

> 待补充

### 5.1.3 手动检查更新设计

#### 5.1.3.1 用户检查更新

1. URL：/api/v1.0/ota/check
2. Method：POST
3. 请求头

代码块

```
Content-Type: application/json
client_id: tope_888
```

4. 请求体

代码块

```
{}
```

5. 响应体

代码块

```
{
    "code": 200,
    "msg": "success",
    "data: {
        "current_version": "v1.0.0",
        "new_version": "v1.0.1",
        "size": 984756  // 以 B 为单位
    }
}
```

#### 5.1.3.2 用户下载更新包

1. URL：/api/v1.0/ota/download
2. Method：POST
3. 请求头：

代码块

```
Content-Type: application/json
client_id: tope_888
```

4. 请求体：

代码块

```
{}
```

5. 响应体：

代码块

```
{
    "code": 200,
    "msg": "success"
}
```

#### 5.1.3.3 查询下载进度

1. URL：/api/v1.0/ota/progress
2. Method：GET
3. 请求头：

```
代码块
1   Content-Type: application/json
2   client_id: tope_888
```

4. 请求参数：

无

5. 响应体：

```
代码块
1   {
2       "code": 200,
3       "msg": "",
4       "status": "downloading",  // downloading, check md5, finish
5   }
```

#### 5.1.3.4 更新固件

1. URL：/api/v.0/ota/update
2. Method: POST
3. 请求头：

```
代码块
1   Content-Type: application/json
2   client_id: tope_888
```

4. 请求体：

```
代码块
1   {}
```

5. 响应体：

```
{
    "code": 200,
    "msg": "success"
}
```

### 5.1.4 订阅检查更新的回复

> 待补充

### 5.1.5 更新进度

1. URL：/api/v1.0/ota/report
2. Method：POST
3. 请求头：

```
Content-Type: application/json
client_id: tope_888
```

4. 请求体：

```
{
    "status": "downloading"
    "msg": ""
}
```

5. 响应体：

```
{
    "code": 200,
    "msg": "success"
}
```

### 5.1.6 向云端上报OTA进度

> 待补充

## 5.2 updater 设计