**OTA云端服务**

概要设计说明书

编 制 许小常\_\_\_\_\_

审 核 覃树才 \_\_

授权范围 智能网联研究院

文件编号 \_\_\_\_\_\_\_\_\_\_\_\_\_\_

版 本 v1.0\_\_\_\_\_\_

涉密截止日期 2025.02.21\_\_\_

**宝能汽车智能网联研究院**

2020年6月6日

**修订记录：**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **作者** | **修订内容** |
| V1.0 | 20200608 | 许小常 | 初版编制 |
| V1.0 | 20200810 | 韩东 | 完善接口设计手机APP服务和APP消息推送 |
| V1.0 | 20200814 | 许小常 | APP服务和推送内容细化 |
| V1.0 | 20200821 | 许小常 | 1. 添加TBOX上行消息指令：  * 添加前置条件检查指令  1. TBOX收到版本检查结果/远程下载/远程安装指令需要与OTA云端确认，OTA云端确认升级状态信息。 2. 重新定义了TBOX远程升级状态表 |
| V1.0 | 20200826 | 许小常 | 1. 修改TBOX上行消息指令：  * 下载进度汇报消息指令中添加了下载终止/下载等待/下载完成/下载完成（成功/失败）等状态上报 * 升级进度汇报消息指令中添加了升级前置条件检查失败/升级完成/升级完成（成功/失败）等状态上报  1. 。 2. 重新定义了TBOX远程升级状态表 |
| V1.0 | 20201028 | 许小常 | 1. 新增OTA升级主控程序中需要补充的属性字段 2. 提取通用参数到消息第一层级（标红+标注） |
| V1.0 | 20201115 | 许小常 | 1. 补充OTA云端架构图 2. 策略和升级计划逻辑修改 3. 补充设备信息同步信息说明 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**保密条款**

本文件受到版权法保护，宝能汽车保留所有权利。未经宝能汽车的书面明确许可，不得将本文件的全部或部分影印、复制或翻译成其他语言。本文件包含的专有信息，未经宝能汽车事先的书面许可，不能被第三方使用或借鉴。

宝能汽车有限公司智能网联研究院

目录

[1. 文档介绍 6](#_Toc26940)

[1.1. 文档目的 6](#_Toc26228)

[1.2. 适用范围 6](#_Toc4740)

[1.3. 术语与解释 6](#_Toc6171)

[1.4. 文档约定 6](#_Toc19250)

[2. 背景概述 7](#_Toc712)

[3. 设计目标 7](#_Toc21540)

[1、 统一平台 7](#_Toc30628)

[2、 丰富的升级策略 7](#_Toc1692)

[3、 安全性 7](#_Toc27833)

[4、 高效性 7](#_Toc17155)

[5、 可靠性 7](#_Toc8496)

[4. 整体架构 8](#_Toc12987)

[4.1. 架构图 8](#_Toc19539)

[4.2. 模块组成 8](#_Toc23949)

[4.3. 系统交互流程 8](#_Toc30158)

[4.4. 版本管理模式 10](#_Toc22386)

[5. OTA后台管理 10](#_Toc18626)

[5.1. 相关表说明 11](#_Toc12750)

[5.2. 设备树/标签管理 11](#_Toc166)

[5.2.1. 设备树管理 11](#_Toc29525)

[5.3. 固件升级对象服务 12](#_Toc17939)

[5.3.1. 升级对象管理 12](#_Toc13354)

[5.3.2. 升级对象（车辆）清单管理 13](#_Toc17120)

[5.4. 固件元数据及版本管理 13](#_Toc10242)

[5.4.1. 固件管理 14](#_Toc12351)

[5.4.2. 固件版本管理 14](#_Toc28238)

[5.4.3. 固件升级包管理 15](#_Toc30401)

[5.5. 任务调度管理 18](#_Toc180)

[5.6. 升级过程管理 18](#_Toc30565)

[5.7. 升级任务管理 18](#_Toc6758)

[5.7.1. 背景 18](#_Toc3607)

[5.7.2. 主要流程 18](#_Toc1813)

[5.8. 升级进度管理 19](#_Toc8535)

[5.8.1. 总体情况 19](#_Toc32400)

[5.8.2. 任务监控 19](#_Toc28632)

[5.9. 策略管理 19](#_Toc6740)

[5.9.1. 定义 20](#_Toc5208)

[5.9.2. 新建策略 20](#_Toc7313)

[5.9.3. 策略审核 20](#_Toc14598)

[5.10. 升级日志管理 21](#_Toc15480)

[5.11. 升级对象（车辆）信息同步 21](#_Toc32548)

[5.11.1. 设备树同步 22](#_Toc2504)

[5.11.2. 车辆信息同步 22](#_Toc1237)

[5.11.3. 车辆标签信息同步 24](#_Toc18932)

[5.11.4. 地理位置信息同步 24](#_Toc561)

[6. 接口设计 25](#_Toc864)

[6.1. 设备树管理 (OTA管理) 25](#_Toc16)

[6.1.1. 设备树接口 25](#_Toc20347)

[6.1.2. 获取设备树下级叶子节点列表 26](#_Toc32309)

[6.2. 固件管理 (OTA管理) 28](#_Toc4809)

[6.2.1. 添加固件接口 28](#_Toc19385)

[6.2.2. 新建固件版本接口 31](#_Toc29205)

[6.3. 固件版本管理 (OTA管理) 32](#_Toc11240)

[6.3.1. 固件版本列表接口 32](#_Toc22120)

[6.4. 固件升级包管理 (OTA管理) 34](#_Toc29424)

[6.4.1. 升级包列表接口 34](#_Toc7539)

[6.4.2. 添加全量升级包接口 35](#_Toc24843)

[6.4.3. 添加差分升级包接口 37](#_Toc17037)

[6.4.4. 添加补丁升级包接口 39](#_Toc30741)

[6.5. 升级任务管理 (OTA管理) 40](#_Toc27775)

[6.5.1. 新建任务接口 40](#_Toc9897)

[6.5.2. 删除任务接口 42](#_Toc28326)

[6.5.3. 启用任务接口 43](#_Toc15665)

[6.5.4. 任务列表接口 44](#_Toc56)

[6.5.5. 新增任务固件清单接口 46](#_Toc1181)

[6.5.6. 新增升级对象车型清单接口 47](#_Toc17075)

[6.5.7. 查询任务详情接口 49](#_Toc19346)

[6.6. 升级任务调度 50](#_Toc14525)

[6.7. 面向TBox的消息上行 51](#_Toc9812)

[6.7.1. 消息格式通用说明 51](#_Toc6394)

[6.7.2. TBOX设备注册请求/确认发送（来源于TBox：保留） 55](#_Toc16054)

[6.7.3. 获取服务端配置 56](#_Toc1191)

[6.7.4. OTA版本检查请求 59](#_Toc15572)

[6.7.5. ECU状态同步上报（来源于TBox） 66](#_Toc10236)

[6.7.6. OTA新版本下载请求发送（来源于TBox） 66](#_Toc22123)

[6.7.7. OTA版本下载确认结果发送（来源于TBOX） 68](#_Toc15004)

[6.7.8. OTA版本升级包下载 71](#_Toc16893)

[6.7.9. OTA下载进度汇报 72](#_Toc18599)

[6.7.10. OTA版本安装确认请求 76](#_Toc29747)

[6.7.11. OTA版本安装确认完成 78](#_Toc12578)

[6.7.12. OTA升级进度/状态汇报 80](#_Toc271)

[6.7.13. OTA升级结果汇报 84](#_Toc27853)

[6.7.14. 升级日志和错误信息文件上传 87](#_Toc2029)

[6.7.15. OTA终止升级结果上行（预留） 88](#_Toc2224)

[6.8. 面向Tbox的消息下行 89](#_Toc2157)

[6.8.1. 唤醒机制 89](#_Toc22997)

[6.8.2. 云端升级通知下行 89](#_Toc12210)

[6.8.3. APP客户端请求TBOX版本检查消息下行 90](#_Toc23563)

[6.8.4. 版本检查结果下发 91](#_Toc26785)

[6.8.5. OTA下载确认结果下行（点击“远程下载”） 94](#_Toc31665)

[6.8.6. OTA安装确认结果下行（确认远程安装） 95](#_Toc3838)

[6.8.7. OTA取消预约安装指令下行 97](#_Toc19152)

[6.8.8. OTA任务终止消息下行 98](#_Toc25759)

[6.9. OTA 手机App服务 98](#_Toc25616)

[6.9.1. API接口请求通用说明 98](#_Toc8464)

[6.9.2. 获取当前TBOX升级状态 99](#_Toc4684)

[6.9.3. OTA版本检查请求 101](#_Toc30542)

[6.9.4. OTA远程下载确认 104](#_Toc18900)

[6.9.5. OTA安装确认 106](#_Toc2707)

[6.9.6. OTA自动下载配置同步 109](#_Toc10388)

[6.10. 面向手机App的消息推送 110](#_Toc21060)

[6.10.1. 消息推送通用格式 110](#_Toc7366)

[6.10.2.](#_Toc11901) ~~[版本检查结果通知（废弃：暂时保留，相关功能由”OTA远程下载确认请求通知”承载）](#_Toc11901)~~ [111](#_Toc11901)

[6.10.3. OTA下载确认请求（携带版本检查结果） 112](#_Toc3538)

[6.10.4. OTA下载确认结果（TBOX 收到下载指令确认） 114](#_Toc14801)

[6.10.5. OTA安装确认请求（远程下载完成） 115](#_Toc392)

[6.10.6. OTA安装确认结果（TBOX端收到安装确认指令） 117](#_Toc2616)

[6.10.7. OTA下载进度推送 118](#_Toc18398)

[6.10.8. OTA安装进度推送（包括前置条件检查失败） 120](#_Toc25152)

[6.10.9. OTA安装结果推送 122](#_Toc18892)

[6.11. OTA 运维服务 123](#_Toc7424)

[6.11.1. OTA版本版本诊断 123](#_Toc15374)

# 文档介绍

## 文档目的

本文档是针对OTA升级服务架构设计，主要包含OTA相关概念定义、架构分层、固件升级包核心功能构成等内容及设计。本阶段该文档主要用于指导OTA升级服务相关模块开发。

## 适用范围

研发部设计、开发、测试和所有被授权人员

## 术语与解释

|  |  |
| --- | --- |
| **术语/缩写** | **解释** |
| OTA升级 | 是终端通过无线网络下载远程服务器上的升级包,对系统或应用进行升级的技术 |
| MS-OTA | OTA升级服务名称 |
| TSP | Telematics Service Provider，特指车联网平台 |
| 固件 | 挂接在设备零件上的软件 |
| 设备树 | 定义设备零件立体管理的数据结构 |
| 全量升级 | 适用版本到目标版本的全量包升级方式 |
| 补丁升级 | 用于定义两个确定版本之间的补丁包升级方式 |
| 差分升级 | 根据起始版本制作多个二进制差分包升级方式 |

## 文档约定

* 接口通用约定

后台提供GET和POST两种类型的请求;基于restful风格接口设计规范。后端返回采用统一的json数据结构，必定包含respCode,respMsg,respData三个字段.结构示例如下：

{

“respCode”:”00000”, // 业务异常码，成功：00000；

“respMsg”:”Success” //

“respData”:obj //

}

* + 错误码约定

错误码为字符串类型，共5位，分成两个部分：错误产生来源+四位数字编号。 说明：错误产生来源分为A/B/C，A表示错误来源于用户，比如参数错误，用户安装版本过低，用户注册错误等问题；B表示错误来源于当前系统，往往是业务逻辑出错，或程序健壮性等问题；C表示错误来源于第三方服务，比如极光推送服务出错，消息投递超时等问题；四位数字编号从0001到9999，大类之间的步长间距预留100。

# 背景概述

近年来，随着车联网系统和技术的不断进步和发展，OTA在汽车上的应用也越来越多。通过OTA（Over－the－Air Technology）空中下载技术，可以实现对汽车上软硬件系统在线更新升级。OTA升级平台的建设正是由此应用而生。

# 设计目标

1. 统一平台
   * 一个独立的OTA平台，支持多车场或客户的设备的接入

支持多种设备类型和软件类型

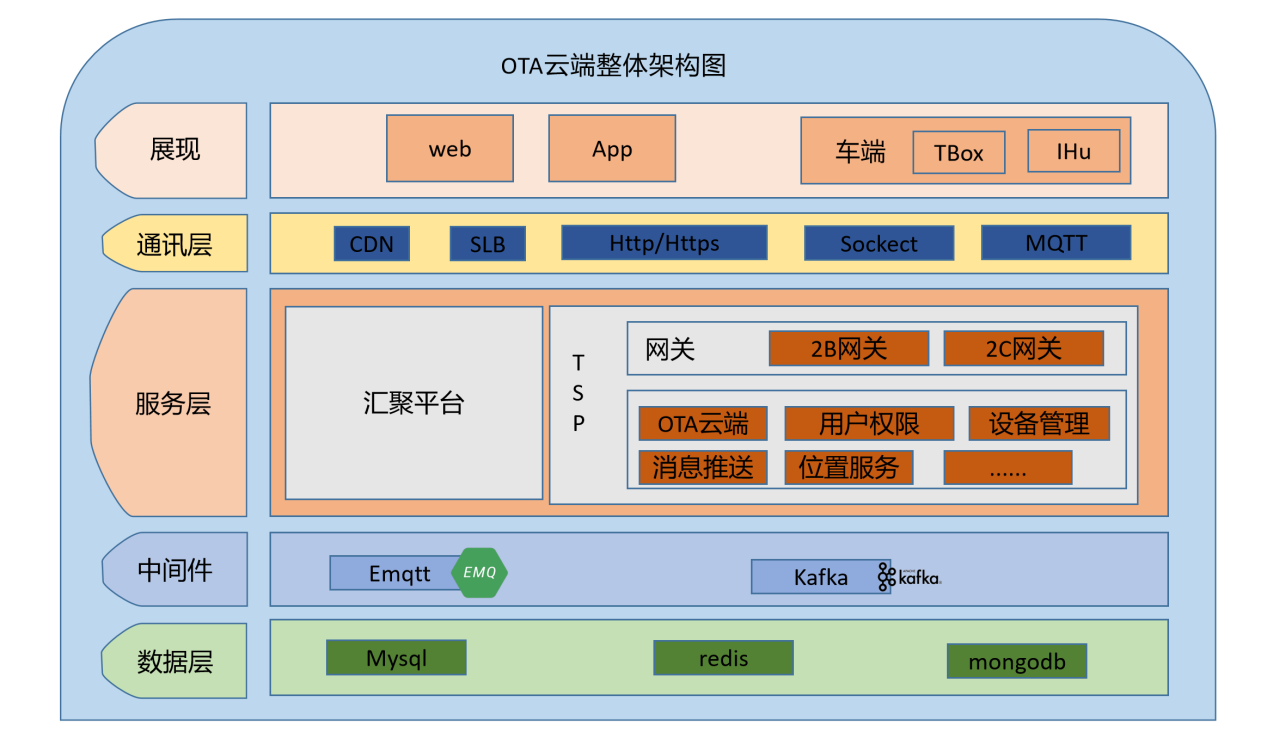
* + 支持多车系，多型号规格，多种类型设备软件（比如TBOX、IPDA、OBD、IVI等）的升级
  + 支持其他非车辆相关软件的升级，为复杂的软件升级提供一套通用的解决方案。

支持OS系统、固件、驱动、SDK、导航软件、其他APP程序等的OTA。

1. 丰富的升级策略
   * 支持全量升级，补丁升级(补丁增量包)，差分升级（二进制级别）。
   * 支持用户指定时间升级（延期或特定时间）
   * 支持静默升级,非静默升级(下载确认，更新确认，升级状态提示)。
   * 支持主动升级和被动通知升级
   * 支持版本的迭代，更新，版本的分支等。
   * 采用批量任务管理分批升级，避免升级拥堵
2. 安全性
   * 支持软件包的加密/解密和签名/验签
   * 双向认证支持
3. 高效性
   * CDN分发，终端通过最近的接入点接入
   * 断点续传，部分错误重传，避免整包重新下载
4. 可靠性
   * 数据包的签名验证，确保数据的完整性和可靠性

# 整体架构

## 架构图



## 模块组成

OTA云端主要包括OP管理平台，OTA升级服务，任务调度服务，OTA差分服务，OTA文件服务几大模块。

OP管理平台需要提供基础的用户，角色，权限，授权，鉴权等的管理和服务，需要提供基础的多项目能力支持。

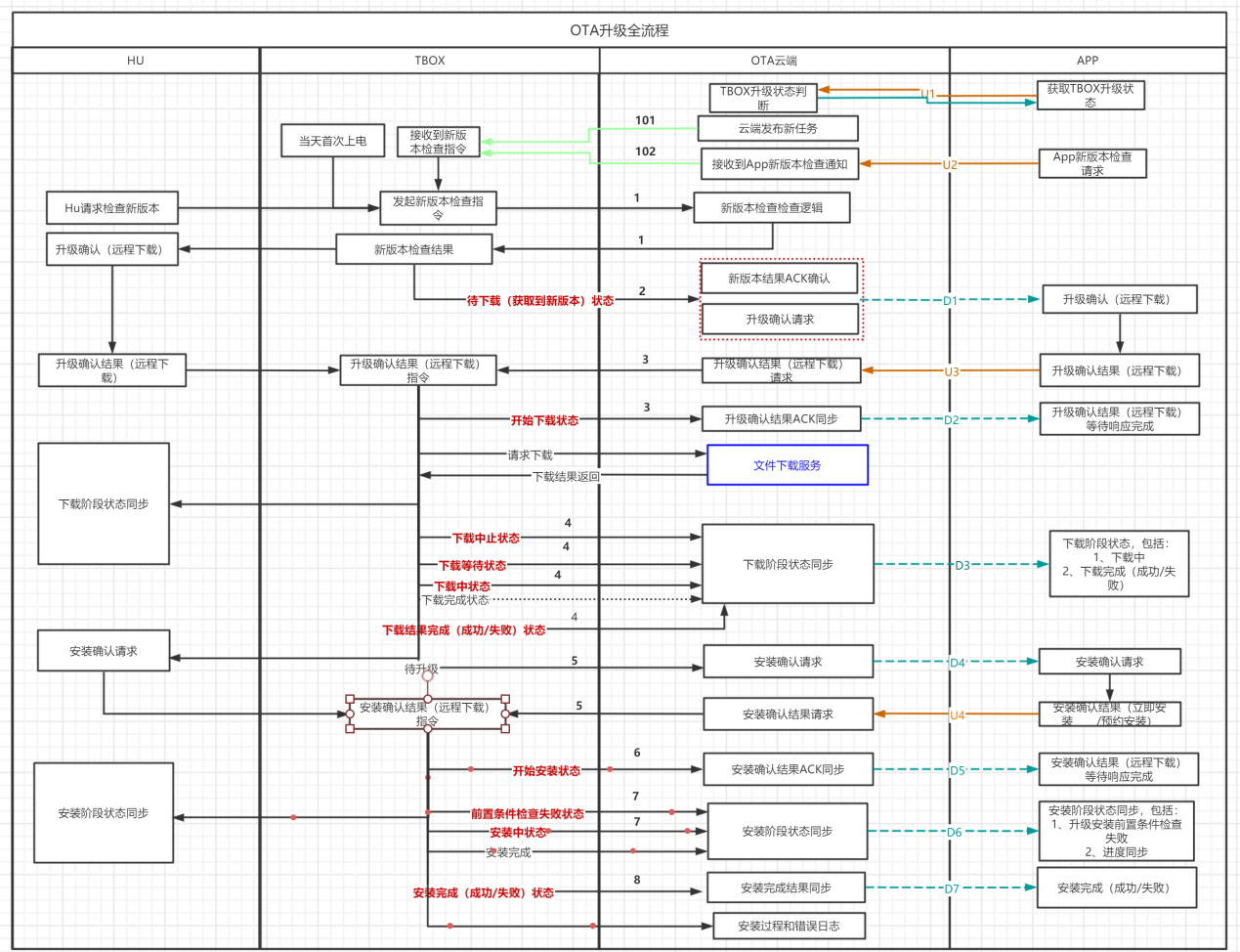
OTA管理将以子模块的形式运行在OP平台。

任务调度系统包含任务调度服务和任务调度管理，两部分，任务调度管理作为OTA依赖的子模块运行于OP平台。

OTA通知指令的下行需要外部系统按接口要求实现和提供下行通知的接口。

## 系统交互流程

OTA升级交互流程图：



图：OTA升级交互流程图

说明：

1. OTA云端与TBOX交互消息包括：

TBOX上行消息到OTA云端

* 1、版本检查
* 2、升级确认请求\新版本ACK确认
* 3、升级确认结果
* 4、下载进度汇报
* 5、安装确认请求
* 6、安装确认结果
* 7、安装进度汇报
* 8、安装结果汇报

补充：升级过程/错误日志上传提供http协议

OTA云端下行消息到TBOX

* 1、版本检查结果
* 3、升级确认结果
* 6、安装确认结果
* 101、云端新版本通知
* 102、APP客户端版本检查

1. OTA云端与APP交互消息包括：

APP上行消息到OTA云端（HTTP协议）

* U1、升级状态查询
* U2、版本检查
* U3、升级确认结果
* U4、安装确认结果

OTA云端下行消息到APP

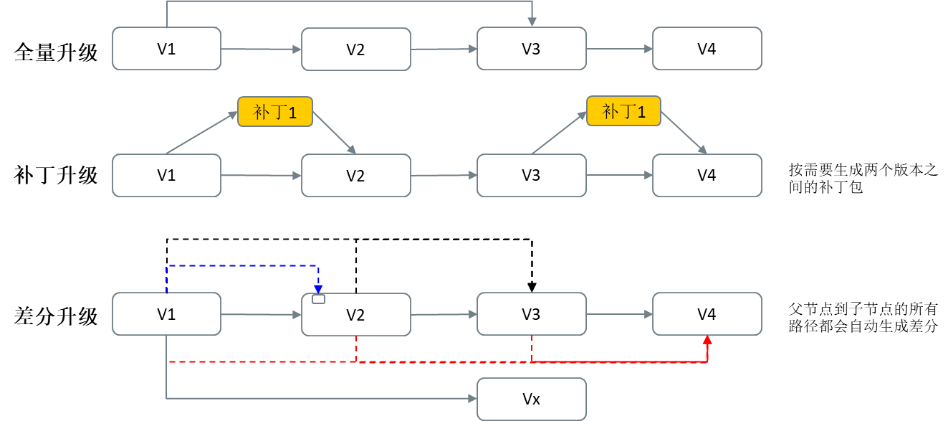
* D1、OTA远程下载确认请求（携带版本检查结果）
* D2、OTA下载确认结果
* D3、OTA下载进度推送
* D4、OTA安装确认请求
* D5、OTA安装确认结果
* D6、OTA安装进度推送
* D7、OTA安装结果推送

具体各段交互参考后面小节详细说明。

## 版本管理模式

OTA版本采用单一路径升级模式，一个软件一个升级路径

路径各个节点之间允许存在差分包或者补丁包



# OTA后台管理

OTA后台管理主要包括设备树/标签管理、升级对象管理、固件及版本管理、任务调度管理、升级过程管理等模块。

* 设备树管理主要用来提供零件搜索和定位，并为后端管理操作提供可视化导航。便签管理用于对需要升级的对象设置不同的便签信息，便于定位要升级的对象（当前指代车辆）。
* 升级对象服务主要用于定义和管理升级对象，当前升级对象为车辆。该服务需提供升级车辆信息导入/初始化，获取升级对象固件清单列表等操作。
* 固件及版本管理包括元数据管理和固件版本管理，如下：
* 固件元数据管理模块需要提供元数据管理维护功能，可以通过初始化导入 或接口方式添加固件信息来完善相关元数据。该模块为固件版本管理提供基础支 撑。
* 固件版本管理需提供对软件版本的管理维护功能，包括新增固件版本、查询 固件版本列表、新增升级包、生成发布包、升级包路径管理等。该模块需要文件 服务提供原始包/安装包上传、下载、分发等功能支持。如果升级包支持差分升级 方式，需要后台提供差分规则和算法支撑。
* 任务调度系统包含任务调度服务和任务调度管理两部分，任务调度管理作为OTA依赖的子模块运行于OP平台。
* OTA升级过程管理需要提供升级记录、升级过程信息记录、升级过程错误、升级进度上报持久化和查询等功能支持。

## 相关表说明

* tb\_device\_tree\_node：设备树数据表，用于存储设备立体层次数据关系
* tb\_fota\_firmware：固件元数据表，存储硬件对应的软件信息
* tb\_fota\_software\_version：固件版本表，存储固件该版本信息，关联固件表。
* tb\_fota\_software\_version\_path：固件版本升级路径表，记录全量包、补丁包和差分包的升级条件信息，记录从适应的版本到当前版本的升级路径。
* tb\_fota\_software\_version\_dependence：固件版本依赖表，用于保存目标版本依赖的其他ecu固件版本信息。
* tb\_fota\_file\_upload\_records：文件上传记录表，用于记录文件上传过程信息。

## 设备树/标签管理

### 设备树管理

#### 定义

设备树：通过构建层级数据结构把设备零件数据持久化到叶子节点，形成统一设备管理结构，同时便于后台管理界面导航操作。

设备树结构简单示例如下：

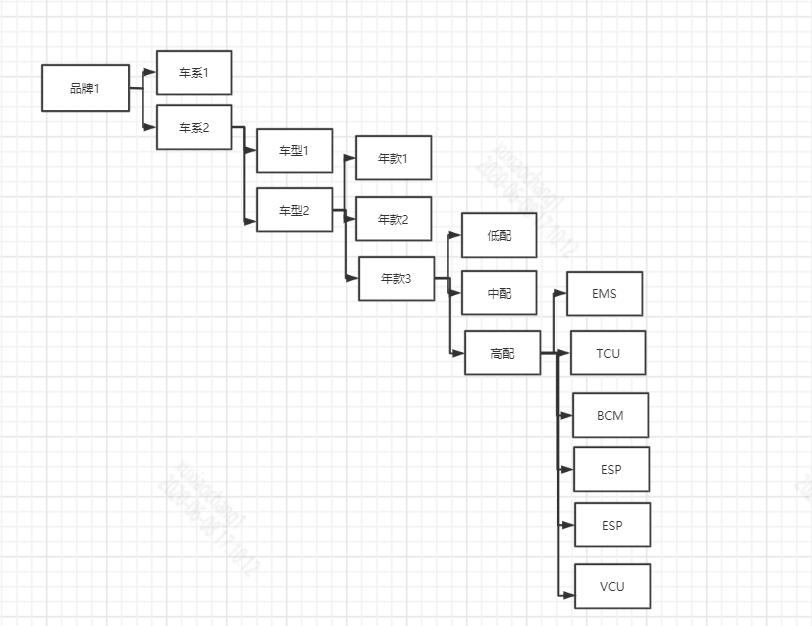


图5.1

#### 说明

设备树数据结构和记录由TSP维护。OTA云端系统中与TSP系统数据表结构保持一致，通过与TSP系统同步的方式来更新相关数据。OTA云端系统建设期间由自身维护。

## 固件升级对象服务

### 升级对象管理

#### 定义

升级对象：定位为允许进行OTA升级的对象(当前为汽车)，也可以将概念扩展迁移到其他的独立智能终端对象，提供更好的扩展性。

升级对象与升级对象清单简单示例如下：

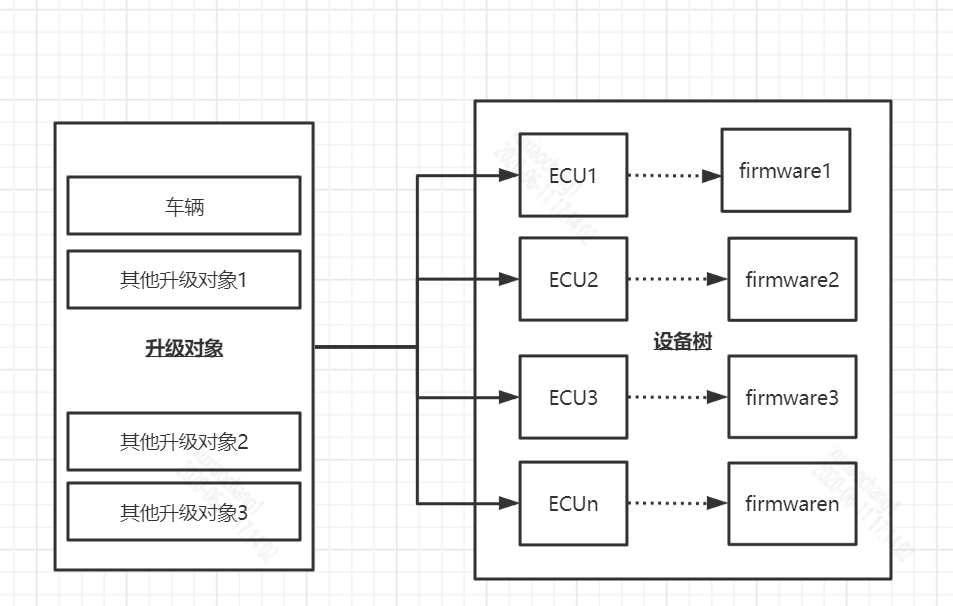


图5.2

#### 说明

升级对象管理当前主要用来管理具体的车辆记录

### 升级对象（车辆）清单管理

#### 定义

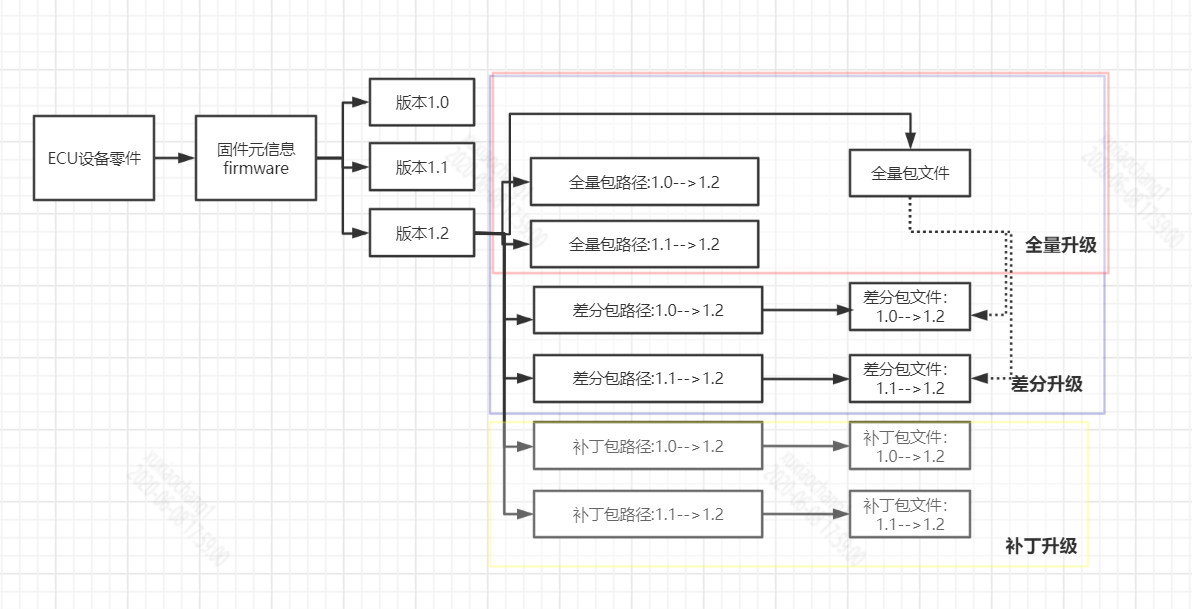
升级对象清单：升级对象（车辆）上所有待升级的固件清单列表。升级对象清单与对象及固件中间的关系参考图5.2

#### 流程说明

升级对象（车辆）通过设备树关联到所有ECU设备，同时根据设备与固件进行关联得到所有待申请的固件清单列表，该清单可以用于后续为对象制作升级计划。

## 固件元数据及版本管理

固件元数据、固件版本、升级路径、升级包关系如下：



说明：

1. 全量升级路径在新增版本时候默认初始化添加，根据适用版本确定升级路径范围
2. 差分升级路径在上传全量包制作差分升级包时添加
3. 补丁升级路径在上传补丁包制作时候添加，一次添加一条补丁路径

### 固件管理

#### 定义

固件：固件指设备ECU零件里搭载的软件，固件用来承载软件基础信息。

#### 流程说明

1. 人工初始化或接口方式导入固件信息， 该固件关联了具体的外部零件号，升级对象（车辆）通过设备树与固件关联来构建升级对象固件清单。
2. 将固件信息挂载到设备树，便于界面导航和立体可视化管理
3. OTA平台基于固件信息构建固件版本及升级包

### 固件版本管理

#### 定义

固件版本：软件版本，一个固件可以对应多个版本。

#### 流程说明

固件版本新增操作完成后，根据“适用版本”属性初始化全量升级路径数据记录集合（可能会有多条）。

1. 界面输入说明：

* 通过设备树导航定位到一个具体的设备零件对应的固件。
* 版本号：前期与OTA产商约定版本名称规则
* 依赖硬件版本号：该零件号对应的设备零件的硬件版本号。界面操作手动输入
* 适用版本：从该固件的历史版本列表中进行选择。该参数决定了到目标版本的所有升级路径，如果为空，则表示所有历史版本都可以升级到当前目标版本。
* 是否强制全量包升级：用于标志终端升级是否强制使用全量包。只针对固件升级模式为“差分升级”类型有效，强制适用全量包进行升级。
* 升级依赖：依赖的其他固件版本集合，需要区分依赖顺序要求。
* 发布说明：备注信息

1. 后台逻辑

校验

* 固件是否存在
* 固件目标版本是否存在
* 校验版本号数字码是否为最新版本序号
* 如果有固件依赖，则进行版本升级依赖校验。选择的依赖是否为与当前固件处于同一设备树下的父节点下的固件

数据逻辑处理：

* 新增固件版本表数据记录（固件版本记录数量n==1）
* 如果有固件依赖，新增记录到固件版本依赖表（依赖记录数量n>=0）
* 根据适用版本，初始化全量升级路径。如果有多个适应版本，则需要计算多条升级路径（表：tb\_software\_version\_path）。如果固件未设置适用版本信息，则默认为生成该固件的所有历史版本到该目标版本的升级路径。

### 固件升级包管理

#### 定义

**全量升级**：使用全量包进行OTA升级，兼容多个适用版本。

**补丁升级**：使用补丁包进行OTA升级，按需生成两个版本之间的补丁包。

**差分升级**：使用全量包作为OTA升级差分包制作来源，生成从起始版本开始到目标版 本的所有差分包。

#### 全量升级

全量升级操作是在固件版本新增完成后，通过补充全量包文件来完成。需要关联全量包文件数据记录Id到固件版本数据记录中去。

1、输入

* 客户端上传全量包，后台通过界面输入SHA码/MD5码及验证码类型验证该文件是否完整有效，存储该包至分布式文件系统，落地文件上传记录到DB，返回存储路径
* 客户端上传测试报告文件，后台存储，并关联文件下载地址到固件包文件
* 附加属性：预计刷写时长/供应商信息/发布说明

1. 后台逻辑

校验：

* 全量包文件/文件校验码/测试报告参数必传
* 固件数据记录中”升级包模式”属性是否支持全量包升级类型，不支持则提示升级类型异常
* 全量包文件sha码/MD5码校验是否正常，不一致提示文件异常或非法

数据逻辑处理：

* 全量包原始文件上传到MinIO分布式文件存储系统，新增文件上传记录（fota\_file\_upload\_records）。存储路径（下载路径）需要冗余到升级包文件表（fota\_software\_pkg）对应记录中
* 测试报告原始文件上传到MinIO分布式文件存储系统，新增文件上传记录。存储路径（下载路径）需要冗余到升级包文件表（tb\_fota\_software\_pkg）对应记录中。
* 新增固件包文件记录（表：tb\_fota\_software\_pkg）,该表中用于存储最终可对外分发的升级包文件信息。该记录需补充加密/加密算法/CDN分发地址等附加属性。同时，该全量包固件文件记录Id需要关联到固件版本表中对应的记录（tb\_fota\_software\_version）。

#### 差分升级

上传全量包文件，用于制作差分升级包。新增起始版本和目标版本之间的差分升级路径，并建立起差分路径与差分包之间的关联关系。

1、界面操作约定

* + 客户端上传全量包，输入SHA码/MD5码
  + 客户端上传测试报告文件，该测试报告关联对应全量包文件
  + 起始版本参数用于确定具体的查分升级包路径
  + 刷写脚本：用于差分包刷版本需要
  + 其他附属信息：预计刷写时长/供应商信息/发布说明

2、后台逻辑

校验：

* 全量包文件/文件校验码/测试报告/起始版本/差分脚本等参数必传
* 校验固件版本是否支持差分升级，如不支持，提示报错
* 全量包文件SHA码/MD5码校验，是否与前端传递校验码一致，不一致提示文件异常或非法
* 检验是否已经存在差分升级记录，判断条件为：存在至少一条差分包文件记录。如果有相关记录存在，不允许新增差分包操作。如果需编辑差分升级包信息，走先删除，再新增操作途径。

数据逻辑处理：

* + 原始包原始文件上传逻辑参考“全量升级”中原始包上传操作，保持一致。
  + 测试报告原始文件上传逻辑参考“全量升级”中测试报告上传操作，保持一致。
  + 新增固件包文件数据记录（表：tb\_fota\_software\_pkg）：包括全量包和多个差分包记录。每条数据记录需补充加密/加密算法/CDN分发地址等附加属性。全量包文件数据记录Id需要关联到固件版本数据记录中，差分包文件数据记录Id需要冗余到差分路径数据记录中。
  + 新增差分路径数据记录：（表：tb\_fota\_software\_version\_path）新增所有从起始版本到目标版本的差分路径记录集合，每条数据记录中需要关联件包文件数据记录ID。

#### 补丁升级

说明：上传补丁包文件，新增起始版本和目标版本之间的升级路径。补丁文件绑定到该条补丁升级路径。

1. 界面操作约定
   * 客户端上传补丁包，输入SHA码/MD5码，用于验证包的完整有效性
   * 客户端上传测试报告文件，该测试报告关联对应补丁包文件
   * 起始版本参数：用于确定具体的补丁升级包路径
   * 其他附属信息：预计刷写时长/供应商信息/发布说明
2. 后台逻辑

校验：

* + 补丁包文件/文件校验码/测试报告/起始版本等必要性参数验证
  + 检验数据中”升级包模式”属性是否支持补丁升级方式
  + 校验该补丁方式的升级路径是否存在，如果存在则放弃该次补丁升级包
  + 补丁包文件SHA码/MD5码校验，是否与前端传递校验码一致，不一致提示文件异常或非法

数据逻辑处理：

* + 补丁包原始文件上传到MinIO分布式文件存储系统，新增文件上传记录（tb\_fota\_file\_upload\_records）。存储路径（下载路径）需要冗余到升级包文件表（tb\_fota\_software\_pkg）对应记录中。
  + 测试报告原始文件上传到MinIO分布式文件存储系统，新增文件上传记录。存储路径（下载路径）需要冗余到升级包文件表（tb\_fota\_software\_pkg）对应记录中
  + 新增固件包文件记录（表：tb\_fota\_software\_pkg）,该表中用于存储补丁升级包文件信息，包括包文件大小、原始存储路径、加密/加密算法、CDN分发地址等。
  + 新增固件路径记录（表：tb\_fota\_software\_version\_path），此处只有一条数据记录，标识补丁升级包的起始路径到目标路径，并关联上个步骤中生成的pkg\_id属性。

## 任务调度管理

## 升级过程管理

## 升级任务管理

### 背景

制作升级任务的背景在于：

1. 对待升级OTA功能进行功能测试验证
2. 对待升级车辆进行灰度发布，合理控制升级风险
3. 通过时间/区域因素对升级车辆进行划分，分散后台服务升级压力，便于数据收集和分析

任务具体内容以下：

* 任务名称：简单描述当次升级目的/内容等信息，用于区分不同任务
* 策略选项：关联当前车型下配置的策略
* 车辆选项：通过品牌/车系/车型/年款/配置等属性，确认该任务适用的车型
* 任务模式：包括测试任务和正式任务，主要区别在于两者可选的车辆范围不同。
* 测试任务：升级车辆为系统指定测试车辆，可以通过标签属性或车辆白名单进行区分
* 正式任务：升级车辆为实际用户车辆
* 升级模式：包括静默下载\正常模式\工厂模式：
* 静默下载：升级包下载在TBOX端自动完成
* 正常模式：升级个阶段都需要人工交互参与
* 工厂模式：升级过程（版本检测\升级\安装）全自动化
* 任务开始时间：
* 任务结束时间：
* 版本说明：
* 免责声明

### 主要流程

新建任务：

1. 新建任务
   1. 添加任务基本信息：任务名称/有效日期/任务描述/免责申明
   2. 添加任务策略：从策略管理列表中选取，该策略主要定义了ECU固件升级方式和基本信息
   3. 添加升级车辆：根据车型基本信息/标签/地区进行车辆筛选，选中升级目标车辆。

说明：

* 标签数据需要从TSP设备管理系统同步
* 车辆地区信息需要从地理位置服务系统同步

1. 提交审核，任务进入到待审核阶段
2. 审核通过，任务进入到生效阶段；审核未通过，任务可以重新编辑，并再次提交审核。

更新和删除任务：未提交审核状态任务理论上都可以进行更新和删除

## 升级进度管理

### 总体情况

### 任务监控

## 策略管理

策略1

策略2

策略n

任务1

任务2

任务n

策略与任务关系（1：N）：

1. 策略和任务只能针对某款具体车型
2. 多个任务可能使用同一个策略
3. 单个任务只能关联单一的策略

### 定义

策略用于定义单次版本升级相关ECU信息及回滚依赖关系等说明。该策略一般来说需要明确该次升级的ECU清单及单个ECU执行的规范动作等，具体内容包括：

* 整车版本号：一系列新增和优化功能集的总和，整车版本号定义唯一，用于识别整车OTA软件能力当前所处的位置。
* 策略名称：可识别和简单描述该策略的主要内容
* 车辆选项：通过品牌/车系/车型/年款/配置等属性，确认该策略适用的车型
* 回滚策略：包括保守策略和激进策略
* 保守策略：任意升级ECU出现升级异常，则回滚当次升级涉及到的所有ECU
* 激进策略：区分有无依赖组。存在依赖组，则依赖组中ECU需要保证同升同降。无依赖组，则只关心当个ECU出现升级异常时，回滚自身ECU即可。
* 升级ECU列表：ECU升级目标列表。ECU目标列表需要指定升级排序号，用于确定升级顺序。升级ECU基本信息包括：
* 固件代码：用于标识别ECU升级所使用固件
* 固件名称：ECU固件名称
* 零件号：标识零件型号
* 零件名称
* 升级方式
* ~~前置版本~~
* 目标版本

### 新建策略

1. 选在品牌/车系/车型/年款/配置确定唯一车型信息
2. 添加策略基本信息
3. 添加策相关ECU到升级ECU列表

相关表：

策略表：

策略计划关联表：

### 策略审核

从细致安全的角度出发，需要对策略配置进行严格检查，通过接入审批流程，达到多人监督检查的目的。

策略审核相关角色：

1. 策略制定员角色
2. 初级策略审核员（可能存在多个）角色
3. 终级策略审核员：最终决定策略审核通过

主要流程：

1. 策略制定员创建策略，提交审核
2. 初级策略审核员审核策略，初级审核通过标准为所有初级策略审核员全部审核通过。
3. 终级策略审核员审核策略

## 升级日志管理

提供给OTA云端管理系统查询和管理车辆升级日志记录信息。OTA云端管理系统运营人员可以通过下载日志文件查看升级过程详情信息。

TBOX

OTA文件服务

OTA云端

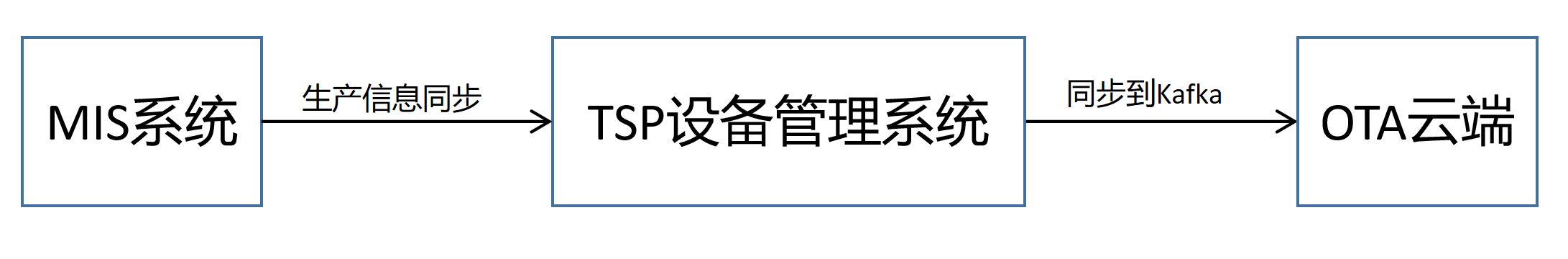
OTA(Web)管理

1. 日志上传：TOBX升级主控将升级过程中相关日志通过http接口上传到OTA文件服务
2. OTA文件服务保存日志文件后，将相关业务信息通过分布式消息队列(Kafka)同步到OTA云端
3. OTA云端将对应车辆和日志文件记录进行关联
4. OTA后台管理平台提供OTA日志查询和下载功能

## 升级对象（车辆）信息同步

背景：OTA云端需要管理升级对象及升级对象/零件与OTA固件之间的关联关系。而升级对象/零件来源于生产系统。

升级对象（车辆）信息同步示意图如下：



说明：

1. 车厂MIS系统主要同步车辆信息到设备管理系统，同步方式由设备管理系统与MIS系统对接商定。
2. 设备管理系统主动同步车辆信息到OTA云端，设备管理系统应提供回调应答机制确保信息成功同步。同时，设备管理系统应提供同一份数据再次同步的能力。
3. 同步方式可考虑两种方式：
   1. 分布式消息队列通知同步(建议使用)
   2. 接口同步

### 设备树同步

#### 定义

设备树当前管理层级从上到下依次为：品牌/车型/车系/年款/配置。在OTA云端管理系统中设备树的管理有两种方式：

1. 通过OTA后台云端管理系统维护设备树
2. 从车厂MIS系统同步上述数据到OTA云端系统，云端系统动态维护该设备树结构数据。

某一车型零部件列表数据同步所需字段定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| brand | String | Y | 品牌 |
| brandCode | String | Y | 品牌编码 |
| series | String | Y | 车系 |
| seriesCode | String | Y | 车系编码 |
| model | String | Y | 车型 |
| modelCode | String | Y | 车型编码 |
| year | String | Y | 年款 |
| yearCode | String | Y | 年款编码 |
| conf | String | Y | 配置 |
| confCode | String | Y | 配置编码 |
| componentName | String | Y | 零件名称 |
| componentCode | String | Y | 零件代码 |
| componentModel | String | Y | 零件型号 |
| diagnoseId | String | Y | 诊断Id |

#### 流程说明

### 车辆信息同步

#### 定义

OTA云端需要管理升级车辆对象元数据信息，该信息来源于车厂MIS系统数据同步。车厂MIS系统一般采取批量同步的方式同步同一车型的相关信息。车辆属性信息包括：品牌、车系、车型、年款、配置、车辆Vin码、整车版本、当前区域等

车辆零件信息包括：零件代码、零件型号、固件出厂版本号。车辆零件信息属性字段说明：

* 零件代码：区分具体零件
* 零件型号：区分具体零件的某一型号
* 固件出厂版本号：定义软件版本

前提说明：

同一车型在OTA云端管理系统可管理的固件清单应该提前入库完成，以保证该款车型可纳入到固件升级流程中去

某一车型车辆同步所需字段定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| brand | String | N | 品牌 |
| brandCode | String | Y | 品牌编码 |
| series | String | N | 车系 |
| seriesCode | String | Y | 车系编码 |
| model | String | N | 车型 |
| modelCode | String | Y | 车型编码 |
| year | String | N | 年款 |
| yearCode | String | Y | 年款编码 |
| conf | String | N | 配置 |
| confCode | String | Y | 配置编码 |
| vin | String | Y | 车辆Vin码 |
| entireVersion | String | Y | 整车版本号 |
| List<ComponentInfo> | componentInfos | Y | 车辆零件列表 |

零件(ComponentInfo)信息定义：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| componentName | String | N | 零件名称 |
| componentCode | String | Y | 零件代码 |
| componentModel | String | Y | 零件型号 |
| firmwareVersionNo | String | Y | 软件（固件）版本号 |

#### 流程

车辆信息同步流程：

1. OTA云端管理系统设备树数据提前维护好基于品牌\车系\车型\年款\配置属性信息的层级关系
2. TSP设备管理系统将车辆及车辆零件列表同步到OTA云端（通过Kakfa）。根据下线车辆真实属性信息：品牌、车系、车型、年款、配置，定位关联到设备树的最底层叶子节点。即保证车辆能唯一关联上某一车型的在设备树上的配置节点

车辆零件信息同步：

1. 保证车辆信息已经同步到OTA云端管理系统
2. 保证该车辆所属车型零件清单数据已提前入库完成
3. 根据真实车辆零件代码/型号匹配OTA云端管理系统零件清单入库。OTA云端系统更多关注车辆零件信息对应的可升级固件清单。

### 车辆标签信息同步

#### 定义

升级任务创建流程中需要通过标签信息筛选车辆，所以需要从TSP设备管理系统中同步信息到OTA云端

#### 流程

1. TSP设备管理系统中存在标签添加/更新等事件产生时，通过kafka消息队列同步消息到OTA云端
2. OTA云端将车辆标签数据入库，用于后续车辆筛选查询

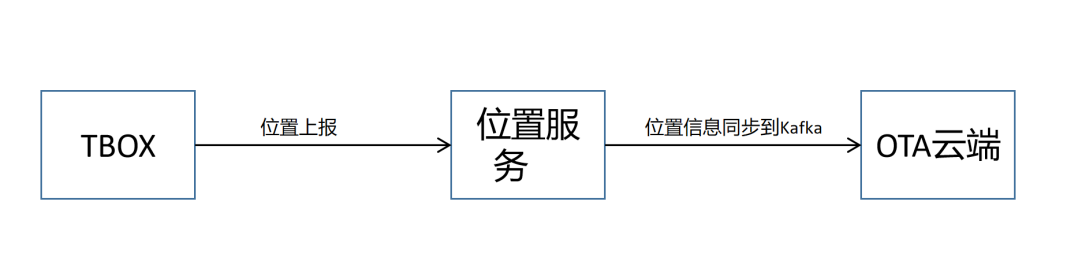
### 地理位置信息同步

#### 定义

新建任务时需要根据车辆当前所在区域数据进行车辆筛选，OTA云端需要同步维护车辆当前位置信息。车辆位置信息来源于车辆位置服务，当位置信息数据存在更新时同步到OTA云端后台。

#### 流程

系统交互流程如下图：



1. 车辆实时上报位置信息
2. 车辆位置服务解析经纬度信息，解析成区域信息
3. 位置信息通过分布式消息队列（Kakfa）同步到OTA云端

# 接口设计

## 设备树管理 (OTA管理)

### 设备树接口

#### 说明

提供给前端构建树状结构的所有设备零件立体层级数据，JSON格式，用于设备零件的定位和导航。

具体操作步骤如下：

1）从db查询所有设备数据数据

2）根据层级和深度构建树状结构，如第一级为品牌，子级为车系，则某一品牌的所有数据

#### 接口url

POST {HOST}/fota/v1/deviceTree/listAll

#### 5.1.3 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| projectId | string | Y | 租户Id，用于不同的设备树 |

#### 接口请求示例

{

“projectId”:”0001”

}

#### 接口出参

树结点（DeviceTreeNode）：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| id | long | Y | 树节点Id |
| nodeName | string | Y | 树节点名称 |
| nodeCode | string | Y | 树节点编码 |
| parentId | long | Y | 父节点Id |
| rootId | long | Y | 根节点Id |
| nodeIdPath | string | Y | 以节点ID作为路径元素 |
| nodeNamePath | string | Y | 以节点NAME作为路径元素 |
| nodeCodePath | string | Y | 以节点CODE作为路径元素 |
| status | int | N | 启用状态:0=正常，1=删除 |
| orderNum | int | N | 排序号 |
| childrenDeviceTreeNodes | List<DeviceTreeNode> | N | 子节点记录列表 |

#### 响应示例

{

“respCode”:200,

“respMsg”:”success”

“respData”:{

}

}

### 获取设备树下级叶子节点列表

#### 说明

选定一个子节点，获取下级节点列表。

具体操作步骤如下：

1）前端选择一个节点，传递该节点ID

2）后台从DB设备数据记录表中查询所有parentId为前端传递参数ID的所有节点数据

#### 接口url

POST {HOST}/fota/v1/deviceTree/listChildren

#### 5.1.3 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| id | long | Y | 节点Id |

#### 接口请求示例

{

“id”:”0001”

}

#### 接口出参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| deviceTreeNodeList | List<DeviceTreeNode> | N | 下级节点列表 |

DeviceTreeNode定义参考6.1.5节所述

#### 响应示例

{

“respCode”:200,

“respMsg”:”success”

“respData”:{

}

}

## 固件管理 (OTA管理)

### 添加固件接口

#### 说明

通过选定车型配置，添加固件信息，该固件用于标识设备零件的软件信息。

固件基本信息包括：名称、代码、零件号、零件名称、诊断Id、刷写脚本、固件说明。

附属信息包括：品牌、车系、车型、年款、配置等，用于页面显示

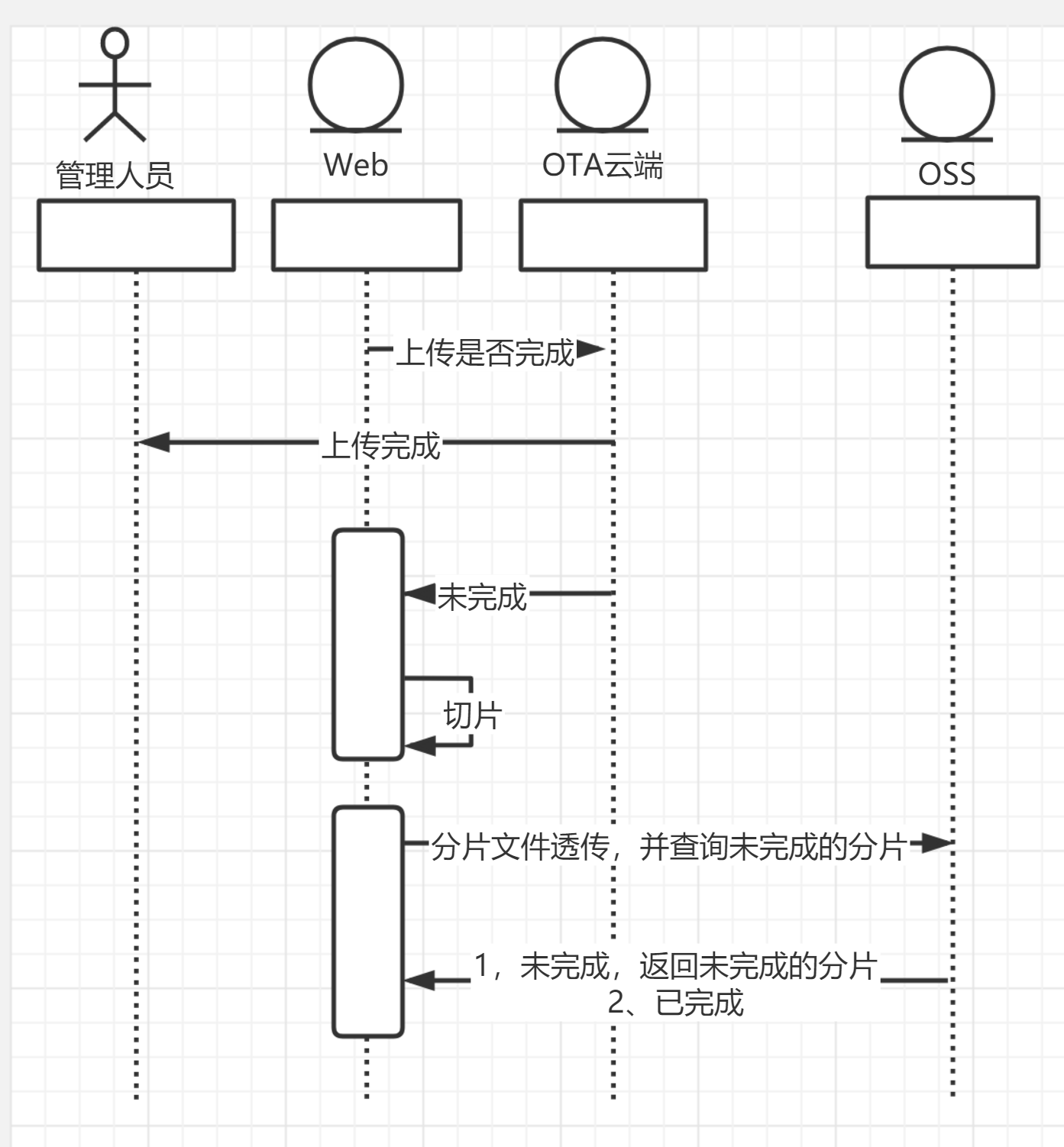
刷写脚本：从配置表中选择对应脚本设置项

操作步骤如下：

1. 固件基本信息参数检查，检查不通过提示参数异常
2. 从数据库中检索该固件记录是否存在，存在则退出新增操作。固件记录唯一性通过固件代码标识。
3. 升级包模式，有三种选择方式：全量/补丁/差分，该属性用于限制后续制作升级包的方式
4. 落地固件数据记录。

在OTA升级固件中，针对IHu设备往往体积庞大，OTA云端需要实现分片续传的功能。

大文件分片续传方案示意图如下：



#### 接口url

POST {HOST}/fota/v1/firmware/add

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| firmwareName | string | Y | 固件名称 |
| firmwareCode | string | Y | 固件代码 |
| componentCode | string | Y | 零件代码 |
| componentModel | string | Y | 零件型号 |
| componentName | string | Y | 零件名称 |
| diagnose | stirng | Y | 诊断ID |
| flashScriptUrl | string | Y | 刷写脚本 |
| description | string | Y | 描述 |
| packageModel | string | Y | 升级包模式 |
| apnChannel | string | Y | 升级渠道 |
| createBy | String | Y | 操作用户Id |

#### 接口请求示例

{

“firmwareName”:”车身控制器软件”,

“firmwareCode”:”120DAKHK4876”,

“componentCode”:”120DAKHK4876”,

“componentModel”:”YDD”,

“componentName”:”车身控制器”,

“diagnose”:”D321321”,

“flashScriptUrl”:”http://file.bnicvc.com/firmware/shell/flushFile.sh”,

“description”:”这是车身控制器固件的一个说明”,

“packageModel”:”1”,

“apnChannel”:”tbox”,

“createBy”:”administrator”

}

#### 接口出参

通用相应码

#### 响应示例

{

“respCode”:200,

“respMsg”:”success”,

“respData”:{

}

}

### 新建固件版本接口

#### 说明

固件版本定义最新功能的变化，通过计算不同固件版本来计算升级路径，最终适配客户端的固件包升级。

1. 参数校验：固件代码/固件版本号/依赖硬件版本号参数校验失败则退出操作
2. 查询DB中是否存在该固件该版本记录，如果存在，则放弃该次新增操作；如果有版本依赖列表，检查版本依赖记录是否存在
3. 持久化该条固件版本记录到DB。
4. 判断是否需要生成全量升级包路径记录集合持久化到DB：
   1. 固件信息记录表中对应“升级包模式”为全量/差分类型，需要初始化全量升级包路径记录
   2. 固件信息记录表中对应“升级包模式”为补丁类型，不需要初始化全量升级包路径记录
5. 如果有版本依赖列表，绑定依赖版本记录到该固件版本依赖表

该接口中DB操作需保证一致性事务

#### 接口url

POST {HOST}/fota/v1/firmware/addSoftwareVersion

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| firmwareId | string | Y | 固件Id |
| firmwareVersionNo | string | Y | 固件版本号 |
| appliedSoftwareVersion | string | Y | 适用的历史版本(分隔符是分号) |
| appliedHardwareVersion | string | Y | 适用的硬件版本集合(分隔符是分号) |
| isForceFullUpdate | int | N | 是否强制全量包升级 |
| dependences | List<Object> | N | 依赖的其他固件版本集合 |
| releaseNotes | string | N | 发布说明 |
| createBy | string | N | 操作人Id |

#### 接口请求示例

{

“firmwareId”:”123456”,

“firmwareVersionNo”:”v1.2”,

“appliedSoftwareVersion”:”v1.0;v1.1”,

“appliedHardwareVersion”:”vb1.00”,

“isForceFullUpdate”:”1”,

“dependences”:,

“flashScriptUrl”:”http://file.bnicvc.com/firmware/shell/flushFile.sh”,

“releaseNotes”:”这是车身控制器固件的一个说明”,

“createBy”:”administrator”

}

#### 接口出参

通用参数

#### 响应示例

{

“respCode”:200,

“respMsg”:”success”

“respData”:{

}

}

## 固件版本管理 (OTA管理)

### 固件版本列表接口

#### 说明

通过固件ID从表tb\_fota\_firmware\_version表中索引所有firmware\_id字段值为给定固件Id的数据记录列表。

#### 接口url

POST {HOST}/fota/v1/firmware/listSoftwareVersions

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| firmwareId | string | Y | 固件ID |

#### 接口请求示例

{

“firmwareId”:”0001”

}

#### 接口出参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| firmwareVersionNo | string | Y | 固件版本号 |
| createDate | string | Y | 发布日期 |
| releaseNotes | string | Y | 发布说明 |

#### 响应示例

{

“respCode”:200,

“respMsg”:”success”

“respData”:{

{

“firmwareVersionNo”:”123001”,

“createDate”:”发布日期1”,

“releaseNotes”:”发布说明1”,

},

{

“firmwareVersionNo”:”123002”,

“createDate”:”发布日期2”,

“releaseNotes”:”发布说明2”,

}

}

}

## 固件升级包管理 (OTA管理)

### 升级包列表接口

#### 说明

给定一个固件版本，获取该版本下所有升级包列表。该升级包列表以升级路径记录为基准进行查询，相关显示字段需要要聚合操作。

具体操作：

1、查找所有目标固件版本ID为选定的版本参数的版本路径记录。如果为差分/补丁升级路径，直接关联升级包文件路径信息；如果为全量包升级另，需要先关联到该固件版本，再关联到升级包文件路径信息。

#### 接口url

POST {HOST}/fota/v1/firmware/listSoftwareVersionPaths

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| firmwareVersionId | string | Y | 固件版本Id |
| firmwareVersionNo | string | Y | 固件版本号 |
| createBy | string | Y | 操作用户Id |

#### 接口请求示例

{

“firmwareVersionId”:”123456”，

“firmwareVersionNo”:”0001”,

“createBy”:”xxc”

}

#### 接口出参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| upgradePathType | string | Y | 升级包类型 |
| releasePkgName | string | Y | 升级包名称 |
| releasePkgFileName | string | Y | 升级包文件名称 |
| releasePkgFileSize | string | Y | 包文件大小 |
| startFirmwareVerNo | string | Y | 起始版本 |
| targetFirmwareVerNo | string | Y | 目标版本 |
| createDate | date | Y | 上传日期 |

#### 响应示例

{

“respCode”:200,

“respMsg”:”success”

“respData”:{

“upgradePathType”:”1”,

“releasePkgName”:”releaseABC”,

“releasePkgFileName”:”10010”,

“releasePkgFileSize”:”4000K”,

“startFirmwareVerNo”:”V1.0”,

“targetFirmwareVerNo”:”V1.2”,

“createDate”:”2020-06-9 12:00:05”

}

}

### 添加全量升级包接口

#### 说明

1. 校验相关必要性参数是否完整
2. 检验固件版本“升级模式”属性是否为“全量升级模式”
3. 提前上传全量升级包，对文件进行加密/标准化操作
4. 添加升级包文件数据记录（tb\_fota\_firmware\_pkg），并记录升级包原始下载路径，同时需要需关联该数据记录Id到表（tb\_fota\_firmware）

#### 接口url

POST {HOST}/fota/v1/firmware/addWholePkg

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| firmwareVersionId | long | Y | 固件Id |
| firmwareVersonNo | string | Y | 固件版本号 |
| originalPkgFilePath | string | Y | 全量包文件Url地址 |
| verifyCode | string | Y | 验证码 |
| verifyType | int | Y | 验证码类型 |
| originalReportFilePath | string | Y | 测试报告文件url地址 |
| estimateTime | int | Y | 预计刷写时长 |
| supplierInfo | string | Y | 供应商信息 |
| releaseDescription | string | n | 发布信息 |
| createBy | string | Y | 操作用户Id |

#### 接口请求示例

{

“firmwareVersionId”:”0001”,

“firmwareVersonNo”:”003023”,

“originalPkgFilePath”:obj1,

“verifyCode”:”sha1”,

“verifyType”:”1”,

“originalReportFilePath”:obj2,

“estimateTime”:”30”,

“supplierInfo”:”supplierABC”,

“releaseDescription”:”descABC”,

“createBy”:”xxc”

}

#### 接口出参

通用响应参数

#### 响应示例

{

“respCode”:200,

“respMsg”:”success”

“respData”:{

}

}

### 添加差分升级包接口

#### 说明

参考全量升级模式，需要完成全量升级包所有业务逻辑。

不同点:

1. 根据起始版本计算差分路径：有多少条差分路径，需要根据差分算法生成多少个目标差分包。
2. 存储差分包文件记录（tb\_fota\_firmware\_pkg），关联该数据记录Id到升级路径表（tb\_fota\_firmware\_version\_path）

#### 接口url

POST {HOST}/fota/v1/firmware/addDiffPkg

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| firmwareVersionId | long | Y | 固件Id |
| firmwareVersonNo | string | Y | 固件版本号 |
| originalPkgFilePath | string | Y | 全量包文件url地址 |
| verifyCode | string | Y | 验证码 |
| verifyType | int | Y | 验证码类型 |
| originalReportFilePath | string | Y | 测试报告文件url地址 |
| difScriptUrl | string | Y | 差分脚本url |
| difFirmwareVersonNo | string | Y | 起始版本号 |
| estimateTime | int | Y | 预计刷写时长，单位为秒 |
| supplierInfo | string | Y | 供应商信息 |
| releaseDescription | string | Y | 发布信息 |
| createBy | string | Y | 操作用户Id |

#### 接口请求示例

{

“firmwareVersionId”:”0001”,

“firmwareVersonNo”:”003023”,

“originalPkgFilePath”:”http://file.bnicvc.com/file/allPkg.apk”,

“verifyCode”:”sha1”,

“verifyType”:”1”,

“originalReportFilePath”:”http://file.bnicvc.com/file/report.docx”,

“difScriptUrl”:”http://file.bnicvc.com/file/difScript.sh”,

“difFirmwareVersonNo”:”V1.0”,

“estimateTime”:”30”,

“supplierInfo”:”supplierABC”,

“releaseDescription”:”descABC”,

“createBy”:”xxc”

}

#### 接口出参

#### 响应示例

{

“respCode”:200,

“respMsg”:”success”

“respBody”:{

}

}

### 添加补丁升级包接口

#### 说明

参考5.3.3.4结点说明

#### 接口url

POST {HOST}/fota/v1/firmware/addPatchPkg

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| firmwareVersionId | long | Y | 固件Id |
| firmwareVersonNo | string | Y | 固件版本号 |
| patchFile | string | Y | 补丁文件url地址 |
| verifyCode | string | Y | 验证码 |
| verifyType | int | Y | 验证码类型 |
| startVersion | string | Y | 起始版本 |
| reportFile | string | Y | 测试报告文件url地址 |
| estimateTime | int | Y | 预计刷写时长 |
| supplierInfo | string | Y | 供应商信息 |
| releaseDescription | string | N | 发布信息 |
| createBy | string | Y | 操作用户Id |

#### 接口请求示例

{

“firmwareVersionId”:”0001”,

“firmwareVersonNo”:”003023”,

“pkgFile”:”http://file.bnicvc.com/file/difPkg.apk”,

“verifyCode”:”sha1”,

“verifyType”:”1”,

“reportFile”:”http://file.bnicvc.com/file/report.docx”,

“estimateTime”:”30”,

“supplierInfo”:”supplierABC”,

“releaseDescription”:”descABC”

}

#### 接口出参

通用响应参数

#### 响应示例

{

“respCode”:200,

“respMsg”:”success”

“respBody”:{

}

}

## 升级任务管理 (OTA管理)

### 新建任务接口

#### 说明

升级管理的任务管理的新建任务接口，提供新建任务功能,用户可以自定义任务的相关属性和策略，时间，固件版本，以及升级对象的范围.

#### 接口url

POST {HOST}/fota/v1/task

#### 接口请求示例

{

“name”:”18027888111”,

“StartTime”:”xxx”,

“finishTime”:”xxx”,

.......

}

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是  否必填 | 说明 |
| upgradeModeId | Int | Y | 升级模式ID |
| name | string | Y | 任务名称 |
| startTime | datatime | Y | 开始时间 |
| finishTime | datatime | Y | 结束时间 |
| maxDownloadFailed | int | Y | 最大下载失败次数 |
| maxVerifyFailed | int | Y | 最大验证失败次数 |
| maxInstallFailed | int | Y | 最大安装失败次数 |
| newVersionTips | string | Y | 新版本说明 |
| downloadTips | String | Y | 下载提示语 |
| disclaimer | string | Y | 免责声明 |
| taskTips | string | Y | 任务说明 |

#### 响应示例

{

“respCode”:200,

“respMsg”:”success”

“data”:{

“taskId”:”123456”

}

### 删除任务接口

#### 说明

删除任务接口，用户可以将未激活的任务的数据进行删除

#### 接口url

DELETE {HOST}/fota/v1/task/{taskId}

#### 接口请求示例

{HOST}/fotacenter/v1/task/123

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是  否必填 | 说明 |
| taskId | Int | Y | 任务ID |

#### 响应示例

{

“respCode”:200,

“respMsg”:”success”

“data”:{

“status”:”true”

}

}

}

### 启用任务接口

#### 说明

升级管理的任务管理的启用任务接口，提供新建任务功能,用户可以自定义任务的启用和停用.

#### 接口url

PUT {HOST}/fota/v1/task

#### 接口请求示例

{

id:”123”,

status:”0”

}

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是  否必填 | 说明 |
| taskId | Int | Y | 升级ID |
| taskStatus | int | Y | 任务状态 |

#### 响应示例

{

“respCode”:200,

“respMsg”:”success”

“data”:{

“status”:”true”

}

### 任务列表接口

#### 说明

升级管理的任务管理的任务列表接口，提供任务管理的查询功能，用户可以通过条件索引到相关任务.

#### 接口url

GET {HOST}/fota/v1/task

#### 接口请求示例

{

page{

}

objectId:”123”

}

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是  否必填 | 说明 |
| page | string | N | 分页相关参数 |
| objectId | int | Y | 升级车型ID |

#### 响应示例

{

“respCode”:200,

“respMsg”:”success”

“data”:{

task1{

name: “123”,

startTime:”2020-02-22”,

finshTime:”2020-02-23”,

objectList:{

.....

},

firmwareList:{

......

},

taskStatus: “1”,

createTime:”2020-02-20”,

del\_flag:”0”,

downloadTips:”下载说明....”

},

task2{

}

}

### 新增任务固件清单接口

#### 说明

升级任务管理的新增升级任务固件清单，一次任务配置一个用户选定的固件清单，并为固件清单配置相关策略。

#### 接口url

POST {HOST}/fota/v1/upgradeFirmwareList

#### 接口请求示例

{

upgradeFirmware{

taskId:”1”,

firmwareId:”1”

rollbackMode:”1”,

targetVersionId:”1”,

upgrade\_seq:”1”

},

upgradeFirmware{

taskId:”1”,

firmwareId:”2”,

...

}

}

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是  否必填 | 说明 |
| List<upgradeFirmware> | List | N | 升级任务固件清单 |
| taskId | int | Y | 升级ID |

#### 响应示例

{

“respCode”:200,

“respMsg”:”success”

“data”:{

Status:”true”

}

### 新增升级对象车型清单接口

#### 说明

升级任务管理的新增升级任务对象车型清单，一次任务配置一个用户选定的升级对象车辆范围，并为升级车型清单配置相关策略。

#### 接口url

POST {HOST}/fota/v1/upgradeObjectList

#### 接口请求示例

{

upgradeObject{

taskId:”1”,

ObjectId:”1”,

vin:”vin1231qwe”

vehicleId:”1123”

},

upgradeObject{

taskId:”1”,

ObjectId:”2”,

vin:”vin1231qwe1”

vehicleId:”11231”

}

}

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是  否必填 | 说明 |
| List<upgradeObject> | List | N | 升级任务固件清单 |
| taskId | int | Y | 升级ID |

#### 响应示例

{

“respCode”:200,

“respMsg”:”success”

“data”:{

Status:”true”

}

### 查询任务详情接口

#### 说明

升级任务管理的任务详情，用户可以查看该任务的详细配置。

#### 接口url

GET {HOST}/fota/v1/task/{taskId}

#### 接口请求示例

{

taskId:”1”

}

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是  否必填 | 说明 |
| taskId | int | Y | 升级ID |

#### 响应示例

{

“respCode”:200,

“respMsg”:”success”

“data”:{

task1{

name: “123”,

startTime:”2020-02-22”,

finshTime:”2020-02-23”,

objectList:{

.....

},

firmwareList:{

......

},

taskStatus: “1”,

createTime:”2020-02-20”,

del\_flag:”0”,

downloadTips:”下载说明....”

}

}

## 升级任务调度

OTA升级通知下发到手机App,

OTA升级通知下发到终端 （终端转发给车机）

## 面向TBox的消息上行

TBOX端通过MQTT协议，通过汇聚平台通道（Kafka消息队列）将业务消息上行到OTA云端。

发起端：TBOX

中间层：汇聚平台

接收端：OTA云端

### 消息格式通用说明

#### 通用请求消息格式（上行消息）

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 请求头-序列号（Tbox端用于消息应答） |
| timestamp | Long | 是 | 请求头-时间戳 |
| vin | String | 是 | 请求头-车辆标识（一般为vin码） |
| businessType | Integer | 是 | 请求头-消息类型：  1=新版本检查请求  2=新版本下载请求  3=新版本下载确认结果  4=下载进度汇报请求  5=新版本安装确认请求  6=新版本安装确认结果  7=安装进度汇报请求  8=安装结果汇报请求  9=车端ECU信息上报  10=获取服务器可升级固件清单列表  51=设备注册请求/确认（保留） |
| transId | Long | 否 | 升级事务Id（服务端第一次版本检查结果确认时生成），用于关联整个升级流程。当businessType不为100（升级通知主动下发）时，不能为空 |
| taskId | Long | 否 | 升级任务Id |
| reqId | Long | 否 | 云端定义，用于与TBOX进行消息应该确认 |
| sourceType | Integer | 否 | 指令来源：  1=来自Hu  2=来自TBOX  3=来自APP  4=来自OTA云端 |
| errorCode | Integer | 否 | 消息状态码 |
| errorMsg | String | 否 | 消息错误信息 |
| **body** | Object | 否 | 请求体-（由具体协议实现，该属性是否为空由businessType决定） |

说明：

1. businessType定义了TBox终端上行的消息类型，**body**定义了请求消息体对象，具体类型由TBox协议端实现。
2. 以下具体小节中格式说明为对应的**body**消息格式

#### **通用响应消息格式（下行消息）**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 响应头-序列号（来自于TBox端生成） |
| timestamp | Long | 是 | 响应头-时间戳 |
| vin | String | 是 | 响应头-车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | 响应头-消息类型：  101=新版本检查结果响应  102=新版本升级确认响应  103=新版本安装确认响应  104=新版本通知（OTA云端发布新任务后主动下发）  105=App客户端请求TOBX进行新版本检查请求（APP客户端发起，OTA云端下行到TBOX）  106=OTA云端向TBOX下发服可升级固件清单列表  107=OTA云端向TBOX下发取消预约安装 |
| transId | Long | 否 | 升级事务Id（服务端第一次版本检查结果确认时生成），用于关联整个升级流程。当businessType不为100（升级通知主动下发）时，不能为空 |
| taskId | Long | 否 | 升级任务Id |
| reqId | Long | 否 | 云端定义，用于与TBOX进行消息应该确认 |
| sourceType | Integer | 否 | 指令来源：  1=来自Hu  2=来自TBOX  3=来自APP  4=来自OTA云端 |
| errorCode | Integer | 否 | 消息状态码 |
| errorMsg | String | 否 | 消息错误信息 |
| **body** | Object | 否 | 请求体-（由具体协议实现，该属性是否为空由businessType决定） |

说明：

1. businessType定义了TBox终端下发的消息类型，**body**定义了请求消息体对象，具体类型由

TBox协议端实现。

1. 以下具体小节中格式说明为对应的**body**消息格式

#### OTA云端定义的车辆所处升级阶段状态表

|  |  |
| --- | --- |
| 状态值 | 说明 |
| -1 | 无新版本：无有效的升级计划 |
| -2 | 无新版本：存在有效的升级计划，TBOX客户端检测不存在需要升级的OTA固件版本 |
| 0 | 存在新版本：TBOX还未请求进行新版检测。 |
| 1 | 存在新版本：TBOX请求进行新版本检测且已经匹配到存在新版本，APP端需要异步等待TBOX完成整个流程后的消息通知。 |
| 11 | 存在新版本：待下载。TBOX本地已收到新版本检测结果,并已经同步该状态到云端。 |
| 12 | 存在新版本：开始下载。TBOX已收到下载指令,开始下载（并已与OTA云端确认状态同步） |
| 13 | 存在新版本：升级下载中 |
| 14 | 存在新版本：升级下载终止（因某些原因导致下载终止） |
| 15 | 存在新版本：升级下载中等待（因某些原因导致下载等待） |
| 16 | 存在新版本：TBOX执行下载完成（中间状态） |
| 17 | 存在新版本：TBOX执行下载成功，安装包校验成功 |
| 18 | 存在新版本：TBOX执行下载失败，安装包校验失败 |
| 31 | 存在新版本：待升级安装。待客户端安装确认(TBOX下载已经完成之后) |
| 32 | 存在新版本：开始升级。TBOX已收到升级安装指令，开始安装(并已与OTA云端确认同步) |
| 33 | 存在新版本：安装前置条件检查失败 |
| 34 | 存在新版本：触发升级（HU端触发） |
| 35 | 存在新版本：取消升级（HU端触发操作） |
| 36 | 存在新版本：升级安装中 |
| 37 | 存在新版本：升级完成（所有ECU升级成功） |
| 38 | 存在新版本：升级未完成（待升级ECU全部没有升级至目标版本） |
| 39 | 存在新版本：升级失败（升级过程完成，待升级ECU出现回滚失败） |
| 111 | 存在新版本：未获取任务终止 |
| 112 | 存在新版本：未下载终止 |
| 113 | 存在新版本：未升级终止 |

表6.7.1-1：OTA云端定义的车辆所处升级阶段状态表

### TBOX设备注册请求/确认发送（来源于TBox：保留）

#### 说明

TBOX升级主程序第一次启动时，需要启动与OTA云端注册，标识设备升级身份。TBOX升级程序会生成一个设备Id，绑定到车辆VIN码，后续交互流程以设备Id为车辆升级身份进行。

消息类型包括：

1. TBOX设备注册请求
2. TBOX设备注册确认

#### 接口url

#### 接口请求示例

{

"businessId": 1,

"businessType": 51,

"timestamp": 123456789,

"vin": "bq001",

"body": {

"transId": 1000086

}

}

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 请求头-序列号（由TBox端生成） |
| timestamp | Long | 是 | 请求头-时间戳 |
| vin | String | 是 | 请求头-车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | businessType=51 |
| body | Object | 是 | 请求消息体 |

* body定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| deviceId | String | 是 | TBOX端生成的设备身份Id |
| registerType | Integer | 是 | 1=注册，2=注册ACK |

#### 响应示例

无

### 获取服务端配置

#### 说明

TBOX通过请求OTA云平台，获取某一配置车型下的所有可升级ECU固件清单。该清单用于指导TBOX升级程序与具体的ECU硬件进行升级通信。

#### 接口url

~~POST {HOST}/fota/v1/tbox/{vin}/get/server/config~~

MQTT通道

#### 接口请求示例

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 请求头-序列号（同一个Tbox端需要唯一，可加日期区分） |
| timestamp | Long | 是 | 请求头-时间戳 |
| vin | String | 是 | 请求头-车辆标识（一般为vin码） |
| businessType | Integer | 是 | businessType=10 |
| body | Object | 是 | 暂无 |

#### 响应示例

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 请求头-序列号（同一个Tbox端需要唯一，可加日期区分） |
| timestamp | Long | 是 | 请求头-时间戳 |
| vin | String | 是 | 请求头-车辆标识（一般为vin码） |
| businessType | Integer | 是 | businessType=106 |
| body | Object | 是 | 请求消息体，定义如下 |
| errorCode | Integer | 否 | 异常码 |
| errorMsg | String | 否 | 异常信息 |

body数据定义：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| confVersion | String | 是 | ECU固件配置版本 |
| ecus | List<EcuConfig> | 是 | ECU可升级零件列表 |

EcuConfig配置信息定义:

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| ecuId | String | 是 | 零件代码 |
| ecuName | String | 是 | 零件名称 |
| ecuDid | String | 是 | 诊断Id |
| ecuSwid | String | 是 | 软件Id |
| ecuResponseId | String | 否 | 响应Id |
| ecuFunctionId | String | 否 | 功能Id |
| installType | String | 否 | 安装类型（通过管理平台配置，下发的值为配置的**字典索引**） |
| infoCollCtrlObj | String | 是 | 信息采集受控对象：指该升级对象/单元由谁负责采集其基础、更新信息 |
| dlCtrlObj | String | 是 | 下载受控对象：指该升级对象/单元的软件包由哪个零件进行下载 |
| instTxCtrlObj | String | 是 | 安装传输受控对象：指该升级对象/单元的软件包由哪个零件进行传输 |
| instCtrlObj | String | 是 | 安装受控对象：指该升级对象/单元的软件包由哪个零件负责安装 |
| instCondition | String | 是 | 安装约束：指该零件需要处于低压或高压条件下才可以安装 |
| busType | String | 是 | 总线类型：指主控端以什么方式和这个ecu通信，即CAN节点或以太网节点 |
| infoCollScriptType | String | 是 | 信息采集脚本类型：指升级对象信息采集时所需要用到的脚本类型，如A或B或C |
| instAlgorithmType | String | 是 | 安装算法类型：指SA算法的类型，由车厂提供总范围，如A或B或C |
| instScriptURL | String | 是 | 安装脚本地址：指升级对象刷写时所需使用的安装脚本地址 |

### OTA版本检查请求

#### 说明

TBOX向OTA云端发起版本检查，确认是否需要进行安装升级。如果存在新的可升级版本，需要将结果以消息推送的方式同步到App端。

新版本检查请求触发场景如下：

* HU主动发起版本检测
* APP端主动发起版本检测
* TBOX每天首次上电后发起版本检测
* 云端发布新任务后推送通知到TBOX，TBOX发起版本检测

版本检测上行通知需要标识该请求来源类型。

主要流程：

* 判断车辆是否存在有效：升级对象基础信息表中是否存在对应记录
* 判断车辆是否在有关联升级任务
* 判断升级任务是否有效：在有效期内/状态为已审核通过
* 若存在有效的升级任务：检查上传的ecu列表参数是否匹配任务的升级固件集合
* 若存在有效的升级固件集合：检查是否存在有效的升级路径（车当前固件版本与升级任务固件清单中目标版本比对）。
* 若至少存在一个ecu固件对应的一条有效的升级路径，则认为该车辆存在可用的升级版本
* 补充判断：检查该路径是否有关联的升级包，若不存在，则该升级路径无效
* 针对一条有效的升级路径：
* 添加关联的升级包信息，用于为TBox下载提供必要的属性信息
* 计算该固件版本依赖的其他固件依赖列表。前提保证：后台固件版本管理功能 中，对于新增固件版本需要控制依赖深度，及避免依赖回环
* 返回版本检查结果：存在一条有效的升级路径，则认为检查结果为真，补充该次升级任务必要属性及升级固件列表信息；否则，检查结果为假。检查结果通过汇聚平台通道下发到TBox。检查结果主要包括如下数据：
* 升级固件版本列表：安装包大小、下载地址、安全校验、刷写脚本
* 升级固件版本依赖的其他固件版本：需要铺平到“升级固件版本列表”中
* 任务前置条件列表：升级前提条件确认
* 任务中途停止条件配置信息：该信息用于为Tbox端任务终止配置提供策略
* 版本检查请求记录需要在保存到数据库(表：tb\_fota\_version\_check)。请求主体信息存储到mysql，并留存一份记录到MongoDb(请求记录中的Ecu列表信息数据会比较大)。如果新版本检查记录为真，需要保存升级版本结果确认表记录(表：tb\_fota\_version\_check\_verify)

~~安全要求：~~

* ~~TBox请求时间戳与后台时间比对，若超过有效时间周期，可认为该请求失效~~

#### 接口url

#### 接口请求示例

{

"body": [

{

"firmwareCode": "VCode1",

"firmwareVersion": "V1.0",

"backupfirmwareVersion": : "V1.0"

}

],

"businessId": 1,

"businessType": 1,

"timestamp": 123456789,

"vin": "bq001"

}

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 请求头-序列号（同一个Tbox端需要唯一，可加日期区分） |
| timestamp | Long | 是 | 请求头-时间戳 |
| vin | String | 是 | 请求头-车辆标识（一般为vin码） |
| businessType | Integer | 是 | businessType=1 |
| sourceType | Integer | 否 | 1、HU主动发起版本检测  2、TBOX每天首次上电后发起版本检测  3、APP端主动发起版本检测  4、OTA云端发布新任务后推送通知到TBOX |
| body | Object | 是 | 请求消息体，定义如下 |

* body定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| ecuModels | List<EcuModel> | 是 | 车辆当前运行固件版本列表 |
| confVersion | String | 是 | 车辆可升级固件配置清单版本号 |
| entireVersionNo | String | 是 | 车辆整车版本号 |

* ecuModel定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| firmwareCode | String | 是 | 固件（软件）代码 |
| firmwareVersion | String | 是 | 固件（软件）版本 |
| ~~backupFirmwareVersion~~ | ~~String~~ | ~~是~~ | ~~固件备份版本~~ |
| bootVersion | String | 是 | Boot软件版本号 |
| hardwareVersion | String | 是 | ECU硬件版本号 |
| ecuId | String | 是 | ECU零件编号 |
| ecuSeq | String | 是 | ECU序列号 |
| diagnose | String | 是 | 诊断Id |
| ecuSystemSupplier | String | 是 | 系统供应商信息 |
| sysName | String | 是 | 系统名称 |

#### 响应示例

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 响应头-序列号（来自于TBox端生成） |
| timestamp | Long | 是 | 响应头-时间戳 |
| vin | String | 是 | 响应头-车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | businessType=101 |
| transId | Long | 是 | 升级事务Id |
| taskId | Long | 是 | 当前任务Id（云端下发） |
| reqId | Long | 是 | 云端生成的请求消息Id，用于与TBOX进行消息确认 |
| errorCode | Integer | 否 | 异常码 |
| errorMsg | String | 否 | 异常信息 |
| body | Object | 是 | 请求消息主体 |

* body属性对象定义如下（checkResult=2）：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| checkResult | Integer | 是 | 新版本检查结果(2=不存在新版本) |

* body属性对象定义如下（checkResult=1）：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| checkResult | Integer | 是 | 新版本检查结果(1=存在新版本) |
| entireVersionNo | String | 是 | 整车版本号 |
| targetVersionNo | String | 是 | 升级任务定义的大版本号 |
| taskTips | String | 是 | 任务提示 |
| upgradeMode | Integer | 是 | 升级模式 |
| disclaimer | String | 是 | 免责申明 |
| downloadTips | String | 是 | 下载提示 |
| autoDown | Integer | 是 | 自动下载设置（1=打开，2=关闭） |
| estimatedDownloadPackageSize | Integer | 是 | 预计总下载包大小,单位字节 |
| estimatedUpgradeTime | Integer | 是 | 预计升级时间,单位秒 |
| networkType | Integer | 是 | 后台约定什么网络可进行下载：待定 |
| upgradeConditions | List<UpgradeCondition> | 是 | 安装升级任务前提条件 |
| ecuFirmwareVersionInfos | List<EcuFirmwareVersionInfo> | 是 | 升级包相关信息 |

* UpgradeCondition定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| condCode | Integer | 是 | 终端升级条件代码，需要与终端约定，如：  1=速度  2=电压... |
| condValue | String | 是 | 升级条件值： |
| condValueType | Integer | 是 | 升级条件值类型：  0=数字类型 |
| operatorType | Integer | 是 | 操作符号类型：  0=算数运算符号  1=逻辑运算符号 |
| operatorValue | Integer | 是 | 操作运算符，如下：   * 当operatorType=0时，   0=“=”，1=“!=”，2=“>”，3=“<”，4=“>=”，5=“<=”   * operatorType=1时，待扩展 |

* EcuFirmwareVersionInfo定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| upgradeSeq | Integer | 是 | 升级顺序 从小到大依次升级 |
| groupSeq | Integer | 是 | 升级分组序号，同一个组中的固件同升同降，属于同一个groupSeq的固件列表升级顺序由顺序upgradeSeq决定 |
| dependenyFlag | Integer | 是 | 1=计划任务指定的包，2=被依赖的升级包 |
| firmwareId | Long | 否 | 固件Id |
| firmwareCode | String | 是 | 固件代码 |
| ecuId | String | 是 | 零件代码 |
| diagnose | String | 是 | 诊断Id |
| srcPkgInfo | PkgInfo | 否 | 原始包信息。  rollbackMode=1时必传 |
| dstPkgInfo | PkgInfo | 是 | 目标包信息。 |
| rollbackMode | Integer | 否 | 回滚模式：0=不回滚，1=回滚 |

升级包详细信息（PkgInfo）：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| pkgId | Long | 是 | 升级包Id |
| pkgType | Integer | 是 | 升级包类型：0=全量包，1=补丁包，2=差分包 |
| firmwareVersion | String | 是 | 固件版本 |
| downloadUrl | String | 是 | 升级包下载地址 |
| originFileSize | Long | 是 | 加密前升级包大小（单位字节） |
| fileSize | Long | 是 | 加密后升级包大小（单位字节） |
| scriptUrl | String | 否 | 刷写脚本下载地址 |
| isSecret | Integer | 是 | 是否为加密包：1=是加密包，2=否 |
| encryptAlg | String | 否 | 发布包加密算法来源于软件表中的加密算法：  1=AES加密/解密方式(当前系统唯一支持) |
| encryptSecret | String | 否 | 发布包加密密钥升级包加密密钥，由系统随机生成，一个包一个密钥（后端服务随机生成） |
| originShaCode | String | 是 | 原始包的Sha码 |
| shaCode | String | 是 | 升级包Sha码，用于与签名验证比对 |
| signAlg | String | 否 | 发布包签名算法来源于软件表中的签名算法。当前支持Sha1WithRSA签名算法，签名原始内容为升级包Sha码（属性：shaCode） |
| sign | String | 否 | 升级包签名信息，OTA云端调用PKI提供签名方法生成 |
| estimatedUpgradeTime | Integer | 是 | 预计升级时间,单位秒 |

示例如下：

{

"body": {

"disclaimer": "",

"downloadTips": "",

"ecuFirmwareVersionInfos": [

{

"chipInfo": "",

"componentCode": "VCode",

"difScriptUrl": "",

"downloadUrl": "www.baidu.com",

"encryptAlg": "",

"encryptSecret": "",

"fileSize": 0,

"firmwareCode": "VCode1",

"firmwareId": "VCode1",

"firmwareVersion": "V1.0",

"groupSeq": 1,

"hardwareVersion": "V1.0",

"maxConcurrent": 0,

"pkgId": 10000,

"pkgType": 0,

"rollbackMode": 0,

"sign": "",

"signAlg": "",

"upgradeSeq": 1

}

],

"estimatedDownloadPackageSize": 0,

"estimatedDownloadTime": 0,

"newVersionStatus": "0=无新版本，1=存在新版本",

"newVersionTips": 0,

"taskId": 0,

"taskPreConditions": [

{

"condCode": "speed",

"condValue": 50,

"condValueType": 1,

"operatorType": 0,

"operatorValue": 1

}

],

"taskTips": "",

"upgradeMode": 1

},

"businessId": 1,

"businessType": 1,

"timestamp": 123456789,

"vin": "bq001"

}

### ECU状态同步上报（来源于TBox）

#### 说明

TBOX根据业务规则定义的时间节点收集整车ECU信息上报到OTA云端，实现与云端信息同步。

#### 接口url

MQTT协议通道

#### 接口请求示例

{

"businessId": 1,

"businessType": 32,

"timestamp": 123456789,

"vin": "bq001",

"body": {

}

}

#### 接口入参

参考：[“版本检查”小节说明](#_OTA版本检查请求)

#### 响应示例

无

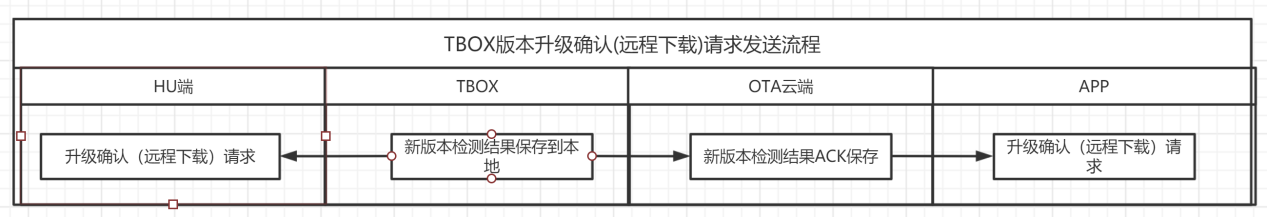
### OTA新版本下载请求发送（来源于TBox）

#### 说明

TBOX通过版本检查请求获取到的新版本升级结果信息保存到TBOX本地缓存后，与OTA远端回应一个确认信息。OTA云端需要将该消息转发到App端，APP客户端通过该消息确认TBOX获取了一个新的升级版本。如果该确认消息在业务定义的超时时间内没有转发同步到APP端，APP端需要通过超时机制保证响应。

OTA云端接收到该上行消息后，变更该车辆在当次升级过程中的状态为“待下载”。车辆升级状态参考表“**OTA云端定义的当前车辆所处升级阶段状态表**”。

流程示意图如下：



#### 接口url

#### 接口请求示例

{

"businessId": 1,

"businessType": 1,

"timestamp": 123456789,

"vin": "bq001",

"body": {

"transId": 1000086

}

}

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 请求头-序列号（由TBox端生成） |
| timestamp | Long | 是 | 请求头-时间戳 |
| vin | String | 是 | 请求头-车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | businessType=2 |
| transId | Long | 是 | 升级事务Id（服务端第一次版本检查结果确认时生成），用于关联整个升级流程，由TBox回传 |
| taskId | Long | 是 | 当前任务Id（云端下发） |
| reqId | Long | 否 | 云端生成，版本检查结果返回，该请求回传，用于与TBox进行消息确认 |
| sourceType | Integer | 否 | 指令来源：  1=来自Hu  2=来自TBOX  3=来自APP  4=来自OTA云端 |
| errorCode | Integer | 否 | 异常码 |
| errorMsg | String | 否 | 异常信息 |
| body | Object | 否 | 请求消息体（暂无） |

#### 响应示例

无

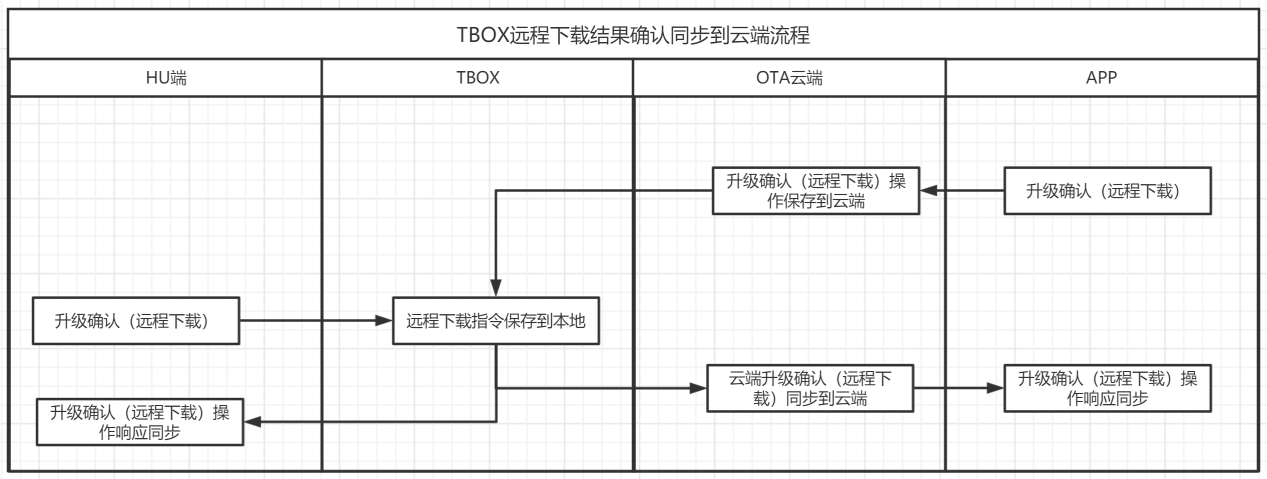
### OTA版本下载确认结果发送（来源于TBOX）

#### 说明

通常在HU/APP端进行“下载确认”操作后，TBOX会最终收到该操作结果。TBOX收到版本“下载确认”操作结果后需保证同步状态到OTA云端。

* HU端操作，HU与TBOX直接通信。TBOX将该结果同步上行到OTA云端，保证车云状态同步。OTA云端需要将“下载确认”操作结果消息转发到APP端
* APP端操作，需要借助OTA云端通道来下发“下载确认”操作结果到TBOX。TBOX将该结果同步上行到OTA云端，保证车云状态同步。OTA云端需要将“下载确认”操作结果消息转发到APP端

流程示意图如下：



APP端操作主要流程：

1. APP端点击升级确认(远程下载)操作，消息指令上行到OTA云端，云端记录该请求指令，并将该消息下行到TBOX。
2. TBOX消息上行：TBox端将接受到的升级确认(远程下载)结果回复一个ACK通过上行通道与云端确认。TOBX同时需将该消息指令同步到HU端。
3. 数据更新：OTA云端修改升级表对应(表tb\_fota\_version\_check\_verify)数据 记录，修改“安装确认”状态及时间字段，
4. 数据更新：更新任务车辆升级表(表tb\_fota\_plan\_obj\_list)中对应的状态字段
5. 消息同步：通过推送通道下发到下游App端。

HU端操作主要流程：

1. TBOX消息上行：HU端点击升级确认(远程下载)操作，TBox端将接受到的升级确认(远程下载)结果回复一个ACK通过上行通道与OTA云端确认。
2. 数据更新：OTA云端修改升级表对应(表tb\_fota\_version\_check\_verify)数据 记录，修改“安装确认”状态及时间字段，
3. 数据更新：更新任务车辆升级表(表tb\_fota\_plan\_obj\_list)中对应的状态字段
4. 消息同步：通过推送通道下发到下游App端。

#### 接口url

#### 接口请求示例

{

"body": 0,

"businessId": 1,

"businessType": 1,

"timestamp": 123456789,

"vin": "bq001"

}

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 请求头-序列号（上一次获取新版本检查请求序列号，由TBox端生成） |
| timestamp | Long | 是 | 请求头-时间戳 |
| vin | String | 是 | 请求头-车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | businessType=3 |
| transId | Long | 是 | 升级事务Id（服务端第一次版本检查结果确认时生成），用于关联整个升级流程，由TBox回传 |
| taskId | Long | 是 | 当前任务Id（云端下发） |
| reqId | Long | 否 | 云端生成，版本检查结果返回，该请求回传，用于与TBox进行消息确认 |
| sourceType | Integer | 否 | 指令来源：  1=来自Hu  2=来自TBOX  3=来自APP  4=来自OTA云端 |
| errorCode | Integer | 否 | 异常码 |
| errorMsg | String | 否 | 异常信息 |
| body | Object | 是 | 请求消息体 |

* body定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| verifyResult | Integer | 是 | 确认状态：1=YES（确认远程下载），2=NO（放弃远程下载） |
| verifySource | Integer | 是 | 确认来源类型：1=Hu端确认，2=TBOX自动下载，3=APP端确认 |
| [immediate](http://dict.cn/immediately)Download | Interger | 是 | 是否马上启动下载过程：1=TBOX会立即启动下载动作，2=TBOX会延后下载（需要车辆上电） |

#### 响应示例

无

### OTA版本升级包下载

#### 说明

通过固件版本检查接口获取到升级固件下载地址进行下载。后台提供基于OSS存储的文件服务器，TBox通过升级包URL地址直接下载。如果需要考虑安全因素，需要通过后台应用服务器提供安全下载接口。

固件升级包需要提供相关安全措施，包括加解密/安全校验等。

#### 接口url

#### 接口请求示例

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| downloadUrl | String | 是 | 包文件下载地址 |

#### 响应示例

http协议响应流

### OTA下载进度汇报

#### 说明

该接口用于TBox端向服务端汇报升级包下载进度及状态，汇报频次采用固定跳动比率：如1%，具体逻辑由TBOX承载。

下载阶段状态定义如下：



图：下载阶段状态图

升级下载进度状态定义如下：

* 下载中
* 下载完成（成功）
* 下载完成（失败）
* 下载完成（保留）
* 升级下载终止（保留）
* 升级下载中等待（保留）

**“下载中”**进度汇报内容包括：

* 基本属性：整体下载百分比、预估下载剩余时间、上报时间、整体下载完成状态等，后台需要更新升级任务对象表(表：tb\_fota\_plan\_obj\_list)记录状态属性字段。

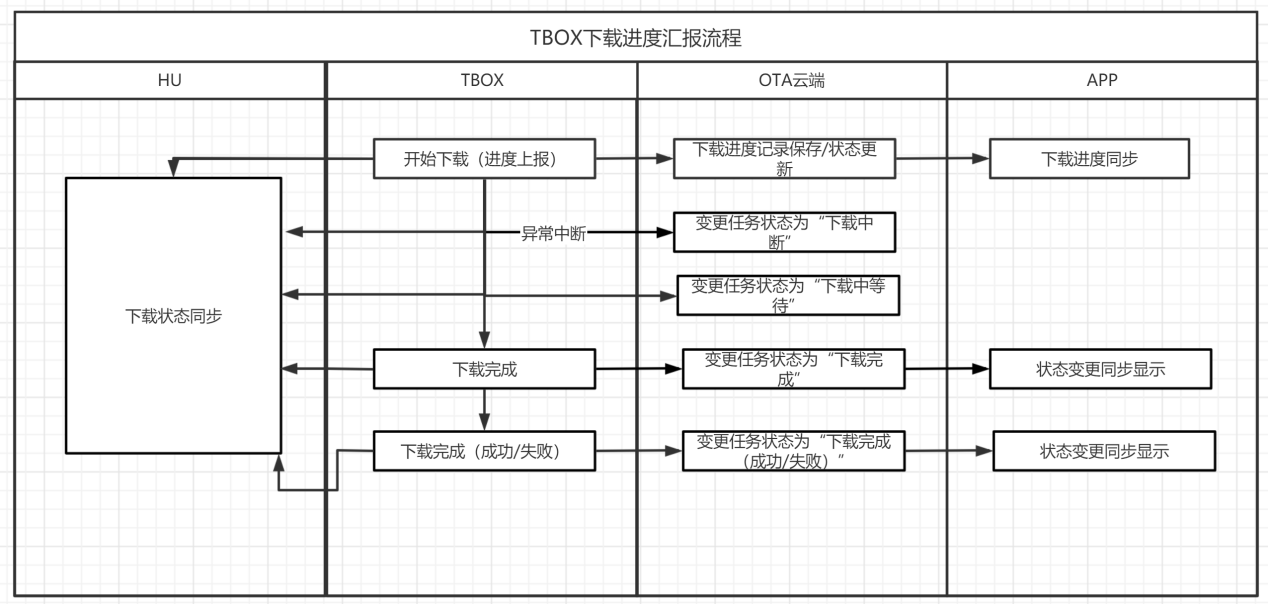
升级下载中异常状态包括：升级下载终止/升级下载中等待

**“升级下载完成（成功/失败）”**结果汇报内容包括：

* 最终完成状态
  + 单个Ecu固件升级下载进度列表详情：

升级包Id，下载进度百分比、花费时间、下载完成状态、下载完成字节数、累计下载次数。

流程示意图如下：



主要流程：

1. Tbox端根据汇报策略上传下载进度信息，后台服务端(根据请求序列号确认对应数据记录)保存下载升级进度汇报信息。
2. 数据保存：

* 基本属性：更新到mysql升级版本结果确认表(tb\_fota\_plan\_obj\_list)数据记录对应字。
* Ecu升级固件下载进度列表详情：存储到Mongodb。

1. 消息转发到APP：后台服务通过消息推送通道将升级记录上报信息同步到App端

业务汇报规则：

TBOX按照下载进度的1%个点进行跳变汇报

#### 接口url

#### 接口请求示例

{

"body": [

"downloadProcessType"：7,

"failMsg":"download fail",

"downloadDetails":

{

"accumulateNum": 0,

"downloadPercentRate": 0,

"downloadSize": 0,

"downloadSpendTime": 0,

"downloadStatus": 0,

"pkgIg": 0

}

],

"businessId": 1,

"businessType": 1,

"timestamp": 123456789,

"vin": "bq001"

}

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 请求头-序列号（由TBox端生成） |
| timestamp | Long | 是 | 请求头-时间戳 |
| vin | String | 是 | 请求头-车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | businessType=4 |
| transId | Long | 是 | 升级事务Id（服务端第一次版本检查结果确认时生成），用于关联整个升级流程，由TBox回传 |
| taskId | Long | 是 | 当前任务Id（云端下发） |
| reqId | Long | 否 | 云端生成，版本检查结果返回，该请求回传，用于与TBox进行消息确认 |
| sourceType | Integer | 否 | 指令来源：  1=来自Hu  2=来自TBOX  3=来自APP  4=来自OTA云端 |
| errorCode | Integer | 否 | 异常码 |
| errorMsg | String | 否 | 异常信息 |
| body | Object | 是 | 请求消息体 |

* body定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| downloadProcessType | Integer | 是 | 升级过程类型：1=下载中（实时进度上报），2=下载中止，3=下载中等待，4=下载完成，5=下载成功（下载包校验成功），6=下载失败（下载包校验失败） |
| downloadPercentRate | Integer | 否 | 整体下载完成百分比。downloadProcessType=1时必传 |
| downloadFullSize | Integer | 否 | 总包大小。  downloadProcessType=1时必传 |
| downloadFinishedSize | Integer | 否 | 已完成下载包大小。  downloadProcessType=1时必传 |
| downloadProcessDetails | List<DownloadProcessDetail> | 否 | * downloadProcessType=1/2/3/4时，该值为空 * downloadProcessType=5时，该值不为空，用于详细说明每个安装包下载情况 * downloadProcessType=6时，该值不为空，用于详细说明每个安装包下载情况 |
| estimateRemainTime | Integer | 是 | 预估下载完成时间，时间单位为秒 |
| startTime | Long | 是 | 全包整体下载开始时间 |
| endTime | Long | 是 | 全包整体下载结束时间 |

* DownloadProcessDetail定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| firmwareId | Long | 是 | 固件Id |
| pkgId | Long | 是 | 安装包Id |
| downloadSpendTime | Integer | 是 | 单个Ecu升级包下载花费时间，时间单位为s |
| downloadStatus | Integer | 否 | 下载完成状态:1=已完成，2=未完成。  downloadProcessType=6时必传 |
| downloadSize | Integer | 否 | 已完成下载字节数。downloadProcessType=6时必传。 |
| accumulateNum | Integer | 是 | 累计尝试下载次数。downloadProcessType=6时必传 |
| startTime | Long | 是 | 下载开始时间 |
| endTime | Long | 是 | 下载结束时间 |
| errorCode | Integer | 否 | 错误码 |
| errorMsg | String | 否 | 异常信息 |

#### 响应示例

无

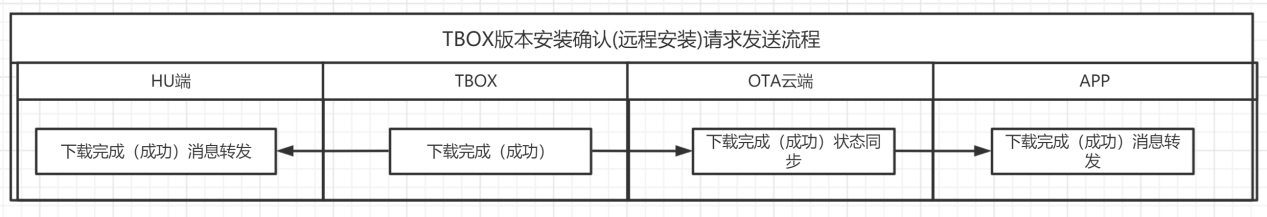
### OTA版本安装确认请求

#### 说明

消息指令上行来源：TBOX

TBOX将新版本检查需要的所有包全部下载完成之后，将该状态同步到OTA云端。OTA云端需要将该消息转发到手机App。TBOX将该消息转发给HU相关逻辑由TBOX本身提供实现。

流程示意图如下：



#### 接口url

#### 接口请求示例

{

"businessId": 1,

"timestamp": 123456789,

"vin": "bq001",

"businessType": 4,

"transId":100086,

"body": {

},

}

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 请求头-序列号 |
| timestamp | Long | 是 | 请求头-时间戳 |
| vin | String | 是 | 请求头-车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | businessType=5 |
| transId | Long | 是 | 升级事务Id（服务端第一次版本检查结果确认时生成），用于关联整个升级流程，由TBox回传 |
| taskId | Long | 是 | 当前任务Id（云端下发） |
| reqId | Long | 否 | 云端生成，版本检查结果返回，该请求回传，用于与TBox进行消息确认 |
| sourceType | Integer | 否 | 指令来源：  1=来自Hu  2=来自TBOX  3=来自APP  4=来自OTA云端 |
| errorCode | Integer | 否 | 异常码 |
| errorMsg | String | 否 | 异常信息 |
| body | Object | 否 | 请求消息体 |

#### 响应示例

无

### OTA版本安装确认完成

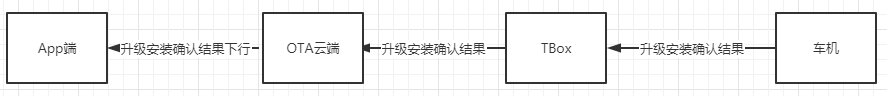
#### 说明

消息指令来源：TBOX端收到“安装确认”指令

通常在HU/APP端进行“安装确认”操作后，TBOX会最终收到该操作结果。TBOX收到版本“安装确认”操作结果后需保证同步状态到OTA云端。

* HU端操作，HU与TBOX直接通信。TBOX将该结果同步上行到OTA云端，保证车云状态同步。OTA云端**需要**将“安装确认”操作结果消息转发到APP端
* APP端操作，需要借助OTA云端通道来下发“安装确认”操作结果到TBOX。TBOX将该结果同步上行到OTA云端，保证车云状态同步。OTA云端**无需**将“安装确认”操作结果消息转发到APP端

流程示意图如下：



主要流程：

1. 消息上行：TBox端根据上次获取新版本检查请求序列号(businessId)通过上行通道汇 报到服务器端
2. 数据更新：根据版本检查请求序列号查找对应(表tb\_fota\_version\_check\_verify)数据 记录，修改“升级确认”状态及时间字段
3. 数据更新：更新任务车辆升级表(表tb\_fota\_plan\_obj\_list)中对应的状态字段
4. 消息同步：通过推送通道下发到下游App端。

#### 接口url

协议：MQTT/汇聚平台

#### 接口请求示例

{

"businessId": 1,

"timestamp": 123456789,

"vin": "bq001",

"businessType": 4,

"body": {

"transId":"transId1",

"verifyResult":"1",

"verifySource":"2",

"installedType":"2",

"installedTime":1000086

}

}

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 请求头-序列号（由TBox端生成） |
| timestamp | Long | 是 | 请求头-时间戳 |
| vin | String | 是 | 请求头-车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | businessType=6 |
| transId | Long | 是 | 升级事务Id（服务端第一次版本检查结果确认时生成），用于关联整个升级流程，由TBox回传 |
| taskId | Long | 是 | 当前任务Id（云端下发） |
| reqId | Long | 否 | 云端生成，版本检查结果返回，该请求回传，用于与TBox进行消息确认 |
| sourceType | Integer | 否 | 指令来源：  1=来自Hu  2=来自TBOX  3=来自APP  4=来自OTA云端 |
| errorCode | Integer | 否 | 异常码 |
| errorMsg | String | 否 | 异常信息 |
| body | Object |  | 请求消息体 |

* body定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| verifyResult | Integer | 是 | 安装确认操作结果：1=确认安装2=放弃安装(暂时保留) |
| verifySource | Integer | 是 | 安装确认操作来源：1=HU端，2=APP端 |
| installedType | Integer | 是 | 1=立即安装，2=预约安装。verifyResult=1时必传 |
| installedTime | Long | 否 | 预约安装时间点。installedType=2时必传 |

#### 响应示例

无

### OTA升级进度/状态汇报

#### 说明

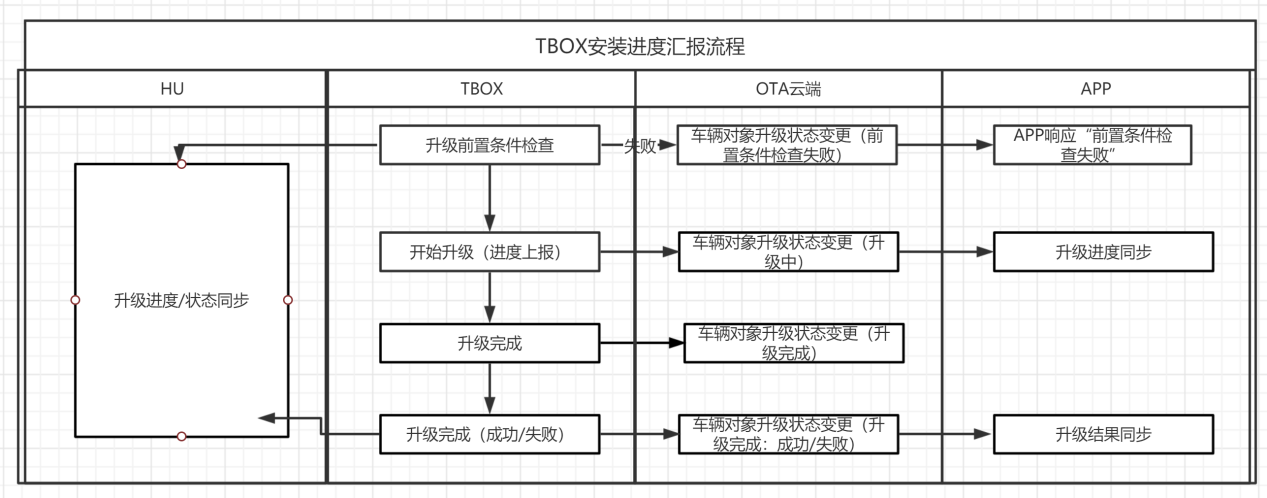
该接口用于TBox端向服务端汇报升级包安装进度，汇报规则由TBox端制定，如当次升级ECU各位为8个，当前升级第3个，表示为3/8。

安装阶段状态定义参考：[OTA云端定义的车辆所处升级阶段状态表](#_OTA云端定义的车辆所处升级阶段状态表)

汇报内容包括：

* 前置条件检查失败：失败提示消息
* 升级取消：来自HU端的取消升级操作
* 升级安装中进度汇报：安装进度（百分比）/升级总时长/升级剩余时间
* 升级结果完成（成功/失败）汇报：完成状态
  + 升级总体信息：安装状态/安装进度/升级总时长/升级剩余时间/失败原因
* Ecu固件列表升级安装进度详情：升级包Id/安装进度/花费时间/安装完成状态/安装次数/失败原因

流程示意图如下：



主要流程：

1. Tbox端根据策略上报安装进度信息，后台服务端根据该次升级任务的请求序列记录 此次安装升级汇报信息。
2. 数据保存：

* 基本属性：更新到mysql升级版本结果确认表(tb\_fota\_version\_check\_verify)数据记录对应字。
* 状态变更：更新到mysql升级版本结果确认表(tb\_fota\_plan\_obj\_list)数据记录对应字。
* Ecu升级固件安装进度列表详情：存储到Mongodb。

1. 消息同步转发：通过消息推送通道将升级进度/状态上报信息同步转发到App端

#### 接口url

协议：MQTT/汇聚平台

#### 接口请求示例

{

"body": [

{

"accumulateNum": 0,

"installedSpendTime": 0,

"installedStatus": 0,

"pkgId": 0

}

],

"businessId": 1,

"businessType": 7,

"timestamp": 123456789,

"vin": "bq001"

}

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 请求头-序列号（上一次获取新版本检查请求序列号，由TBox端生成） |
| timestamp | Long | 是 | 请求头-时间戳 |
| vin | String | 是 | 请求头-车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | **businessType=7** |
| transId | Long | 是 | 升级事务Id（服务端第一次版本检查结果确认时生成），用于关联整个升级流程，由TBox回传 |
| taskId | Long | 是 | 当前任务Id（云端下发） |
| reqId | Long | 否 | 云端生成，版本检查结果返回，该请求回传，用于与TBox进行消息确认 |
| sourceType | Integer | 否 | 指令来源：  1=来自Hu  2=来自TBOX  3=来自APP  4=来自OTA云端 |
| errorCode | Integer | 否 | 异常码 |
| errorMsg | String | 否 | 异常信息 |
| body | Object | 是 | 请求消息体 |

* body定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| installedProcessType | Integer | 是 | 升级过程类型：1=前置条件检查失败上报，2=升级安装进度上报（升级中），3=取消安装（包括取消预约安装），4=开始升级 |
| totalPkgNum | Integer | 否 | 升级包总个数。  installedProcessType=2时必传 |
| currentPkgIndex | Integer | 否 | 当前第几个升级包  。installedProcessType=2时必传 |
| installedSpendTime | Integer | 否 | 整体升级花销时间（单位为秒）。  installedProcessType=2时必传 |
| installedRemainTime | Integer | 否 | 整体升级完成预估剩余时间（单位为秒）。downloadProcessType=2时必传 |
| upgradeConditions | List<UpgradeCondition> | 否 | installedProcessType=1时必传 |
| installProcessDetails | List<InstallProcessDetail> | 否 | 当前升级的列表详细情况 |

InstallProcessDetail定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| ecuId | String | 是 | 零件代号 |
| firmwareCode | String | 是 | 固件代码 |
| diagnose | String | 是 | 诊断Id |
| firmwareVersion | String | 是 | 固件版本号 |
| pkgId | Long | 是 | 升级包Id |
| installedPercentRate | Integer | 是 | 升级进度:1-100 |
| installedSpendTime | Integer | 是 | 当前ecu升级花销时间（单位为秒）。  installedProcessType=2时必传 |
| installedRemainTime | Integer | 是 | 安装剩余下载时间，单位为秒 |
| erroCode | Integer | 否 | 错误码 |
| errorMsg | String | 否 | 异常描述消息 |

#### 响应示例

### OTA升级结果汇报

#### 说明

将最终升级成功或失败的结果汇报到OTA云端。

汇报内容包括：

* 基本属性：升级结果(成功/失败)
  + Ecu固件升级安装结果列表详情，记录单个Ecu固件升级安装的最终结果

流程示意图如下：



主要流程：

1. 更新升级版本结果确认表(tb\_fota\_version\_check\_verify)数据记录“升级结果”字段，该数据记录进入终态，后续不再参与升级流程逻辑。
2. 更新升级任务相关表属性字段，包括：

* OTA升级计划对象清单表(tb\_fota\_plan\_obj\_list)：“升级状态”字段，标识该Tbox终端升级状态。
* 更新该次任务关联升级任务明细(tb\_fota\_plan\_task\_detail)：该表描述了升级对象该次升级任务中每个固件升级的最终结果及失败情况下的信息收集
* 更新OTA升级对象固件清单表(tb\_fota\_firmware\_list)：该表描述了升级对象该次升级任务完成后最新的固件版本信息，包括最新版本、最后一次升级时间等

1. 消息同步：通过消息推送通道将升级结果信息同步到App端

#### 接口url

协议：MQTT/汇聚平台

#### 接口请求示例

{

"body": {

"taskId":"taskIdNO3",

"endTime": 123456,

"installedResult": 1,

"installedResultDetails": [

{

"endTime": 133333333333,

"firmwareId": 1234567,

"firmwareVersionNo": "V1.10",

"lastFailReason": "顶顶顶顶",

"lastFailTIme": 122222222222,

"retryNum": 3,

"retryTime": 122222222222,

"startTime": 122222222222,

"status": 0

}

],

"startTime": 123456,

"status": 1

},

"businessId": 1,

"businessType": 1,

"timestamp": 123456789,

"vin": "bq001"

}

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 请求头-序列号（由TBox端生成） |
| timestamp | Long | 是 | 请求头-时间戳 |
| vin | String | 是 | 请求头-车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | businessType=8 |
| transId | Long | 是 | 升级事务Id（服务端第一次版本检查结果确认时生成），用于关联整个升级流程，由TBox回传 |
| taskId | Long | 是 | 当前任务Id（云端下发） |
| reqId | Long | 否 | 云端生成，版本检查结果返回，该请求回传，用于与TBox进行消息确认 |
| sourceType | Integer | 否 | 指令来源：  1=来自Hu  2=来自TBOX  3=来自APP  4=来自OTA云端 |
| errorCode | Integer | 否 | 异常码 |
| errorMsg | String | 否 | 异常信息 |
| body | Object | 是 | 请求消息体 |

* body属性定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| installedResult | Integer | 是 | 升级结果：1=升级完成（成功），2=升级未完成，3=升级失败 |
| startTime | Long | 是 | 升级开始时间 |
| endTime | Long | 是 | 升级结束时间 |
| installedResultDetails | List<InstalledResultDetail> | 否 | 升级结果明细详情，包括该次升级所有固件升级包相关信息 |

* InstalledResultDetail定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| firmwareId | Long | 是 | 固件Id |
| firmwareCode | String | 是 | 固件代码 |
| firmwareVersionNo | String | 是 | 升级完成后固件版本 |
| ecuId | String | 是 | 零件号 |
| diagnose | String | 是 | 诊断Id |
| pkgId | Long | 是 | 升级包Id |
| ~~backupFirmwareVersionNo~~ | ~~String~~ | ~~否~~ | ~~当前系统最终运行版本（如果是双系统能升级，可能会存在备份版本）~~ |
| startTime | Long | 是 | 升级开始时间 |
| endTime | Long | 是 | 升级结束时间 |
| retryNum | Integer | 否 | 重试次数 |
| retryTime | Long | 否 | 最后重试时间 |
| status | Integer | 是 | ecu固件升级结果：1=成功，2=失败 |
| errorCode | String | 否 | 最终失败错误码。根据实际业 |
| errorMsg | Integer | 否 | 最终失败原因务情况进行定义 |

#### 响应示例

无

### 升级日志和错误信息文件上传

#### 说明

Tbox需要保存OTA整个过程中的⽇志及报⽂。当⻋辆升级结束时，需要将以上日志上报⾄OTA云端。

接口协议：HTTPS协议。

过程：

1. TBox请求OTA文件日志服务，上传日志文件。
2. OTA文件日志服务存储日志到对象存储中，下发kafka消息通知到OTA云端
3. OTA云端处理消息，建立日志文件记录到车辆升级过程的关联关系

#### 接口url

POST {HOST}/vehicle/file/upload/otaUpgrade

#### 接口请求示例

#### 接口入参

请求参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| vin | String | 是 | 车辆Vin码 |
| transId | Long | 是 | 升级事务Id |
| taskId | Long | 是 | 升级任务Id |
| type | Integer | 是 | 类型：待定义 |
| file | mulitipart | 是 | 过程和错误日志文件 |

#### 响应示例

### OTA终止升级结果上行（预留）

#### 说明

Tbox端将取消指令后终止结果上行到云端，云端转发给APP端（待定）

#### 接口url

无

#### 接口请求示例

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 请求头-序列号（由TBox端生成） |
| timestamp | Long | 是 | 请求头-时间戳 |
| vin | String | 是 | 请求头-车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | businessType=10 |
| body | Object | 是 | 响应消息体 |

* body属性定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| transId | Long | 是 | 升级事务Id（服务端第一次版本检查结果确认时生成），用于关联整个升级流程，由TBox回传 |
| stopResult | Integer | 是 | 终止结果：1=未获取任务终止,2=未下载终止，3=未执行任务终止 |

#### 响应示例

无

## 面向Tbox的消息下行

### 唤醒机制

OTA云端需要根据车辆实际在线状态，判断是否需要对TBOX进行唤醒操作。如果车辆不在线时，需要保存对应的消息指令，同时唤醒TBOX，在TBOX完成唤醒后完成指令下发操作。可以根据交互端对消息的实延要求，对TBOX执行唤醒操作

### 云端升级通知下行

#### 说明

用于向TBox端下发新版本通知信息，由OTA云端服务服务根据升级计划调度主动触发。OTA云端需要制定相应的推送策略（包括重试机制）保证TBOX能接收到该计划通知。

TBOX收到该消息时需要向OTA云端发起版本检查请求，OTA云端通过该请求确认升级通知消息已送达到TBOX端。

消息推送计划表：tb\_fota\_upgrade\_push\_shedule。

主要流程：

1. OTA云端云端发布新任务后，针对升级车辆对象根据推送规则（不同地区/不同日期）制定推送计划任务，新增调度记录表记录
2. OTA云端定时计划主动下发升级通知推送到对应升级车辆
3. 升级车辆TBOX端收到新版本升级通知后，发起版本检查请求

流程示意图参考：[APP升级流程](#_系统交互流程)

#### 接口url

#### 接口请求示例

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 序列号（服务器端生成） |
| timestamp | Long | 是 | 时间戳 |
| vin | String | 是 | 车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | businessType=104,服务器端主动下发 |
| reqId | Long | 是 | 云端生成消息交互Id，用于与TBox进行消息确认 |
| body | Object | 否 | 消息体，定义如下 |

#### 响应示例

### APP客户端请求TBOX版本检查消息下行

#### 说明

消息来源：APP客户端的新版本检查请求，由OTA云端下发到TBOX端

流程示意图参考：[APP升级流程](#_系统交互流程)

#### 接口url

#### 接口请求示例

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 序列号（服务器端生成） |
| timestamp | Long | 是 | 时间戳 |
| vin | String | 是 | 车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | businessType=105,App客户端请求下发 |
| reqId | Long | 是 | 云端生成消息交互Id，用于与TBox进行消息确认 |
| body | Object | 否 | 消息体，定义如下 |

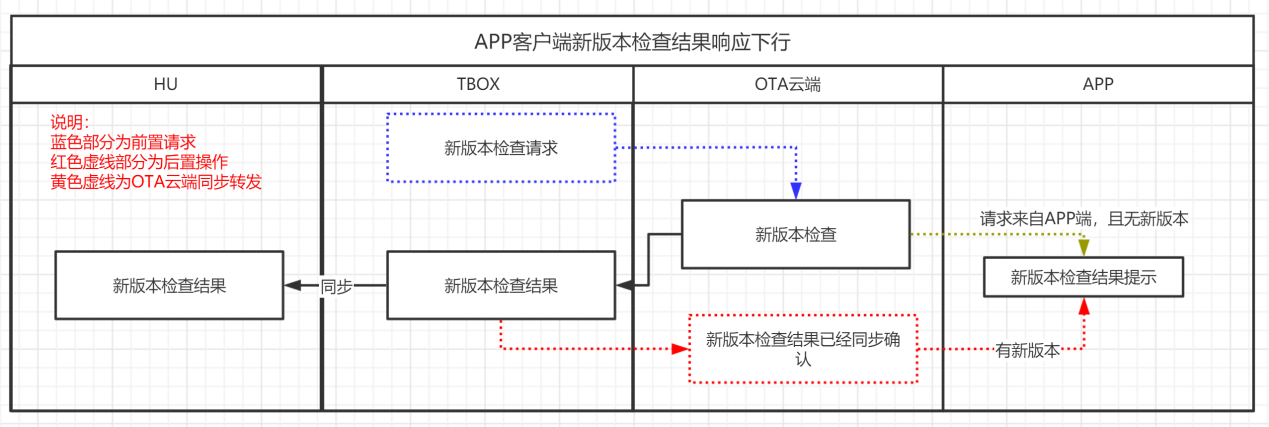
#### 响应示例

### 版本检查结果下发

#### 说明

OTA云端向TBox端下发版本检查结果信息，该响应来自于版本检查请求。TBOX端接收到该消息时，需要发送一个ACK消息将已收到版本检查结果状态同步到OTA云端。该ACK消息由“版本升级确认”上行消息承载。

下行示意图：



特殊说明：

1. 版本检查请求来自于APP客户端：不存在新版本情况下，OTA云端直接推送消息（不存在新版本）转发到APP客户端；存在新版本情况下，需要TBOX与OTA云端确认后再由OTA云端同步消息转发到APP客户端
2. 新版本检查结果下发到TBOX后与HU操作逻辑由TBOX决定

#### 接口url

#### 接口请求示例

#### 接口入参

参考：[新版本检查请求](#_响应示例_17)

#### 响应示例

示例如下：

{

"body": {

"disclaimer": "",

"downloadTips": "",

"ecuFirmwareVersionInfos": [

{

"chipInfo": "",

"componentCode": "VCode",

"difScriptUrl": "",

"downloadUrl": "www.baidu.com",

"encryptAlg": "",

"encryptSecret": "",

"fileSize": 0,

"firmwareCode": "VCode1",

"firmwareId": "VCode1",

"firmwareVersion": "V1.0",

"groupSeq": 1,

"hardwareVersion": "V1.0",

"maxConcurrent": 0,

"pkgId": 10000,

"pkgType": 0,

"rollbackMode": 0,

"sign": "",

"signAlg": "",

"upgradeSeq": 1

}

],

"estimatedDownloadPackageSize": 0,

"estimatedDownloadTime": 0,

"newVersionStatus": "0=无新版本，1=存在新版本",

"newVersionTips": 0,

"taskId": 0,

"taskPreConditions": [

{

"condCode": "speed",

"condValue": 50,

"condValueType": 1,

"operatorType": 0,

"operatorValue": 1

}

],

"taskTips": "",

"upgradeMode": 1

},

"businessId": 1,

"businessType": 1,

"timestamp": 123456789,

"vin": "bq001"

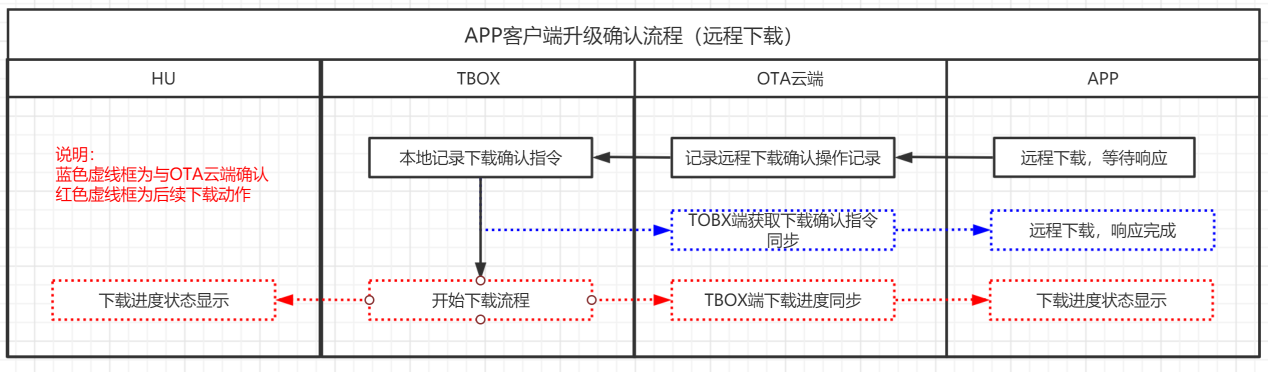
}

### OTA下载确认结果下行（点击“远程下载”）

#### 说明

向TBox下发来自App端的升级确认消息。APP实际进行的是“远程下载”操作。

消息下行示意图如下：



#### 接口url

#### 接口请求示例

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 响应头-序列号（TBox端产生，对应TBox上行“升级确认请求”消息头-序列号） |
| timestamp | Long | 是 | 响应头-时间戳 |
| vin | String | 是 | 响应头-车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | businessType=102 |
| transId | Long | 是 | 升级事务Id（服务端第一次版本检查结果确认时生成），用于关联整个升级流程，由TBox回传 |
| taskId | Long | 是 | 当前任务Id（云端下发） |
| reqId | Long | 否 | 云端生成，版本检查结果返回，该请求回传，用于与TBox进行消息确认 |
| sourceType | Integer | 是 | 指令来源：  1=来自Hu  2=来自TBOX  3=来自APP  4=来自OTA云端 |
| errorCode | Integer | 否 | 异常码 |
| errorMsg | String | 否 | 异常信息 |
| body | Object | 是 | 确认操作结果明细 |

* body定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| verifyResult | Integer | 是 | 确认操作结果：1=Yes（确认进行远程下载），2=No（放弃进行远程下载） |

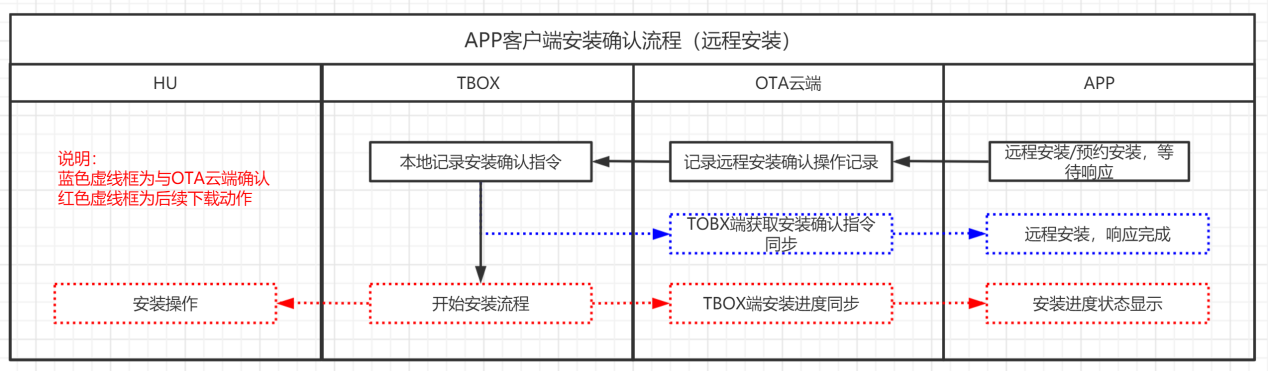
#### 响应示例

无

### OTA安装确认结果下行（确认远程安装）

#### 说明

向TBox下发来自App端的升级安装确认消息。APP实际进行的是“远程安装/预约”操作。



#### 接口url

#### 接口请求示例

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 响应头-序列号（TBox端产生，对应TBox上行“安装确认请求”消息头-序列号） |
| timestamp | Long | 是 | 响应头-时间戳 |
| vin | String | 是 | 响应头-车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | businessType=103 |
| transId | Long | 是 | 升级事务Id（服务端第一次版本检查结果确认时生成），用于关联整个升级流程，由TBox回传 |
| taskId | Long | 是 | 当前任务Id（云端下发） |
| reqId | Long | 否 | 云端生成，版本检查结果返回，该请求回传，用于与TBox进行消息确认 |
| sourceType | Integer | 是 | 指令来源：  1=来自Hu  2=来自TBOX  3=来自APP  4=来自OTA云端 |
| errorCode | Integer | 否 | 异常码 |
| errorMsg | String | 否 | 异常信息 |
| body | Object | 是 | 确认操作消息体 |

* body定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| verifyResult | Integer | 是 | 安装结果：1=确认安装，2=放弃安装 |
| installedType | Integer | 是 | 1=现在安装，2=预约安装.verifyResult=1时必传 |
| installedTime | Long | 否 | 预约安装时间点。installedType=2时必传 |

#### 响应示例

### OTA取消预约安装指令下行

#### 说明

向TBox下发来自App端的取消预约安装指令消息。APP实际进行的是“取消预约安装”操作。

#### 接口url

#### 接口请求示例

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 响应头-序列号（TBox端产生，对应TBox上行“安装确认请求”消息头-序列号） |
| timestamp | Long | 是 | 响应头-时间戳 |
| vin | String | 是 | 响应头-车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | businessType=103 |
| transId | Long | 是 | 升级事务Id（服务端第一次版本检查结果确认时生成），用于关联整个升级流程，由TBox回传 |
| taskId | Long | 是 | 当前任务Id（云端下发） |
| reqId | Long | 否 | 云端生成，版本检查结果返回，该请求回传，用于与TBox进行消息确认 |
| sourceType | Integer | 是 | 指令来源：  1=来自Hu  2=来自TBOX  3=来自APP  4=来自OTA云端 |
| errorCode | Integer | 否 | 异常码 |
| errorMsg | String | 否 | 异常信息 |
| body | Object | 否 | 确认操作消息体 |

#### 响应示例

### OTA任务终止消息下行

暂无

## OTA 手机App服务

### API接口请求通用说明

##### 接口调用原则

面向APP的服务接口用来提供消息上行通道，同步响应结果用于确保消息指令已经同步到云端。App端主逻辑（新版本检查等待/下载进度/安装进度同步）由推送通道异步消息通知实现。

##### 请求协议及格式

手机上行接口采用http服务。http请求头中需添加token，该token是用户登录后由用户系统返回到APP客户端，整个请求服务的鉴权在TSP网关完成。

##### APP客户端超时保证机制

OTA云端需要提供超时回补机制，通知到APP取消正在等待的界面展现逻辑。

##### 唤醒模式

OTA云端需要根据车辆实际在线状态，判断是否需要对TBOX进行唤醒操作。如果车辆不在线时，需要保存对应的消息指令，同时唤醒TBOX，在TBOX完成唤醒后完成指令下发操作。

### 获取当前TBOX升级状态

#### 说明

提供给APP客户端需要的TBOX当前所处的升级状态，APP客户端进入OTA版本升级检查操作需要根据该接口返回的升级状态来决定后续逻辑。

TBOX升级状态定义如下：

|  |  |
| --- | --- |
| 状态值 | 说明 |
| -1 | 无新版本：无有效的升级计划 |
| -2 | 无新版本：存在有效的升级计划，TBOX客户端检测不匹配 |
| 0 | 存在新版本：TBOX还未请求进行新版检测。 |
| 1 | 存在新版本：TBOX请求进行新版本检测且已经匹配到存在新版本，APP端需要异步等待TBOX完成整个流程后的消息通知。 |
| 11 | 存在新版本：待下载。TBOX本地已收到新版本检测结果,并已经同步该状态到云端。 |
| 12 | 存在新版本：开始下载。TBOX已收到下载指令,开始下载（并已与OTA云端确认状态同步） |
| 13 | 存在新版本：升级下载中 |
| 14 | 存在新版本：升级下载终止（因某些原因导致下载终止） |
| 15 | 存在新版本：升级下载中等待（因某些原因导致下载等待） |
| 16 | 存在新版本：TBOX执行下载完成（中间状态） |
| 17 | 存在新版本：TBOX执行下载完成（结果=成功）。同31 |
| 18 | 存在新版本：TBOX执行下载完成(结果=失败) |
| 31 | 存在新版本：待安装。待客户端安装确认(TBOX下载已经完成之后) |
| 32 | 存在新版本：开始升级。TBOX已收到升级安装指令，开始安装(并已与OTA云端确认同步) |
| 33 | 存在新版本：安装前置条件检查失败 |
| 34 | 存在新版本：升级安装中 |
| 35 | 存在新版本：执行升级，进入到升级结果准备阶段 |
| 101 | 存在新版本：升级安装成功 |
| 102 | 存在新版本：升级安装失败 |
| 111 | 不存在新版本：未获取任务终止 |
| 112 | 存在新版本：未下载终止 |
| 113 | 存在新版本：未升级终止 |

表：TBOX升级状态表

#### 接口url

POST {HOST}/fota/v1/app/upgrade/getTboxUpgradStatus

#### 接口请求示例

{

‘’vin‘’: "bq001",

‘’deviceType ‘’: 2

}

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| vin | String | 是 | 车辆标识(一般用Vin码代替) |
| deviceType | Integer |  | 01=安卓 02=ios |

#### 响应示例

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| status | Integer | 是 | 升级状态类型：参考TBOX升级状态表 |
| desc | String | 是 | 升级状态描述 |

{

“respCode”:” 00000”,

“respMsg”:”success”,

"respTime": 1596763523

“respData”:{

“status”:1,

“desc”:

}

}

### OTA版本检查请求

#### 说明

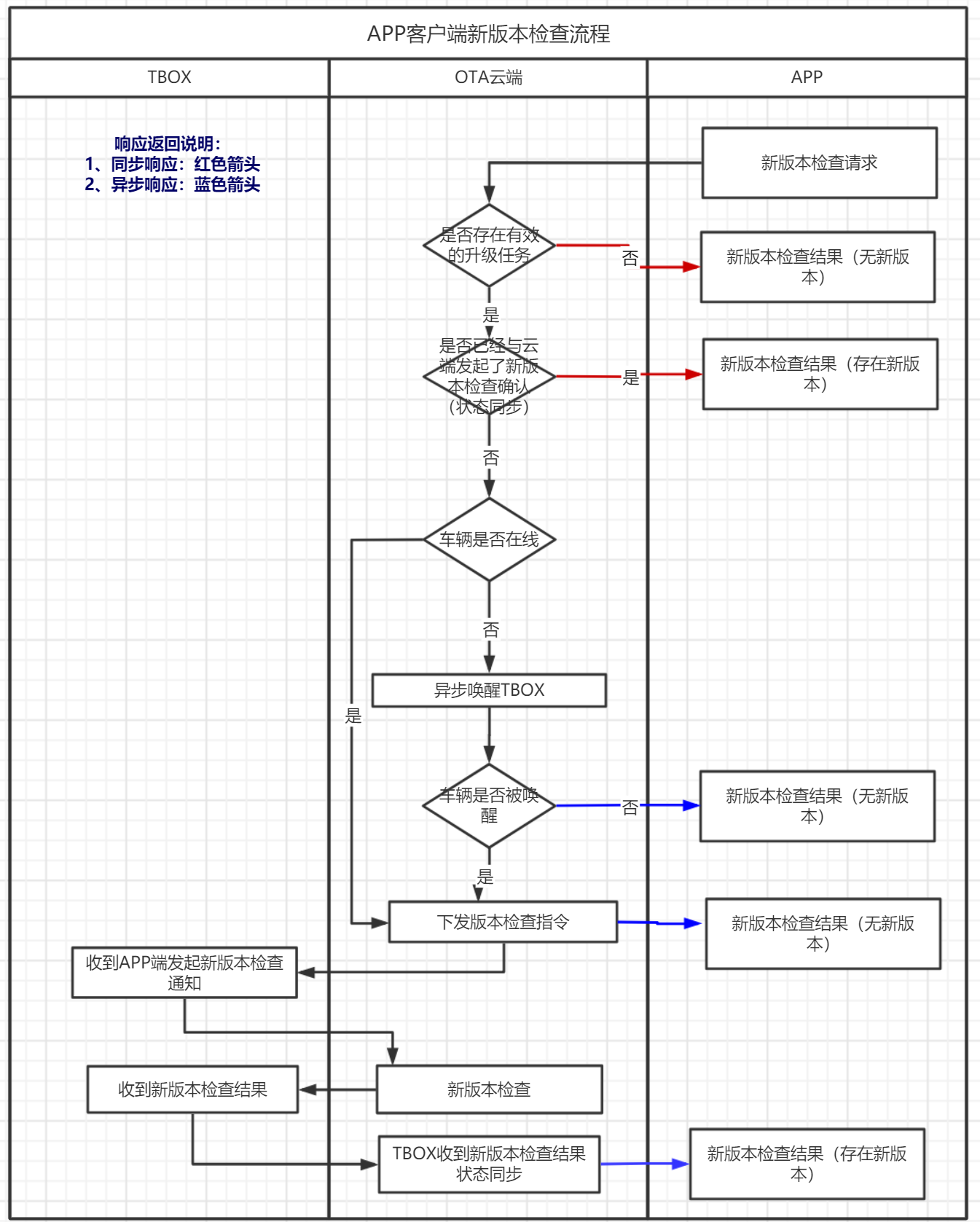
手机App端检查当前车辆是否存在新版本。新版本是否存在标准：手机APP绑定车辆是否存在有效的升级任务/存在有效升级任务但是检查失败。

* 如果不存在有效的升级任务，则直接同步“无新版本”响应到App端；
* 如果存在有效的升级任务，则判断TBOX是否已经与OTA云端进行版本升级确认，存在如下可能的情况：
* TBOX已进入新版本升级确认状态：OTA云端直接拼装新版本升级需要的相关信息转发给APP端。
* TBOX还没进入新版本升级确认状态：下发升级检查指令到TBOX，TBOX端收到OTA版本检查请求后主动发起版本检查。TBOX端不在线的情况下可能需要由云端唤醒操作。TBOX完成版本检查结果后需要同OTA云端同步状态，OTA云端将该消息转发给APP端。

超时处理：

APP客户端设定10S超时时间，OTA云端在超时时间内没有将新版本检查结果推送给APP客户端，则APP客户端默认等待结束，显示“当前已是最新版本”。

消息流程交互示意图如下：



表：APP新版本检查交互流程图

#### 接口url

POST {HOST}/fota/v1/app/upgrade/checkVersion

#### 接口请求示例

{

‘’vin‘’: "bq001",

‘’deviceType ‘’: 2

}

接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| vin | String | 是 | 车辆标识(一般用Vin码代替) |
| deviceType | Integer |  | 1=安卓 2=ios |

#### 响应示例

响应参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| checkResult | Integer | 是 | 1=有新版本，2=无新版本 |
| currentVersion | String | 是 | 当前版本 |
| targetVersion | String | 否 | 目标新版本。checkResult=1时不为空 |
| transId | Long | 否 | 升级事务Id。checkResult=1时且processType!=0时不为空 |
| newVersionTips | String | 否 | 新版本提示。checkResult=1时不为空 |
| downloadTips | String | 否 | 下载提示。checkResult=1时不为空 |
| disclaimer | String | 否 | 免责申明。checkResult=1时不为空 |
| taskTips | String | 否 | 任务提示。checkResult=1时不为空 |

{

“respCode”:” 00000”,

“respMsg”:”success”,

"respTime": 1596763523

“respData”:{

“checkResult”:1

}

}

### OTA远程下载确认

#### 说明

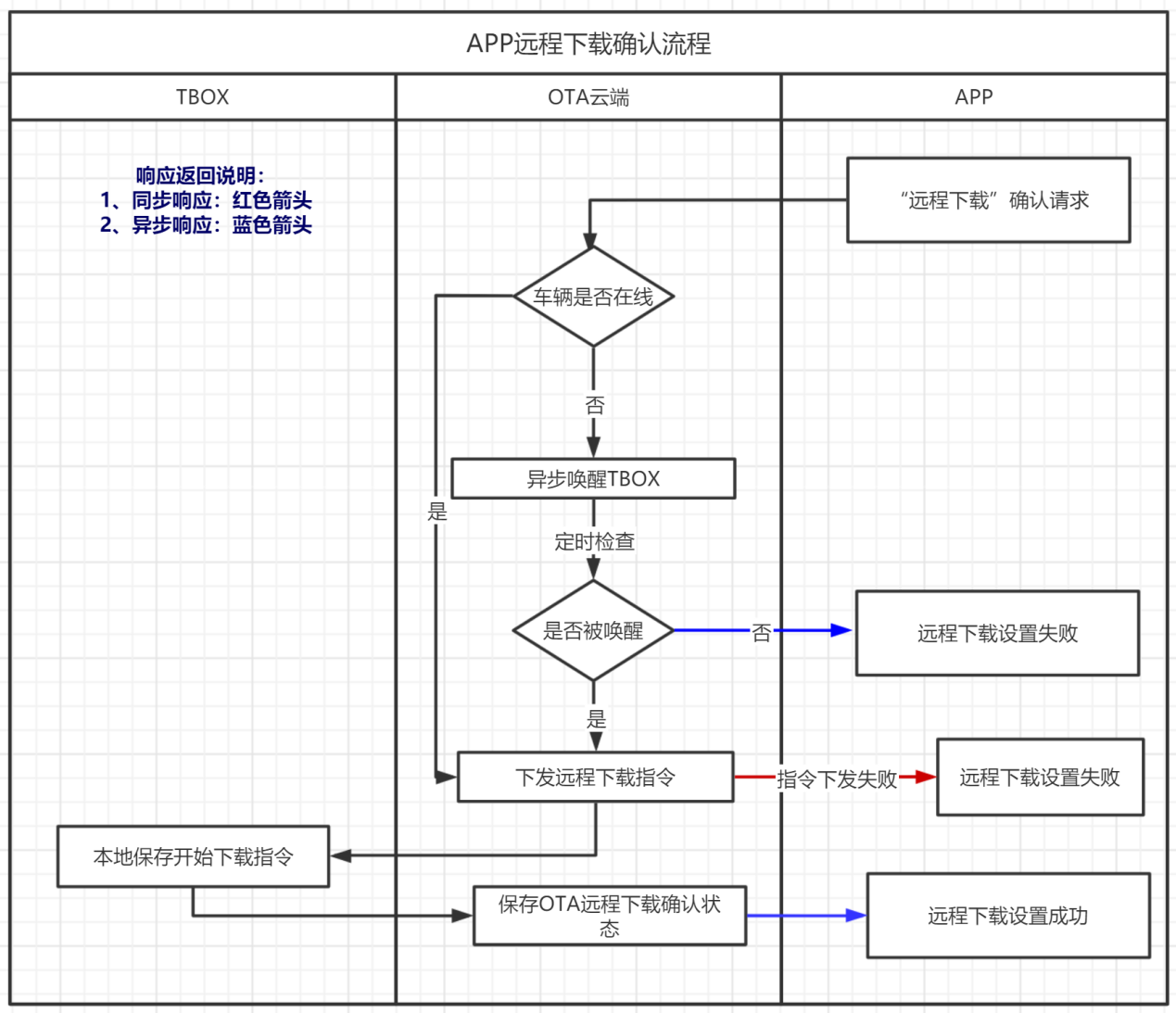
用户从手机APP通过此接口确认执行下载操作。APP将“远程下载”确认消息上行到OTA云端，OTA云端记录相关信息并转发消息给到TBOX, TBOX需要将该确认消息同步到HU。APP端需要异步等待回调响应消息。

* OTA云端指令下发失败OTA云端确认已收到指令：需要提示APP操作失败消息。
* TBOX收到该消息后，需要与OTA云端进行状态同步。成功确认的消息需要转发回APP

可能存在的异常情况：

OTA云端在业务定义的超时时间内无法收到TOBX“远程下载”确认同步状态后，会将下发异常信息同步到APP。实际情况可能是，最终在超时时间外，TOBX与OTA完成了该确认操作，会导致APP得到一个错误的提示。业务等待的超时时间需要设置一个合理值来避免此种情况。

消息流程转发示意图如下：



#### 接口url

POST {HOST}/fota/v1/app/upgrade/remoteDownload

#### 接口请求示例

{

‘’vin‘’: "bq001",

‘’deviceType ‘’: 2

}

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| vin | String | 是 | 车辆标识(一般用Vin码代替) |
| deviceType | Integer |  | 1=安卓 2=ios |

#### 响应示例

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| verifyResult | Integer | 是 | 1=需要异步等待消息回调，2= 操作失败 |

同步响应参数：

{

“respCode”:” 00000”,

“respMsg”:”success”,

"respTime": 1596763523

“respData”:{

“verifyResult”:1

}

}

### OTA安装确认

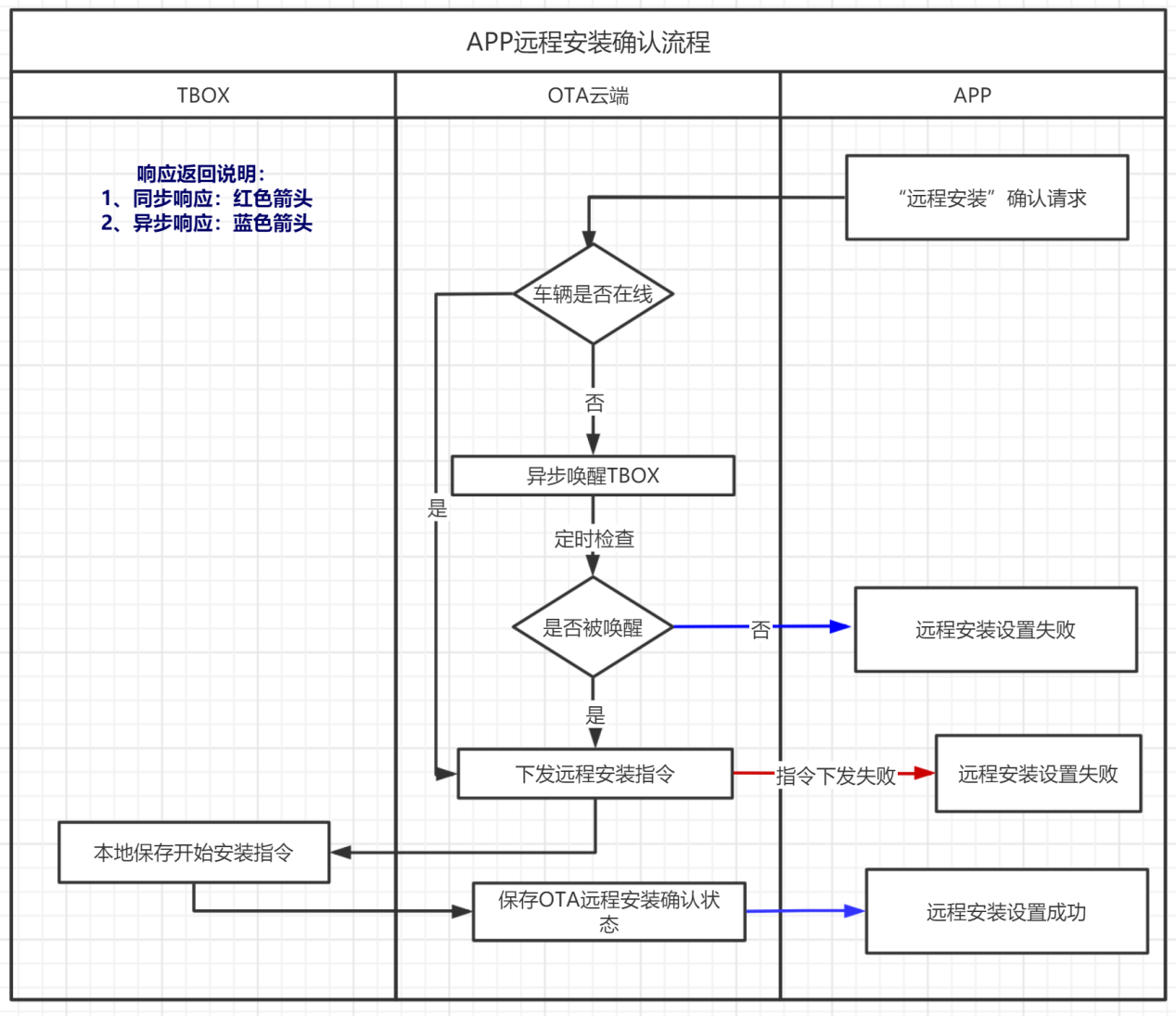
#### 说明

TBOX终端升级包下载完成后将同时向手机APP和HU发送安装确认的请求指令，以获取用户的安装确认。

该接口向手机APP用户提供升级安装确认操作接口。APP确认安装结果操作上行到OTA云端，然后通过OTA云端转发给TBox，TBOX负责推送到HU。

Tbox端收到确认后，将按确认安装的时间进行OTA软件的安装。

消息流程转发示意图如下：



#### 接口url

POST {HOST}/fota/v1/app/upgrade/remoteInstalled

#### 接口请求示例

{

‘’vin‘’: "bq001",

‘’deviceType ‘’: 2,

‘installedType‘’: 2,

‘installedTime‘’: 1000086,

‘verifyType‘’: 1

}

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| vin | String | 是 | 车辆标识(一般用Vin码代替) |
| transId | Long | 是 | 升级事务Id |
| installedType | Integer | 否 | 安装类型 ：1=立即安装；2=预约安装。verifyType=1时不能为空 |
| installedTime | Long | 否 | 预约安装软件的时间 |
| verifyType | Integer | 是 | 是否安装：1=安装，2=不安装（保留：默认为1） |
| deviceType | Integer |  | 01=安卓 02=ios |

#### 响应示例

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| verifyResult | Integer | 是 | 1=需要异步等待消息回调，2=操作失败 |

同步响应参数：

{

“respCode”:” 00000”,

“respMsg”:”success”,

"respTime": 1596763523

“respData”:{

}

}

### OTA自动下载配置同步

#### 说明

APP或HU端车辆设置上针对“自动下载”同步设置，需要同步状态给到对端(HU端/APP端)。实际该设置保存在“TSP-设备管理后台应用”系统，OTA云端内部通过条用TSP系统对应功能完成数据保存和消息转发。

#### 接口url

POST {HOST}/fota/v1/app/upgrade/autoDownload

#### 接口请求示例

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| vin | String | 是 | 车辆标识(一般用Vin码代替) |
| deviceType | Integer | 是 | 1=安卓 2=IOS系统 |
| autoDown | Integer | 是 | 自动设置选项：1=自动开启，2=关闭 |

#### 响应示例

## 面向手机App的消息推送

通过极光推送的通道推送到APP客户端

### 消息推送通用格式

* **通用消息格式**：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 响应头-序列号（来自于TBox端生成） |
| timestamp | Long | 是 | 响应头-时间戳 |
| vin | String | 是 | 响应头-车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | 响应头-消息类型：  ~~1=新版本检查结果响应（保留：相关功能由businessType=2承载）~~  2=新版本远程下载确认请求（来自TBOX）  3=新版本远程下载确认结果（来自Hu端操作发起）  4=新版本下载进度汇报响应（来自TBOX）  5=新版本远程安装确认请求（来自TBOX）  6=新版本远程安装确认结果（来自Hu端操作）  7=新版本安装进度汇报响应（来自TBOX端）  8=新版本远程安装结果汇报（来自TBOX）  9=保留暂不使用（来自TBOX）  10=前置条件检查响应（来自TBOX端） |
| **body** | Object | 否 | 请求体-（由具体协议实现，该属性是否为空由businessType决定） |

body构成说明

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| transId | Long | 否 | 升级事务Id（服务端第一次版本检查结果确认时生成），用于关联整个升级流程。当businessType不为100（升级通知主动下发）时，不能为空 |
| other | Object | 是 | 其他请求参数 |

说明：

1、businessType定义了TBox终端下发的消息类型，**body**定义了请求消息体对象，具体类型由TBox协议端实现。

2、以下具体小节中格式说明为对应的**body**消息格式

### ~~版本检查结果通知（废弃：暂时保留，相关功能由”OTA远程下载确认请求通知”承载）~~

#### 说明

向App端下发来自TBox端的版本检查请求消息，用于提示App端有新的升级版本，其中还包含：当前版本信息，~~以及OTA“自动更新”是否开启~~，根据状态显示可执行的操作。

流程参考：“6.7面向TBox消息上行”一节实现。

#### 接口url

#### 接口请求示例

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 序列号 |
| timestamp | Long | 是 | 时间戳 |
| vin | String | 是 | 车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | businessType=101 |
| body | Object | 否 | 该值为空 |
| version | String | 是 | 当前版本信息 |

body定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| checkResult | Integer | 是 | 0=无新版本，1=有新版本 |
| transId | Long | 否 | 事务序列（服务器用该字段标识升级事务流程）。  checkResult=1时不为空 |
| currentVersion | Long | 是 | 车辆当前版本号 |
| targetVersion | String | 否 | 升级目标版本号。checkResult=1时不为空 |
| planDesc | String | 否 | 升级描述。checkResult=1时不为空 |
| disclaimer | String | 否 | 免责申明。checkResult=1时不为空 |

#### 响应示例

### OTA下载确认请求（携带版本检查结果）

#### 说明

向App端下发来自TBox端的新版本升级检查结果消息，用于提示App端有车端有新的升级版本，进行后续的下载操作。该推送消息用来完成APP客户端“OTA版本检查请求”异步结果响应。

版本升级相关信息，包含：当前版本信息/可升级版本信息/升级任务描述/免责声明等~~相关信息/OTA“自动下载”开启设置（待确认）。~~

主要流程：

1. 数据保存：TOBX新版本检查完成后，向OTA云端回复ACK确认。OTA云端需要修改当前车辆对象为“待下载”状态
2. 消息转发：OTA云端转发该消息到APP客户端，提示APP客户端用户开始“远程下载”操作。

#### 接口url

#### 接口请求示例

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 响应头-序列号（TBox端产生：保留） |
| timestamp | Long | 是 | 响应头-时间戳（TBox端产生：保留） |
| vin | String | 是 | 响应头-车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | businessType=2 |
| type | Integer | 是 | 应用类型类型：车控服务=1，OTA服务=3 |
| body | Object | 否 | 响应消息主体（APP客户端开发对应解析字段content） |

body定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| checkResult | Integer | 是 | 1=有新版本，2=无新版本 |
| transId | Long | 否 | 事务序列（服务器用该字段标识升级事务流程）。  checkResult=1时不为空 |
| taskId | Long | 否 | 升级任务Id。checkResult=1时不为空 |
| currentVersion | Long | 是 | 车辆当前版本号。 |
| targetVersion | String | 否 | 升级目标版本号。checkResult=1时不为空 |
| planDesc | String | 否 | 升级描述。checkResult=1时不为空 |
| newVersionTips | String | 否 | 新版本提示。checkResult=1时不为空 |
| downloadTips | String | 否 | 下载提示。checkResult=1时不为空 |
| disclaimer | String | 否 | 免责申明。checkResult=1时不为空 |
| taskTips | String | 否 | 任务提示。checkResult=1时不为空 |

#### 响应示例

### OTA下载确认结果（TBOX 收到下载指令确认）

#### 说明

该消息来源：

1. APP客户端“远程下载”操作
2. HU端“远程下载”操作

TBOX收到下载确认指令后向OTA云端同步状态，并向App端下发升级确认结果消息。该消息需要区分确认来源（HU/APP），APP客户端需要对不同来源的确认结果消息进行不同的逻辑控制。

流程参考：

主要流程：

1. Hu或APP“远程下载”操作指令下发到TBOX后，TBOX需要向OTA云端回复一个ACK确认结果，并更新TBOX升级状态为“开始下载”。
2. OTA云端需要将该消息转发至APP客户端，APP客户端收到该消息后进入下载进度显示逻辑（确认等待loading消失）

#### 接口url

#### 接口请求示例

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 序列号（TBox端产生，用于对应“升级确认”消息） |
| timestamp | Long | 是 | 时间戳 |
| vin | String | 是 | 车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | businessType=3 |
| type | Integer | 是 | 应用类型类型：车控服务=1，OTA服务=3 |
| body | Integer | 是 | 确认操作消息体如下 |

* body定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| transId | Long | 是 | 升级事务Id（服务端第一次版本检查结果确认时生成），用于关联整个升级流程，由TBox回传 |
| taskId | Long | 是 | 升级任务Id |
| verifyResult | Integer | 是 | 确认操作结果：1=Yes（确认进行远程下载），2=No（放弃进行远程下载） |
| verifySource | Integer | 是 | 确认来源：1=Hu端，2=APP端 |

#### 响应示例

### OTA安装确认请求（远程下载完成）

#### 说明

该消息来源：TBOX端升级包下载完成，向OTA云端同步下载结果。

主要流程：

1. TBOX端升级包下载完成，向OTA云端同步下载结果。OTA云端变更TBOX升级状态
2. OTA云端转发该消息到APP客户端，提示用户进行“远程安装”操作。

流程示意图参考：

#### 接口url

#### 接口请求示例

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 序列号（TBox端产生） |
| timestamp | Long | 是 | 时间戳 |
| vin | String | 是 | 车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | businessType=5 |
| type | Integer | 是 | 应用类型类型：车控服务=1，OTA服务=3 |
| body | Object | 是 | 消息体（APP开发端解析字段对应content） |

* body定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| transId | Long | 是 | 升级事务Id（服务端第一次版本检查结果确认时生成），用于关联整个升级流程，由TBox回传 |
| taskId | Long | 是 | 升级任务Id |

#### 响应示例

### OTA安装确认结果（TBOX端收到安装确认指令）

#### 说明

消息来源：

1. 来自HU端“远程安装”操作
2. 来自APP端“远程安装”操作

主要流程：

1. TBox端收到安装确认指令后，与OTA云端同步安装确认结果ACK消息。OTA云端变更TBOX升级状态为“待安装”
2. OTA云端向APP客户端下发通知，APP客户端根据消息来源进行不同交互操作。

#### 接口url

#### 接口请求示例

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 响应头-序列号（TBox端产生） |
| timestamp | Long | 是 | 响应头-时间戳 |
| vin | String | 是 | 响应头-车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | businessType=6 |
| type | Integer | 是 | 应用类型类型：远控服务=1，OTA服务=3 |
| body | Integer | 否 | 确认操作消息体 |

* body定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| transId | Long | 是 | 升级事务Id（服务端第一次版本检查结果确认时生成），用于关联整个升级流程，由TBox回传 |
| taskId | Long | 是 | 升级任务Id |
| verifyResult | Integer | 是 | 确认操作结果：1=确认安装 |
| verifySource | Integer | 是 | 安装确认操作来源：1=HU端，2=APP端 |
| installedType | Integer | 否 | 确认安装：1=立即安装，2=预约安装 |
| installedTime | Long | 否 | 预约安装的时间 |

#### 响应示例

### OTA下载进度推送

#### 说明

消息来源：TBOX端开始升级下载后根据TBOX端固定频次推送下载进度消息，跳动频率以下载进度跳动%1上报。

下载进度状态消息类型推送到APP包括：

* 下载中
* 下载完成（验证成功）
* 下载完成（验证失败）

以下下载进度消息类型（PRD定义）不推送到APP包括：

* 下载中止
* 下载等待
* 下载完成

主要流程：

1. TBOX端上报下载进度/花费时间到OTA云端，OTA云端记录下载进度信息，并更新TBOX升级状态为“安装中”。下载进度汇报只描述整体的下载过程
2. OTA云端同步进度信息到APP客户端

#### 接口url

#### 接口请求示例

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 序列号（由TBox端生成） |
| timestamp | Long | 是 | 时间戳 |
| vin | String | 是 | 车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | businessType=4 |
| type | Integer | 是 | 应用类型类型：车控服务=1，OTA服务=3 |
| body | Object | 是 | 请求消息体 |

* body定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| transId | Long | 是 | 升级事务Id |
| taskId | Long | 是 | 升级任务Id |
| downloadProcessType | Integer | 是 | 升级过程类型：1=下载中（实时进度上报），2=下载中止，3=下载中等待，4=下载完成（保留），5=下载完成（验证成功），6=下载完成（验证失败） |
| ~~downloadPercentRate~~ | ~~Integer~~ | ~~否~~ | ~~整体下载完成百分比。downloadProcessType=1时必传~~ |
| ~~downloadFullSize~~ | ~~Integer~~ | ~~否~~ | ~~总包大小。~~  ~~downloadProcessType=1时必传~~ |
| ~~downloadFinishedSize~~ | ~~Integer~~ | ~~否~~ | ~~已完成下载包大小。~~  ~~downloadProcessType=1时必传~~ |
| errorCode | String | 否 | 错误状态码（根据实际情况定义） |
| errorMsg | String | 否 | 下载异常（中止/等待）或下载完成失败错误信息 |

#### 响应示例

### OTA安装进度推送（包括前置条件检查失败）

#### 说明

消息来源：TBOX端开始升级安装中，将各阶段状态向OTA云端上报

安装进度状态推送消息类型包括：

* 前置条件检查失败：推送检查失败消息及失败的错误提示信息到APP端
* 取消升级：来自Hu端的取消升级操作
* 安装中：根据固定频次推送安装进度消息，推送频率以安装进度跳动%1上报

以下安装进度消息类型（PRD定义）不推送：

* 升级完成
* 升级未完成
* 升级失败

主要流程：

1. 如果开始安装阶段前置条件检查失败，则OTA云端更新TBOX升级状态为“前置条件检查失败”，并同步到APP客户端，提示安装环境检查失败。
2. 如果安装阶段前置条件检查满足，TBOX端上报安装进度/花费时间到OTA云端，OTA云端记录安装进度信息，并更新TBOX升级状态为“安装中”。安装进度只描述整体的安装过程。OTA云端需要同步安装进度信息到APP客户端

#### 接口url

#### 接口请求示例

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 序列号（由TBox端生成） |
| timestamp | Long | 是 | 时间戳 |
| vin | String | 是 | 车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | businessType=7 |
| type | Integer | 是 | 应用类型类型：车控服务=1，OTA服务=3 |
| body | Object | 是 | 消息体（APP客户端开发解析content字段） |

* body定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| transId | Long | 是 | 升级事务Id（服务端第一次版本检查结果确认时生成），用于关联整个升级流程，由TBox回传 |
| taskId | Long | 是 | 升级任务Id |
| installedProcessType | Integer | 是 | 升级过程类型：1=前置条件检查失败上报，2=升级安装进度上报，3=取消升级（来自HU端取消升级操作）。 |
| totalPkgNum | Integer | 否 | 升级包总个数。  installedProcessType=2时必传 |
| currentPkgIndex | Integer | 否 | 当前第几个升级包  。installedProcessType=2时必传 |
| installedSpendTime | Integer | 否 | 整体升级花销时间（单位为秒）。  installedProcessType=2时必传 |
| installedRemainTime | Integer | 否 | 整体升级完成预估剩余时间（单位为秒）。downloadProcessType=2时必传 |
| erroCode | Integer | 否 | 错误码 |
| errorMsg | String | 否 | 异常描述消息 |

#### 响应示例

### OTA安装结果推送

#### 说明

消息来源：TBOX端开始升级安装完成，将成功/失败的结果向OTA云端上报

安装完成状态推送消息类型包括：

* 安装完成
* 安装未完成
* 安装失败

#### 接口url

#### 接口请求示例

#### 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| businessId | Long | 是 | 序列号（由TBox端生成） |
| timestamp | Long | 是 | 时间戳 |
| vin | String | 是 | 车辆标识(一般用Vin码代替) |
| businessType | Integer | 是 | businessType=8 |
| type | Integer | 是 | 应用类型类型：车控服务=1，OTA服务=3 |
| body | Object | 是 | 推送消息体（APP客户端开发对应content字段） |

* body属性定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| transId | Long | 是 | 升级事务Id |
| taskId | Long | 是 | 升级任务Id |
| installedResult | Integer | 是 | 升级结果：1=升级完成成功，2=升级未完成（升级完成，且全部回滚成功），3升级失败（升级失败，且存在回滚失败的情况） |
| errorCode | Integer | 否 | 安装失败原因错误码 |
| errorMsg | String | 否 | 安装失败提示消息 |

#### 响应示例

## OTA 运维服务

### OTA版本版本诊断