**金财管家客户端自动升级设计方案**

部门：技术平台

时间：2020年8月5日

修改记录：

|  |  |  |
| --- | --- | --- |
| 标题 | 修改人 | 修改时间 |
| 流程梳理及设计 | 蔡晓冰 | 2020-8-5 |
| 增量更新的场景分析 | 蔡晓冰 | 2020-8-6 |
| 下载进度 | 蔡晓冰 | 2020-8-6 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## 升级场景(背景)

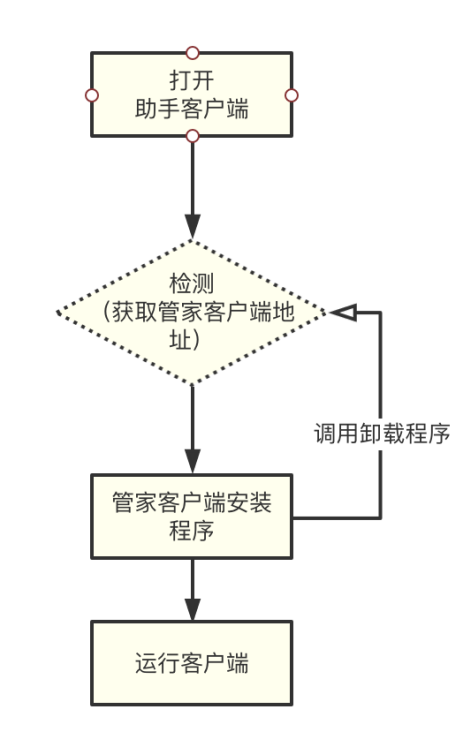
1、助手打开时，提示升级，卸载自己。升级到管家客户端。

2、安装管家客户端时，卸载助手客户端，

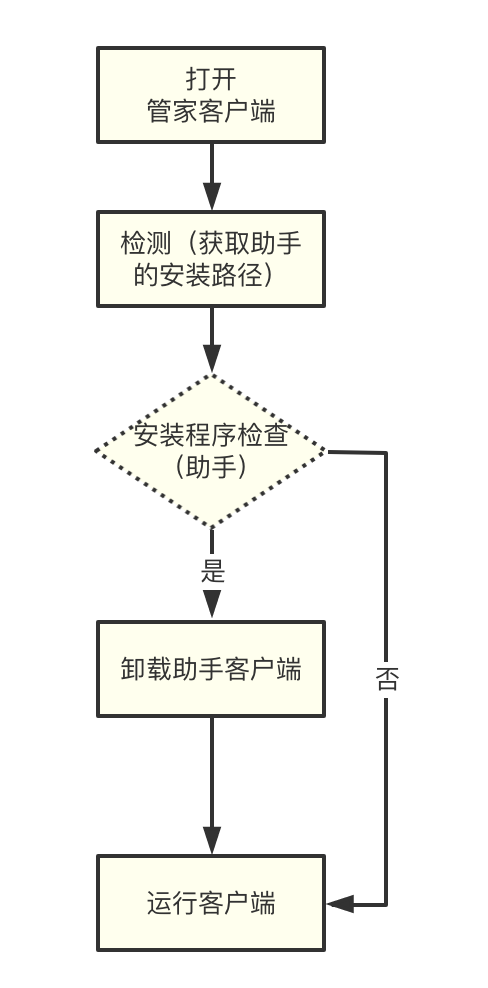
3、提供自动升级、手动升级两种机制维护自身客户端。

## 业务升级流程

1. 打开助手客户端时执行流程



1. 打开管家客户端时执行流程（已安装）



3、管家自动强制升级、手动更新业务流程

自动升级分两种场景（根据版本逻辑控制）

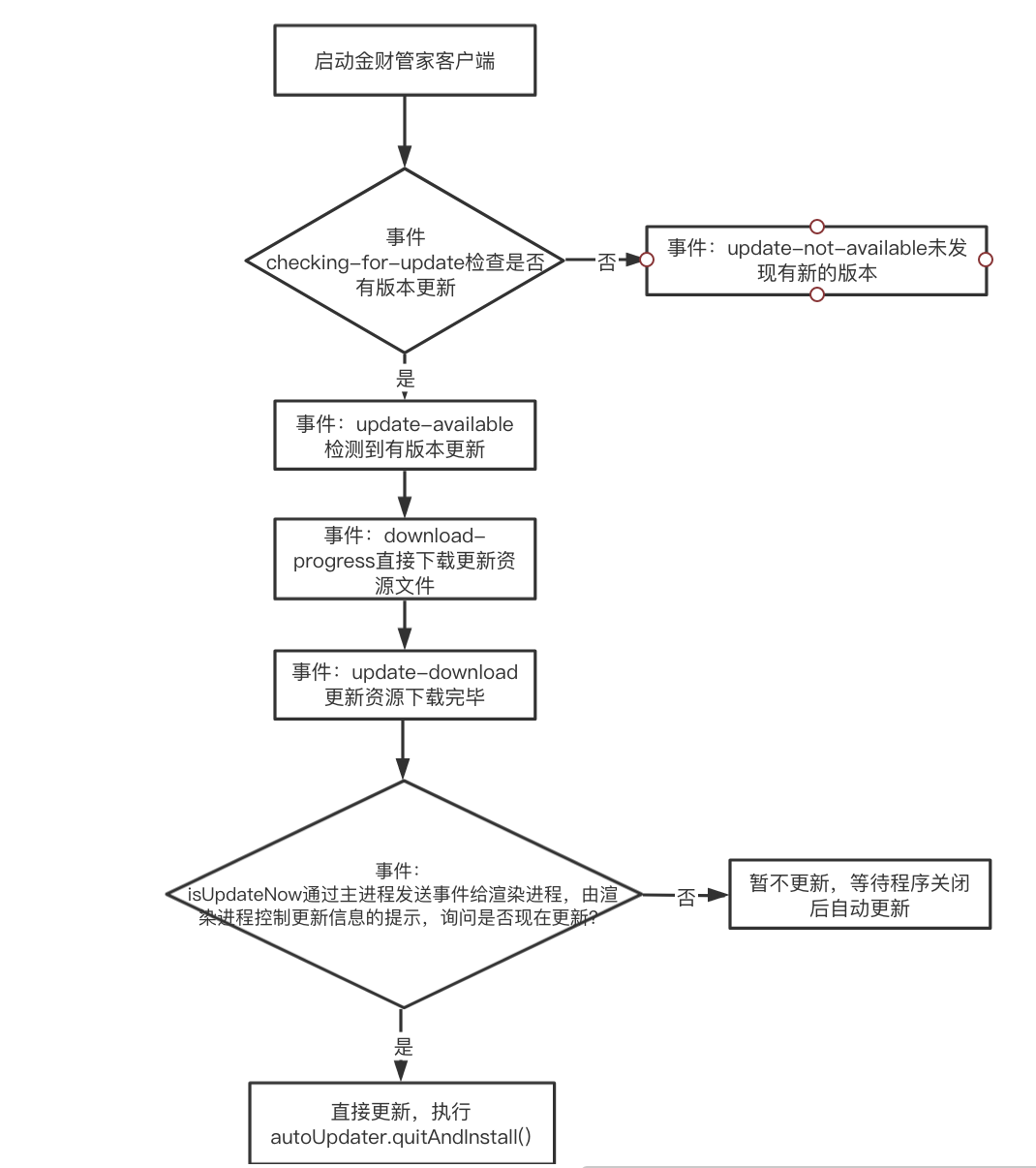
* 弱更新

如果某次更新只是功能特性的改进、小功能的上线、普通 BUG 的修复，不影响主流程，那么这被定性为一次弱更新，对用户不做强制更新。(静默升级，提前下载)

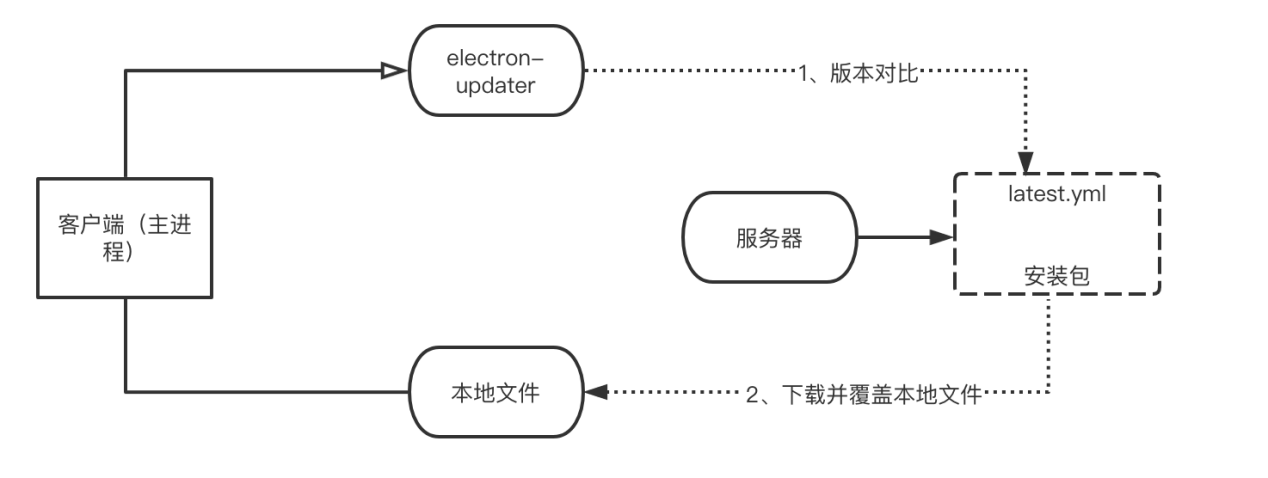
* 强升级

如果某次更新包含了重大功能的上线、重大 BUG 的修复、大的改版流程改进，可以设定为一次强更新，强制客户进行安装更新，否则客户端将无法使用。

升级主流程



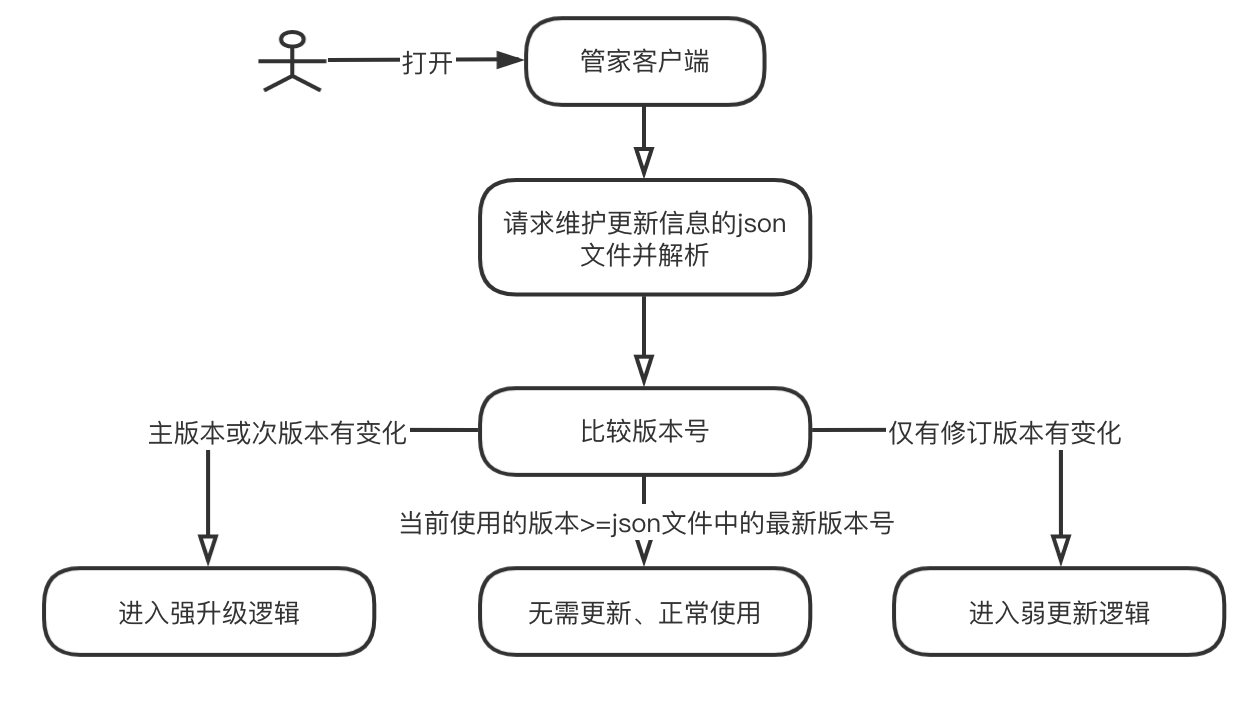
## **技术方案设计**



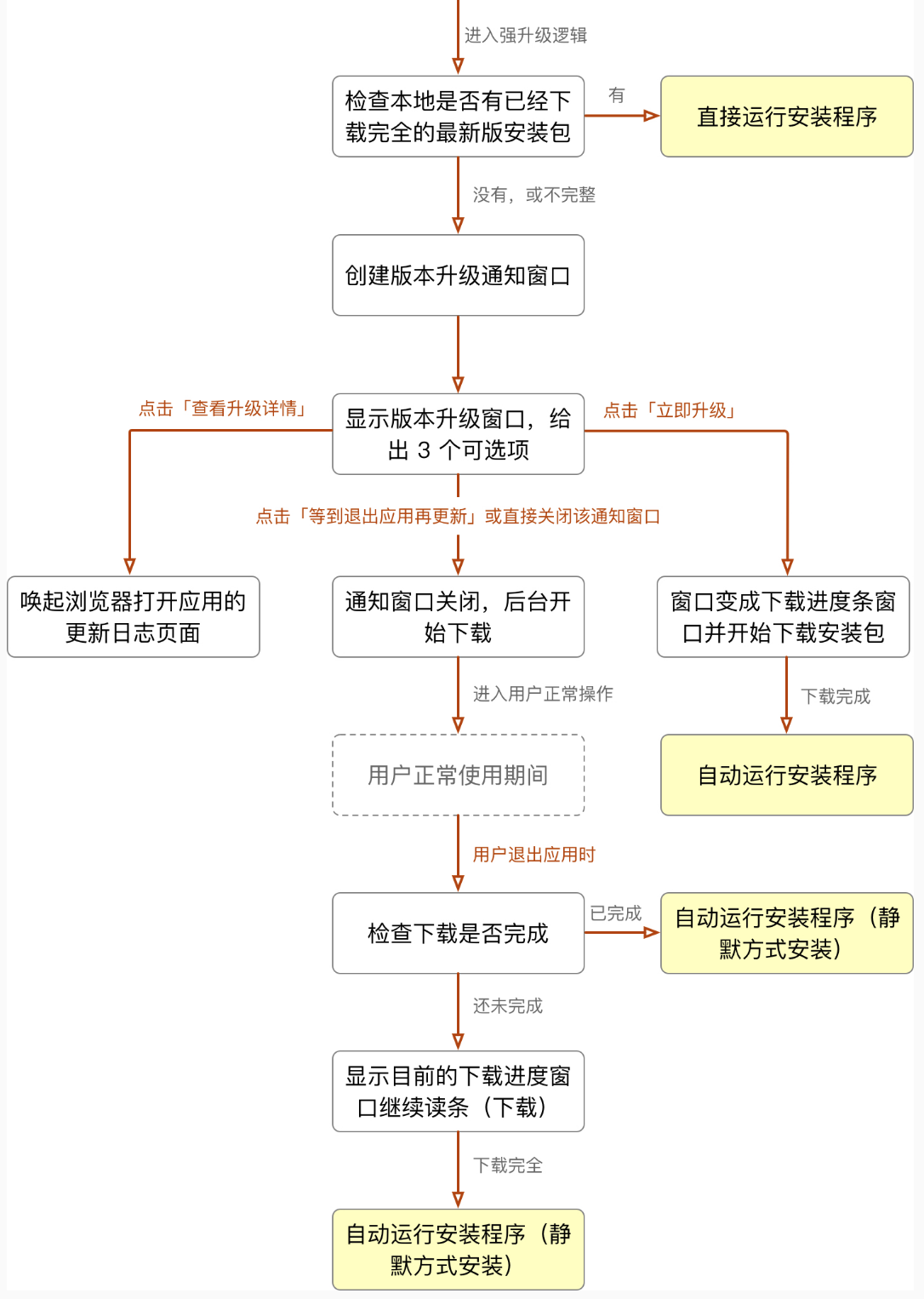
### 3.1版本对比规则

我们采用类似于「X.Y.Z」 的版本号，对比当前客户端的版本和最新版，我们可以把 X、Y 不变，仅仅 Z （修订版本号）改变的当做一次弱更新，如果对比发现，X 或 Y 有任何一个比当前的高了，那么就定义为一次强升级。这种方式的好处是不用额外的字段，也不用比对中间跨越的版本，仅仅需要对比当前的版本和最新的版本。

执行流程：



#### 3.1.1强升级逻辑

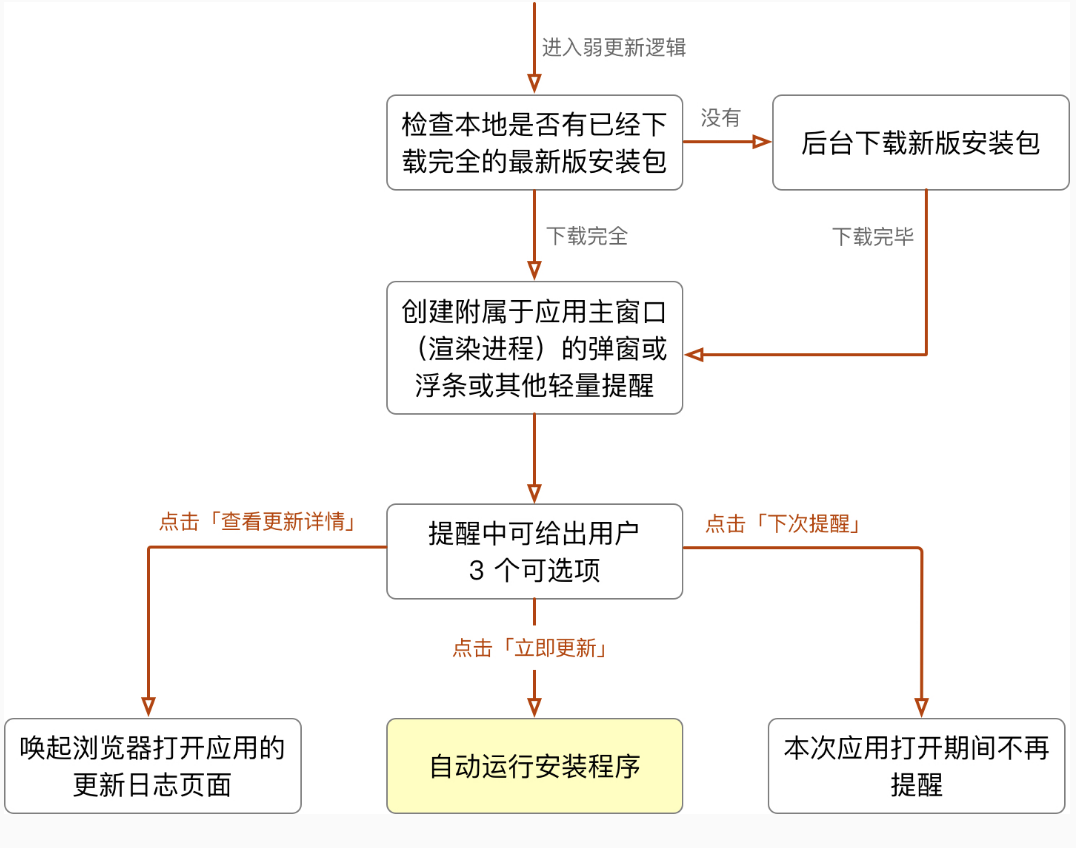


强升级下，一旦发行有可用的更新，在下载前就直接通知用户，而通知的方式是新建一个通知窗口，在 Electron 中也就是新建一个渲染进程。

用户无法选择「下次提醒」或「跳过」，而是只有立即升级和退出应用再升级两种选项。值得注意的是， 我们在对于退出应用再升级流程下的安装更新采用的是「静默的安装方式」 ，在用户看来，退出应用时，会出现一个 Windows 的 UAC 弹窗（系统级别，避免不了，至于 UAC 是什么，这里不展开），过了 UAC 之后， 之后的安装过程对用户来说是透明的，用户下次打开就是新版 。这是因为既然用户选择了退出，那么我们就认为用户不再需要看到应用安装后再启动，然后用户再关一次，而是在下一次打开再看到新版。

而非静默的安装方式，安装过程和进度对用户来说是可见的，而且安装好，应用会自动打开。

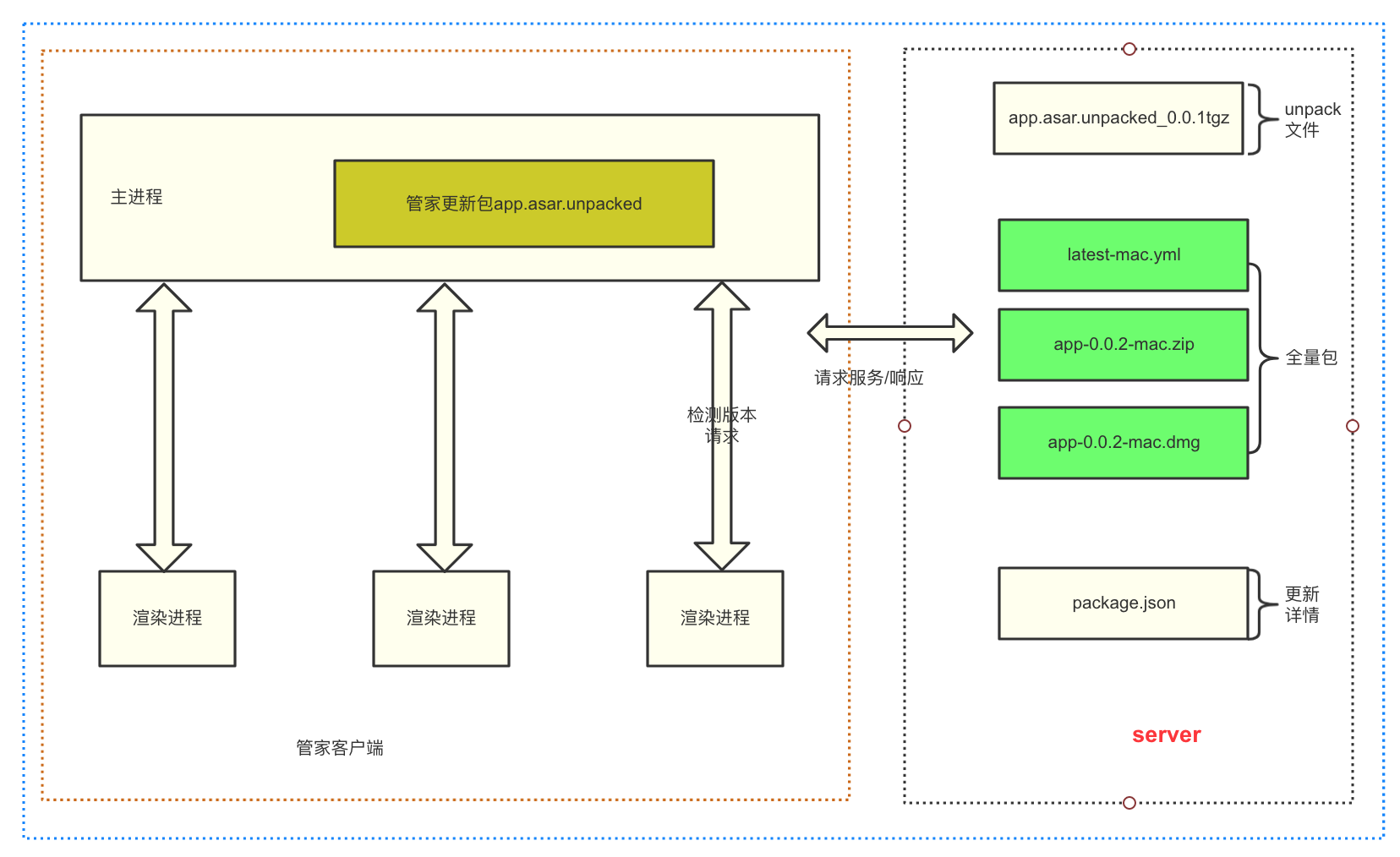
#### 3.1.2弱升级逻辑



弱更新就像我们经常能看到的更新一样，首先它不应该打扰用户的操作，所以我们选择轻量的提醒方式，实际应用中我们采用的是如 VS Code 那样的更新提醒方式，是依附于主应用窗口的浮条形式来提醒。而且我们隐去了下载过程，下载完了（即本地有下载完全的安装包）才提醒用户，用户点击更新，实际上仅仅是运行已经下载好了的安装程序，安装好了后自动打开新版应用。而下次提醒功能，就是本次应用打开期间不再提醒。

### 3.2更新步骤

1. 客户端通过定时检测、或者服务端推送方式检测是否有更新
2. 执行autoUpdater.checkForUpdates()的检测逻辑，读取资源服务器latest-mac.yml文件，对比文件hash摘要
3. 有更新则执行文件下载操作，可以配合UI层显示下载进度
4. 下载完毕之后，通知UI层并显示本次更新的相关内容
5. 应用重启进行更新



### 3.3增量更新

全量更新优点

打包配置简单，只需添加 publish 即可

代码逻辑简单，添加 autoUpdater 逻辑即可

缺点

安装包体积过大时，浪费带宽，增加用户升级时间

代码改动量小时，全量升级完全没有必要

### 3.4异常处理

下载异常、安装异常的处理流程

### 3.5 配置文件说明

发布配置、打包配置

## **升级交互设计（效果）**

**主要代码**