软件方案设计说明书

**项目名称：省流加速SDK打包回内核设计文档**

**文档修订历史**

|  |  |  |
| --- | --- | --- |
| 修订版本 | 修订作者 | 修订备注 |
| 20180228 | 张守年 | 新增 |

## 术语表【可选】

|  |  |
| --- | --- |
| 名称 | 解释 |
| 打包 | 仅仅是模块功能的编译由内核触发，同时消除UI对此模块的接口调用。不涉及具体代码逻辑的重塑。 |

**目录**

[术语表【可选】 1](#_Toc490492496)

[1 业务分析【可选】 3](#_Toc490492497)

[2 系统框架设计 4](#_Toc490492498)

[2.1 模块功能设计 5](#_Toc490492499)

[2.2 数据监控 6](#_Toc490492500)

[3 兼容性和可扩展性设计【可选】 7](#_Toc490492501)

[4 安全性设计【可选】 7](#_Toc490492502)

[5 性能设计【可选】 7](#_Toc490492503)

[6 功耗设计【可选】 7](#_Toc490492504)

[7 可测试性设计【可选】 7](#_Toc490492505)

# 业务分析【可选】

将当前独立打包的vspeedy.jar中包含的三个主要部分：

1 speedy转发，主要是在SDK中建立监听socket, 将数据请求转发给okhttp。

2 javacore管理，管理一些业务名单及代理接口功能。

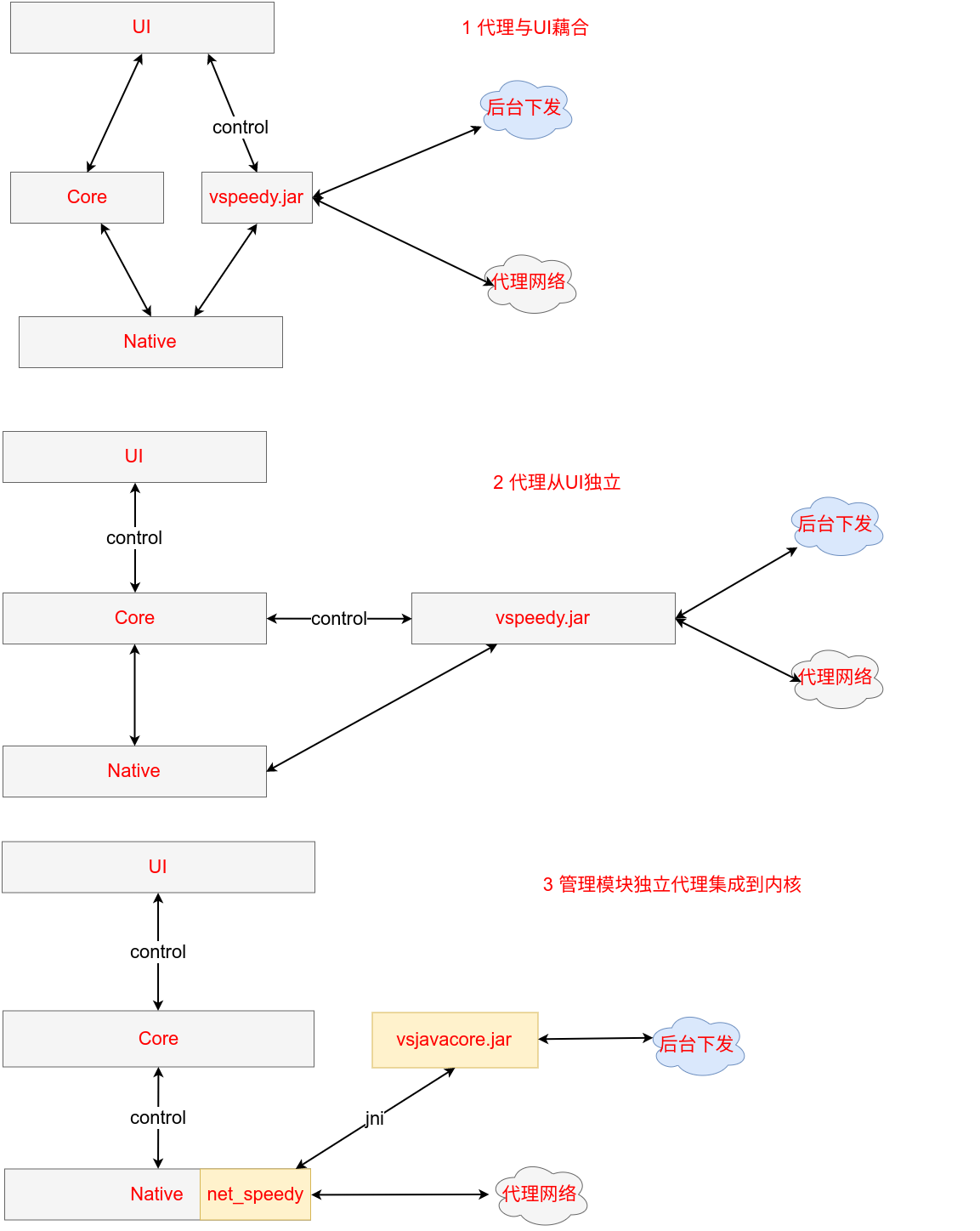
3 okhttp库，接收来自speedy模块的请求，将数据发给后台的代理IP。

终极目标是只保留javacore管理功能，speedy转发和okhttp库由于迁移到内核网络库，将进行移除。

移植分为2个阶段：

第一阶段，也就是由下图中的1 到 2的调整。

第二阶段主要是下图中 2 到 3的调整。

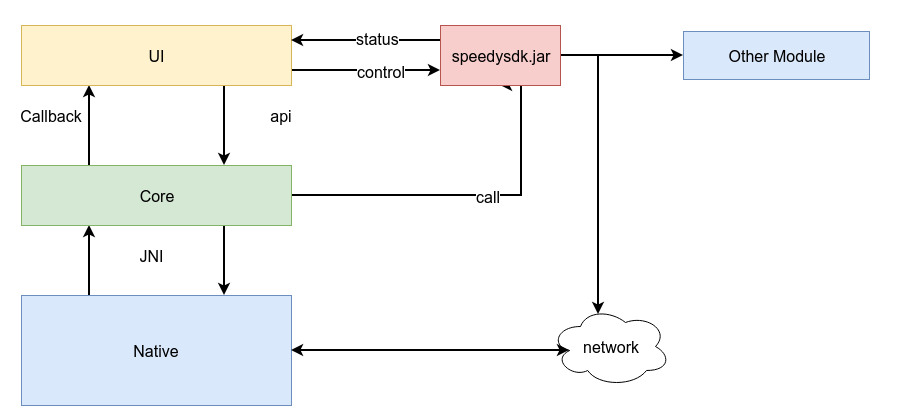


图(一) 整体移植过程图示

目前省流加速的SDK模块需要每次单独打包编译成jar，然后提交到UI的libs目录下。此功能模块本身是与内核密切绑定的，但是却需要单独打包，而且与内核的代码分别属于不同的仓库，维护起来很困难。

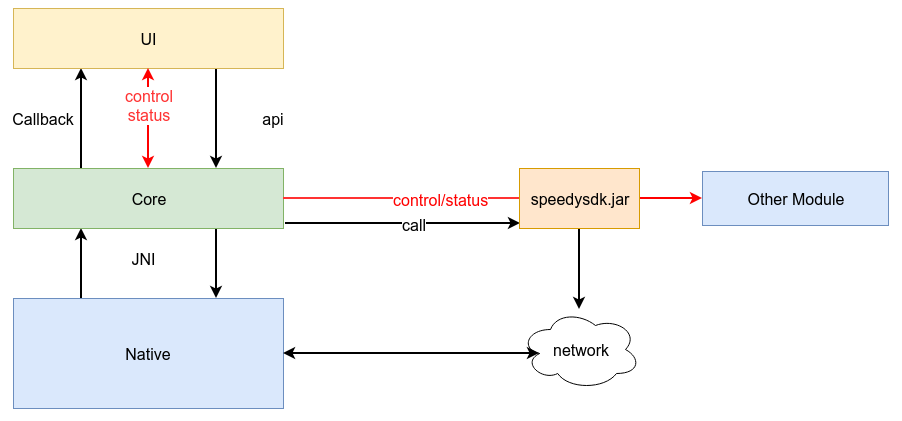
同时此模块与UI并无本质的关联关系，但是控制的接口，start/stop及对心跳包的控制都依赖于UI模拟，这在架构上是不合理的。

即下图中的status/control两条控制线，是需要给断开的。



图(二) 当前的模块依赖及交互架构

# 系统框架设计

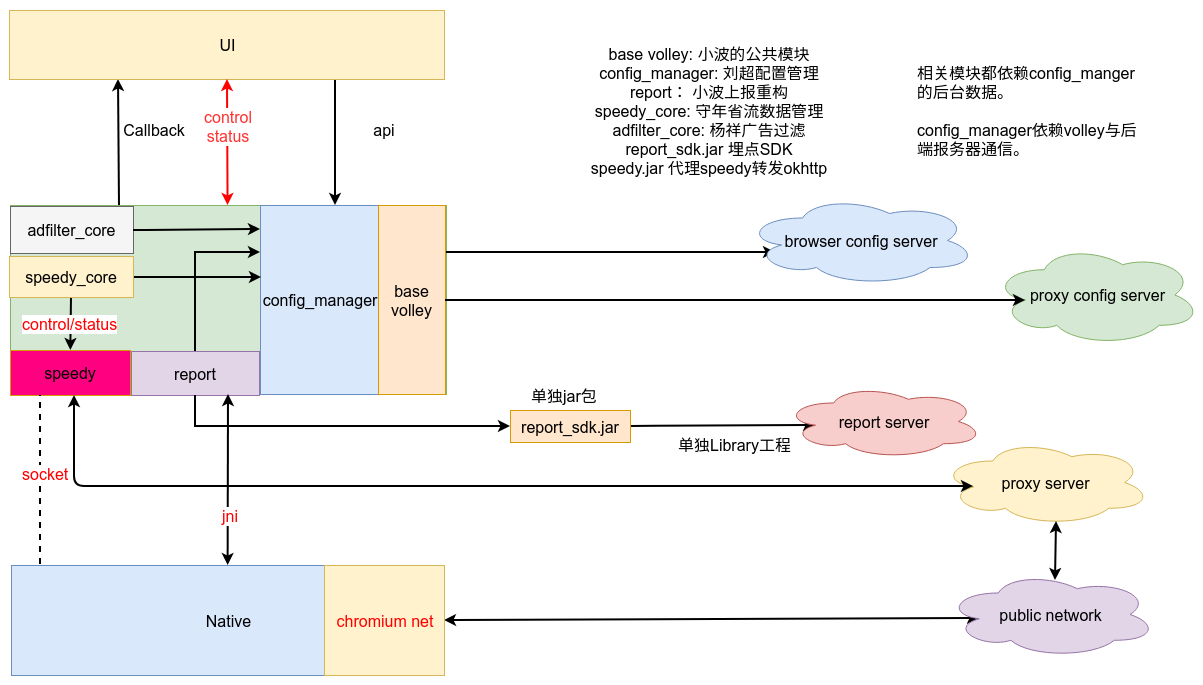


图(三) 第二阶段模块依赖及交互架构

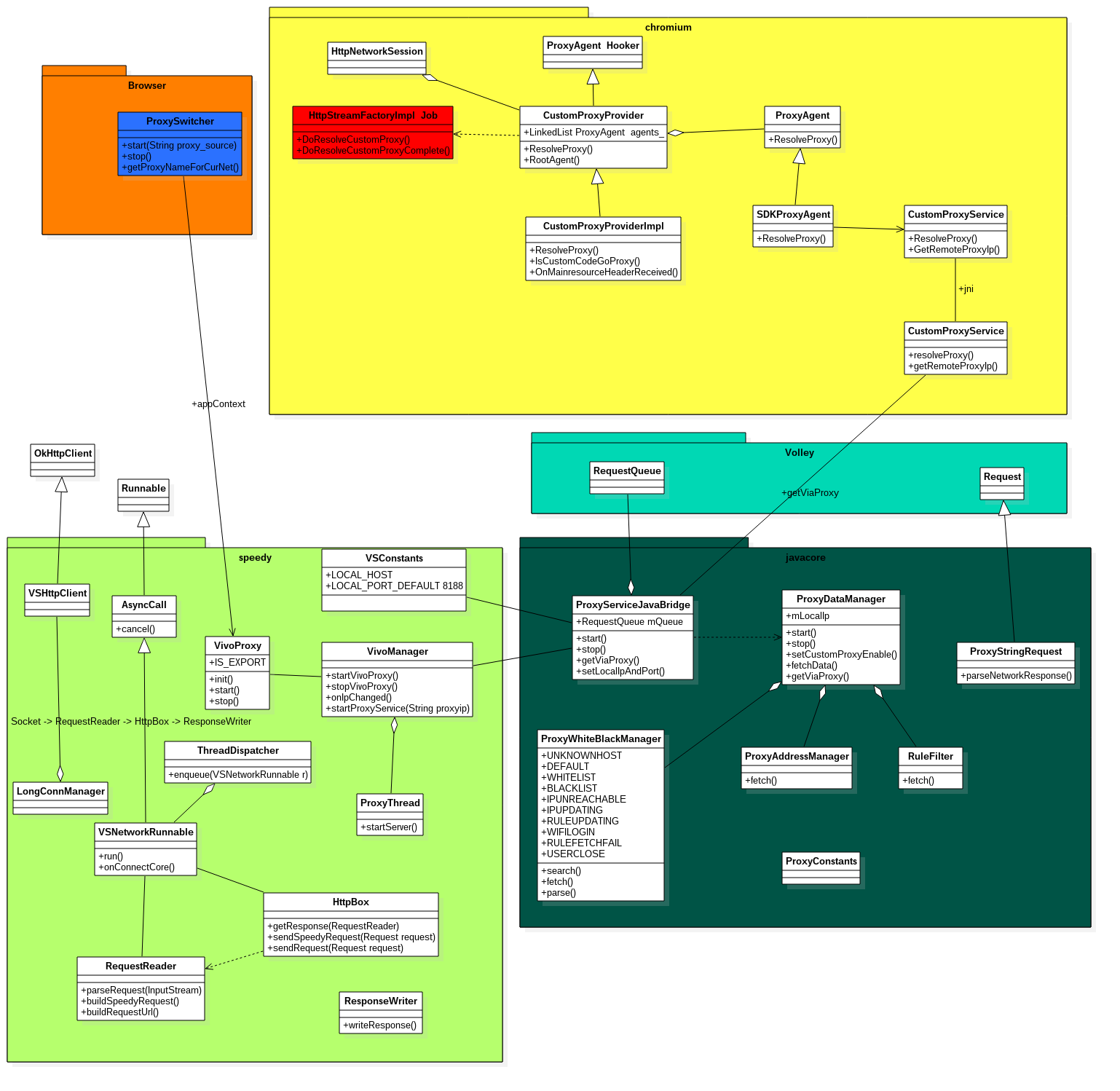
移植的主要工作就是将结构由图一变成图二，主要分为如下几个步骤：

1. 将UI的调用接口去除，确保UI功能正常。
2. UI之前调用的具体功能通过api下发给Core，由Core完成与SDK模块的交互。
3. 之前依赖的Other Module同步移植到内核（主要是与服务器的交互），对接上。
4. 将sdk的Library编译功能集成到内核的编译流程当中，使之复制内核jar包里将生成的sdk jar包复制到UI目录。

结合其他模块，整体设计图如下：



## 模块功能设计



图(三) 各模块涉及的主要交互关系

对后台服务器的网络请求主要依赖Volley库，这块小波会重构，所以最主要的就是与UI之前接口的转移和编译跟跟随内核的功能。

# 兼容性和可扩展性设计【可选】

无

# 安全性设计【可选】

无

# 性能设计【可选】

无

# 功耗设计【可选】

无

# 可测试性设计【可选】

无