软件方案设计说明书

**项目名称：省流加速逻辑调整--数据驱动模式概要设计**

**文档修订历史**

|  |  |  |
| --- | --- | --- |
| 修订版本 | 修订作者 | 修订备注 |
| 20180507 | 张守年 | 新增 |

## 术语表【可选】

|  |  |
| --- | --- |
| 名称 | 解释 |
|  |  |
|  |  |

**目录**

[术语表【可选】 1](#_Toc16161)

[1 业务分析【可选】 3](#_Toc17165)

[2 系统框架设计 4](#_Toc20544)

[3 功能设计 5](#_Toc6446)

[3.1 模块1功能设计 5](#_Toc31358)

[3.1.1逻辑流程图： 5](#_Toc14644)

[3.1.2 时序图： 6](#_Toc20561)

[3.1.3 数据存储【可选】 6](#_Toc24489)

[3.1.4依赖关系： 7](#_Toc2121)

[3.2 模块2功能设计 8](#_Toc19552)

[4 埋点设计 8](#_Toc16482)

[5 测试建议 8](#_Toc29949)

[6 兼容性和可扩展性设计 9](#_Toc28926)

[7 安全性设计 9](#_Toc3186)

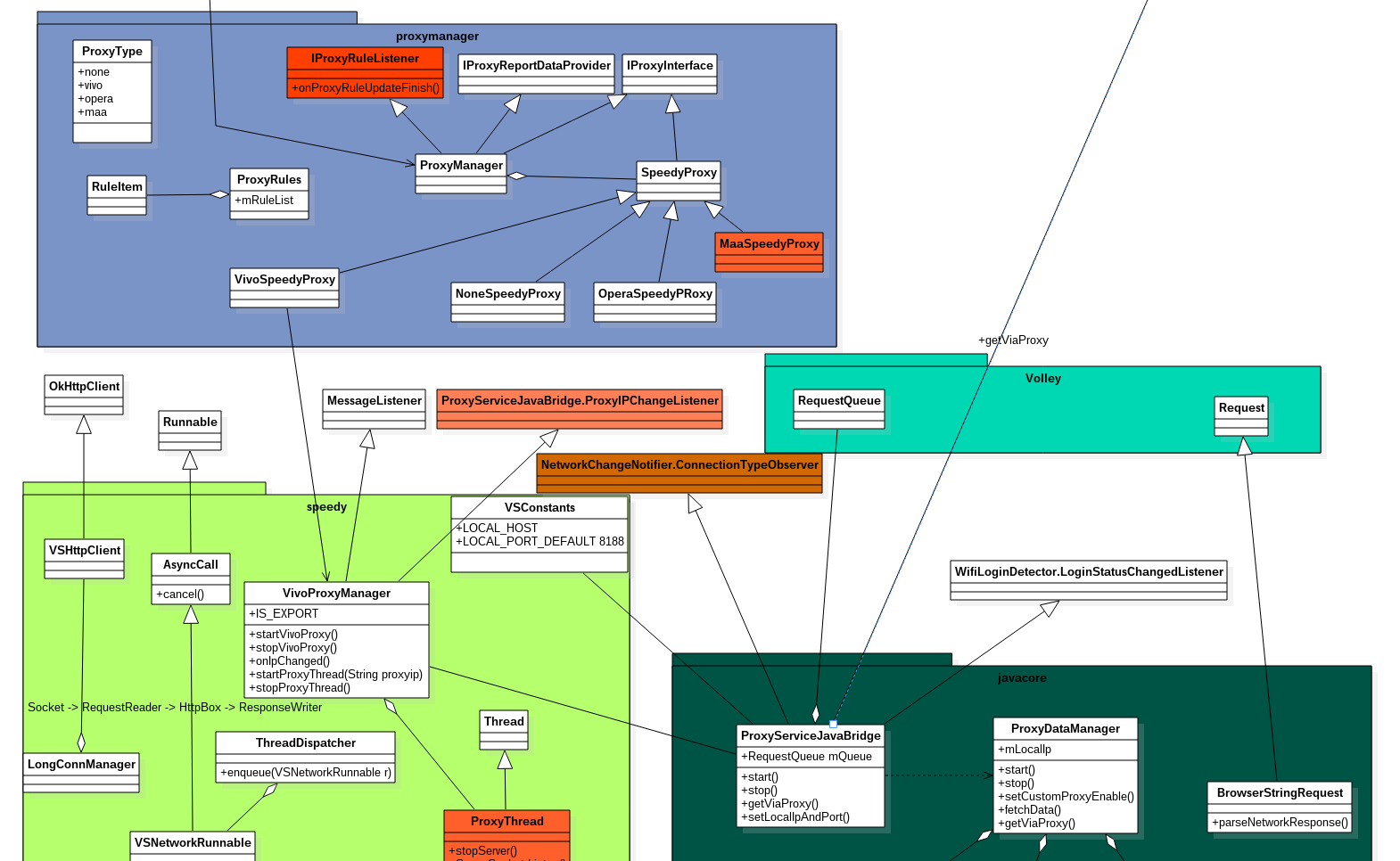
[8 性能设计【可选】 9](#_Toc13533)

[9 功耗设计【可选】 10](#_Toc4026)

[10 可测试性设计【可选】 10](#_Toc30076)

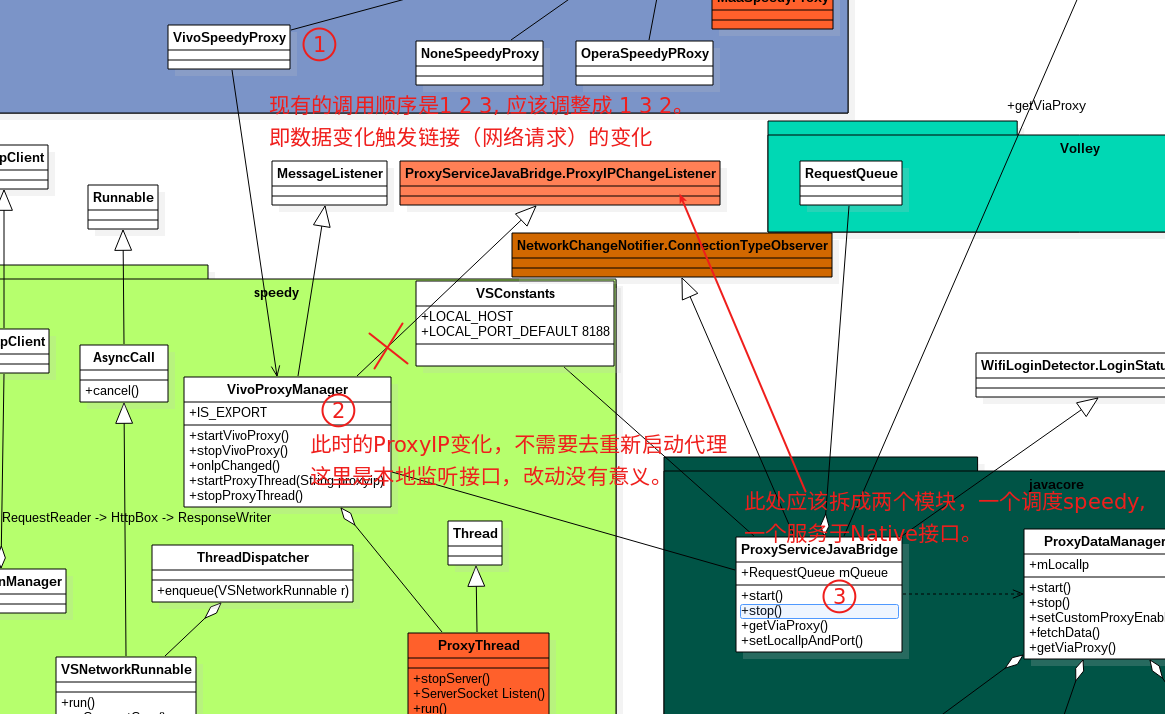
# 业务分析【可选】

现有逻辑如下：



# 系统框架设计

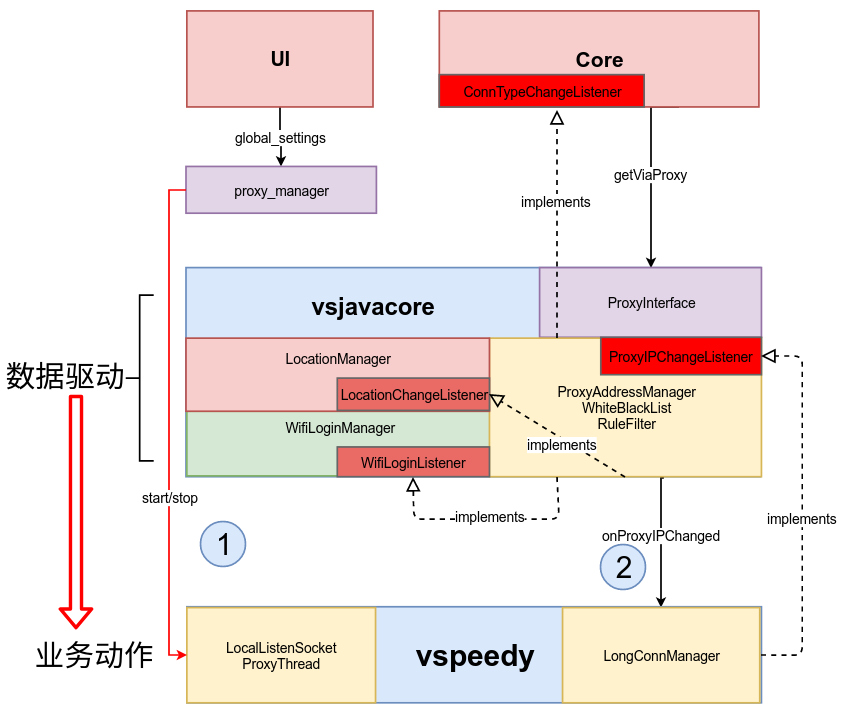
1. 省流逻辑调整和简化。主要如下描述的三个部分，对应a b c 。



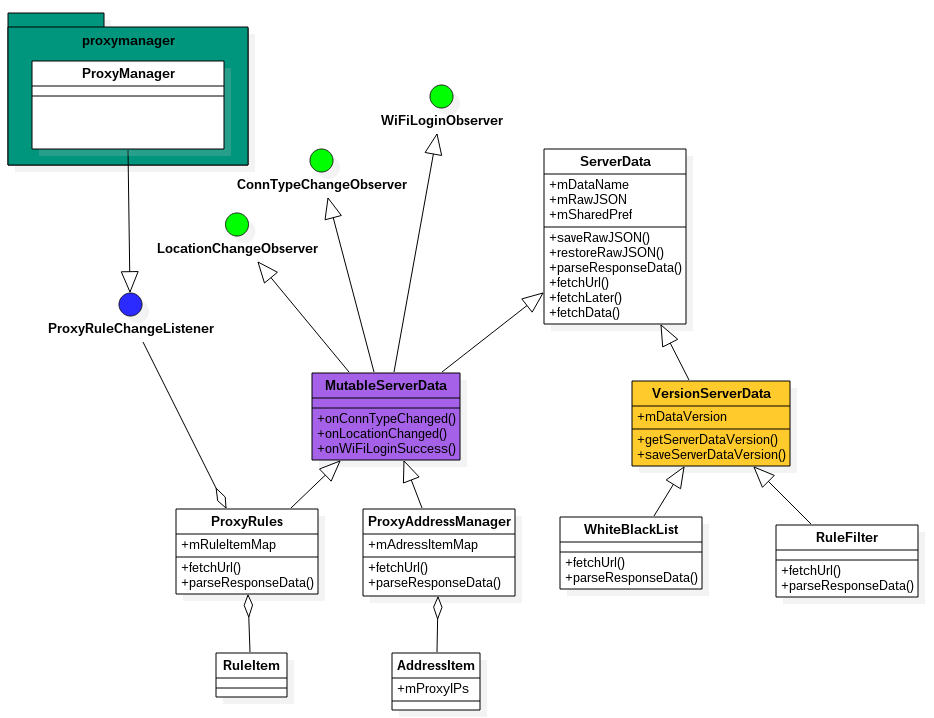
1. 1 2 3 的调用顺序业务逻辑上搞复杂了，调用到2时，再去查询确认数据，再通过回调的方式由3再通知到2, 最终实际的调用顺序是 1 2 3 2的方式。有些小的逻辑甚至是1 2 3 2 3 的逻辑，非常绕。

改成 1 3 2 由数据驱动，没有 2 和 3 之间的多余的各类数据交换，数据是单向流动的。只有 3 到 2的数据流动，即vsjavacore到vspeedy的流动，没有反向的数据流动。

如下图：



1. ProxyThread的操控，上图中的圈1, 只由最原始的Proxy\_manager下的启动和停止加速有关，与代理IP结点变化独立开。当临时取不到代理IP时，可以从内核的控制接口返回相应的数据，而不是频繁的销毁和重建监听线程。
2. 将原来的ProxyServiceJavaBridge拆分成两个功能模块，一个负责控制vspeedy 本地监听ProxyThread的启动与停止。 另一个负责与内核Core提供数据接口。原来的监听网络变化与直接的数据模块（ProxyAddressManager）去提供。所有的数据驱动主要靠红色的四个接口，来监听一些外部环境的变化，环境的变化引起数据变化，进而调整业务逻辑。除此之外来自proxy\_manager(UI)的接口和内核的数据接口是对外交互的两个源头。
3. 对原有的数据拉取和管理代码抽离公共逻辑，只将易变数据进行外部环境的监听，实现数据变动驱动业务逻辑变动。这里只有ProxyRules和ProxyAddress会主动影响代理的逻辑，其他两个数据单元都是被等其他模块来取数据。



# 功能设计

## 模块1功能设计

## 3.1.1逻辑流程图：

示例流程图

## 3.1.2 时序图：

## 3.1.3 数据存储【可选】

## 3.1.4依赖关系：

# 埋点设计

# 测试建议

【测试建议】:

1. 四类数据下发是不是正常， ProxyRules ProxyAddressManager ProxyWhiteBlackManager RuleFilter， 关注 TAG “ServerData"
2. 网络切换时，如果是第一次切换过去（无论是Wifi还是手机网络）都要拉取（ProxyRules ProxyAddressManager），如果之前曾经切过而且拉取成功，不会拉取。
3. 网络切换前后，看是不是正常走了加速，测试方法按4)操作。
4. about:core中的省流调试开关，已经设置成了持久化存储，除非手动设置，否则打开后会一直开启。Toast提示Proxy Start只在开启加速时显示，其他走加速时提示"Speedy Mode"。
5. 省流所有的日志TAG，都是以 "PROXY\_LOG\_" + 类名的方式显示， 暴问题时请附带此日志。 开启BBKLog的时候 控制台才会打印此日志。
6. 测试有Wifi认证的网络切换，确认在WIFI认证成功后才进行数据的拉取。
7. 当四类数据拉取成功后，再次以无网络方式进入，看看有没有从缓存恢复。
8. 注意一些稳定性问题，环境配置的问题。

# 兼容性和可扩展性设计

*此部分可以从以下几个方面考虑：*

*1.sdk兼容，例如：接口等设计需要兼容SDK版本；*

*2.内外销兼容，例如：功能设计需要考虑内外销版本，并通过开关等可控；*

*3.可扩展，例如：*

*框架以及业务逻辑的可扩展性设计，例如：数据是否服务器可配置修改；*

*升级后的数据不变，例如：设置项是否影响，用户数据是否丢失等；*

*升级后接口变化的处理方式，例如：GMS兼容性测试*

# 安全性设计

*此部分主要描述此软件在安全性上的设计及考虑，如：*

*网络接口请求参数加密；*

*本地敏感数据的加密；*

*第三方SDK及业务集成的控制,例如：SDK失效、移除SDK、控制SDK权限等方面的规避措施；*

*核心技术的保护；*

*移动网络下的流量控制；*

# 性能设计【可选】

*此部分主要需要说明*

*例如：*

*图片占用内存;*

*动画滑动流畅性;*

*流量损耗;*

*…*

# 功耗设计【可选】

例如：

*电量损耗情况，*

*服务常驻的必要性*

*应用是否被频繁拉起*

*…*

# 可测试性设计【可选】

例如：

*为了做白盒测试的设计考虑*

*为了做黑盒测试的设计考虑*

*为了做压力测试的设计考虑*

*为了做极限测的设计考虑*

…