**短信平台概要设计文档-v1.0**

|  |  |
| --- | --- |
| 系统名称 | SMP(SMS Marketing Platform) 短信营销平台 |
| 项目负责人 | 易红莹 |
| 作者 | 殷仁杰 |
| 文档提交日期 | 2016-09-24 |

北京票之家科技有限公司

(版权所有,翻版必究)

# 概述

SMP 是SMS Marketing Platform的缩写，即短信营销平台。利用短信与客户进行商业交流的一种营销方式。短信类型主要分为两种：验证码类型、营销类型。

业务需求对短信依赖越来越强，诸如注册验证码、订单确认短信等多种业务场景。需要提供稳定的、高到达率的短信平台支撑，为此重新搭建短信平台。

# 现短信服务问题

* 风险管控

风险管控方案中，优先考虑多通道覆盖的问题，防止某一通道网络故障影响短信发送能力。 现阶段，只接入了鸿联通道，一旦通道网络稳定性不足将严重影响公司业务的稳定性。

* 安全隐患

未对手机号做流量、频次的限制，极端情况会出现短时间内向同一号码发送大量短信，投诉 率升高。

* 设计缺失

无设计可言，只是满足短信发送功能。对短信量、成功率、对账等功能完全缺失。

# 业务架构

鉴于现有短信服务能力严重不足，决定重新构建新一代的SMP平台。

SMP平台大体由三部分组成：生产平台、发送平台和统一管理平台组成。生产平台负责生成短信发送任务，发送平台负责将短信任务发送到最优通道。

生产平台和发送平台间使用Redis优先级队列传递短信任务。基于操作简单，学习成本低及支持较复杂优先级队列等特点，优先选择Redis作为队列使用。

统一管理平台主要对短信生产、发送任务的统一调度。包括模板管理、黑名单管理、退订管理等。

SMP的业务架构图如下：

## 生产平台

生产平台对接各端业务系统的MQ消息或远程RPC调用生成短信发送任务，按照业务线提前约定的任务优先级规则将短信内容发送到不同任务队列中。

短信任务生成流程中，需要对手机号码是否设置了黑名单、对手机的流量检测控制。一旦不符合任务生成规则则记录详细的错误信息标注出来并记录到待人工发送任务列表中。由人工决定是否需要发送。

### 黑名单过滤

对手机号设置黑名单，加入到黑名单的手机号将不进行任务发送。生成的短信记录到待人工发送任务列表中。

### 流控过滤

现阶段提供粗粒度的流量控制，限定同一手机号每天在同一通道下的发送最大量。超过阈值后，短信平台选择其他通道进行发送。

### 模板渲染

每条业务线使用短信平台之前，需要先提供该业务线的短信模板及该短信的优先级。人工将模板导入到短信平台中。

需要为每个模板指定一个模板编号，发送短信任务使用模板编号进行发送，由短信平台根据模板编号做匹配并替换模板内容。

模板渲染失败，则记录短信到待人工发送任务列表，方便业务系统技术人员排查问题。

## 发送平台

负责从任务队列实时拉取任务并进行短信投递。根据投递结果做成功率分析，智能计算出通道的稳定性。

一旦某个通道稳定性不足或发送失败率过高，则智能切换其他的通道进行投递。

负责任务拉取监听器的维护，监听器按业务线+队列优先级维度进行监听，需要维护多份不同业务线维度的监听器。

### 任务拉取

每条业务线有一个拉取监听器，负责从该业务线的不同优先级队列中拉取短信任务。监听器只负责拉取，具体的短信投递工作交由短信投递模块进行投递处理。

### 短信投递

接收拉取监听器的短信任务，实时计算通道的稳定性并进行真实的短信投递工作。调用通道指定的URL，将短信发送出去。接收通道返回值，成功情况下将返回信息交给统计模块负责统计分析。

### 智能切换

为防止短信通道的不稳定导致短信投递失败的情况，短信平台需要能只能切换通道，选择最优的通道进行投递任务。

智能切换主要有两块：投递之前和投递失败之后。

投递之前由短信投递模块实时计算出哪个通道最优并选择该通道投递

投递失败之后，短信投递模块需要重新选择通道并重发一次。

## 管理平台

### 模板管理

各业务线使用短信平台之前，需要向短信平台申请发送权限。提供所有的短信模板及模板优先级。

业务端向短信平台申请发送权限，并提供详细的短信模板及各模板的优先级。由短信平台负责录入模板并匹配每种模板的编号，并输出给申请人。

业务端在进行短信发送时，根据模板的编号进行发送。不匹配的编号将不予接收。

基于约定大于配置的原则，模板编号由短信平台负责分配，目的就是保持格式统一，规范化。

编号生成规则：

编号总长度为5+1+2+2 = 11，每段之间通过下划线进行分割。

已交易系统模板编号为例：26个英文字母足够满足优先级规则了。

高优先级的有：trade\_A\_01、trade\_A\_02

中优先级的有：trade\_B\_01、trade\_B\_02

低优先级的有：trade\_C\_01、trade\_C\_02

### 黑名单管理

现阶段通过手工配置的方式设置黑名单，后期可根据数据分析自动设置黑名单。

黑名单是限定手机号码是否接收短信使用，设置了黑名单则不对手机号进行真实的短信投递，但需要生成短信内容。

一个手机号只能设置一次，重新设置不起作用。

### 通道管理

维护公司签订的短信通道运营商。

和通道运营商签订合作协议并做技术对接完成时，将通道录入到短信平台。

一旦合作终止或极端情况出现费用不足临时下线情况，可手动操作将该通道进行临时性下架处理。下架后的通道将不再进行短信投放，改由其他通道。

### 退订管理

暂不实现

# 技术方案

该部分描述SMP系统整体技术实现方案及实现细节。

## 技术架构

生产平台负责接入业务端系统，支持RPC及MQ消息方式。发送平台负责和第三方通道打交道，负责发送短信内容。生产平台和发送平台之间使用Redis队列传递短信任务。基于MySQL数据库持久化短信内容及统计信息。

短信平台采用Java语言开发设计，前端采用Bootstrap渲染页面，后端服务采用SpringMVC、Dubbo。基于RocketMQ做异步消息，Redis做消息队列，采用MySQL做数据存储。

## 技术选型分析

短信平台以Java语言开发设计，具体各模块技术选型规则如下：

消息队列：采用RocketMQ消息中间件，主要考虑技术栈的一致性及中间件本身适用范围决定。公司已经搭建RocketMQ消息服务器且运行良好，为保持统一采用RocketMQ作为消息队列。

优先级队列：采用Redis，主要考虑Redis简单，学习成本低。最主要的，开源社区中的多种消息队列并不能很好的满足复杂优先级的队列需求，而Redis原生支持队列功能。另外，现阶段短信平台应从简单角度入手满足业务需求为主，所以对技术的要求尽量简单适用。

流量控制：同一手机号在某一通道下流量限制将采用Redis的计数器接口进行设计。

数据库：采用MySQL作为数据存储介质，永久记录短信发送信息。

前端技术：建议采用Bootstrap，不会前端开发，不做过多分析。可讨论其他方案。

## 关键节点描述

### 优先级队列

采用Redis实现优先级队列，主要因为接入简单、学习成本低；基于List数据结构blpop、brpop接口可方便实现优先级设置。采用Redis分片架构实现，一是避免Redis单点故障，二是将多种优先级队列分散到不同分片，避免出现数据集中在某一个分片而形成热点分片。

优先级队列将按业务线+优先级的维度进行设计，参考短信模板编号的生成规则。现阶段优先级只分为三种：A（高）、B（中）、C（低）。

最终确定不同业务线将分为三个不同优先级的队列。比如:trade\_A\_queue、trade\_B\_queue、trade\_C\_queue。

一种业务线一个消费端。所以一个消费者需要同时消费三个不同级别的队列数据。采用Redis提供的List数据结构及lpush和brpop接口实现优先级队列的消费。

伪代码：

while(true){

List<String> values = jedis.brpop(“trade\_A\_queue”,”trade\_B\_queue”,” trade\_C\_queue”);

doHandler(values);

}

doHandler(List<String>);//真正进行投递任务

根据以上brpop接口可以获知，当trade\_A\_queue中无数据则获取trade\_B\_queue中的数据，以此类推可模拟实现优先级的队列。

采用这种方式，适合优先级别不多的情况，当级别数量不确定情况下不适用。

### 流量控制

从安全角度考虑，如果一段时间内对同一手机号发送过多短信可能会出现投诉率升高及短信通道认为是垃圾短信而将通道关停的风险。

为避免这些问题，需要对手机号做限流处理。同一手机号在同一通道下，一段时间内限制最大发送量。超过该阈值，短信将暂停发送。

基于Redis计数器实现按天流量控制。约定每个通道每天对同一手机号发送总阈值，计数器过期时间根据当天第一次发送到当天23:59:59秒计算获得。若超过阈值则顺序选择另一通道发送，极端情况下所有通道都不可用时，将短信记录待发送列表由管理平台手动触发。

计数器Key由四部分组成：固定前缀+两位日期+通道名简写+手机号。

例如：手机号13522330020在鸿联通道的9月25号的统计量：sms:25:hl:13522330020

各部分以英文冒号分割，别问我为什么，自己思考。

### 重发

重发是短信平台非常重要的一个隐性需求，因某一通道网络不稳或超流量阈值而导致发送失败的情况，需要运营人员手动执行发送动作。

重发分为两种：程序自动重试发送；运营手动触发强制发送。这里主要描述的是第二种。

短信因各种原因最终发送失败情况下，记录失败的原因及短信内容到“待发送列表”，提供手动触发按钮进行强制发送。强制发送将不受到流量阈值、黑名单等限制，直接调用发送平台进行发送。

## 模块设计

### 短信生产流程

生产平台提供RPC或MQ消息接收业务系统的短信发送请求，根据模板编号、手机号进行过滤控制短信任务的生成。

流程图



从流程图中可以看出，并未对黑名单、限流等做限制要求，出于两点考虑：

生产平台只负责生产短信，任务功能最小化，边界划分清晰。短信记录是否应该被投递，不是短信生产平台决定，而是由短信消费方决定。房产开发商只管负责盖房，购房人买后住不住是购房人的事情，跟开发商无关。

短信记录有记录可循，发送平台发送过程中失败或被黑名单、限流等限制条件过滤，该短信记录也都应该被记录下来，方便开发人员排查或手动进行重发操作。



### 短信发送流程

发送平台从优先级队列拉取短信任务，根据黑名单、限流等限制条件进行过滤，并选择合适的短信通道最终发起投递任务。

短信发送流程中，无论何种原因导致无法投递或投递失败，都需要将详细的错误信息记录下来，方便问题排查及人工进行投递。

 从流程图中看，对手机号黑名单、流量做了限制。未通过限制条件，将失败的原因记录下来，交由人工触发。

由于流量限制粒度为通道级别，所以某一手机号在一个通道已超最大阈值，可能在另外通道还未超过，所以在限流验证流程中加入了多次重试机制。

短信投递流程也加入了重试机制，当某一通道网络不稳定等情况下，重试其他通道。



## 短信通道账户余额设计方案

### 短信通道余额涉及应用场景

~~1、发送短信时需要判断通道余额是否足够发送短信~~

2、统计各个通道根据现有余额计算通道可发送短信条数，以及后续对账时会定时获取阶段性通道余额并记录入库

3、定时调度获取通道余额，根据通道余额阈值通知财务或技术后续相关操作流程（充值或者禁用通道或者修改通道筛选指标）

4、前端模块提供web端查询各通道实时余额

### 具体实现思路

~~账户余额限制发送：调度任务会根据一定频率去通道获取通道最新余额，并将获取的余额存入缓存，在发送短信时判断通道余额是否足够，如果足够才选择该通道进行投递。~~

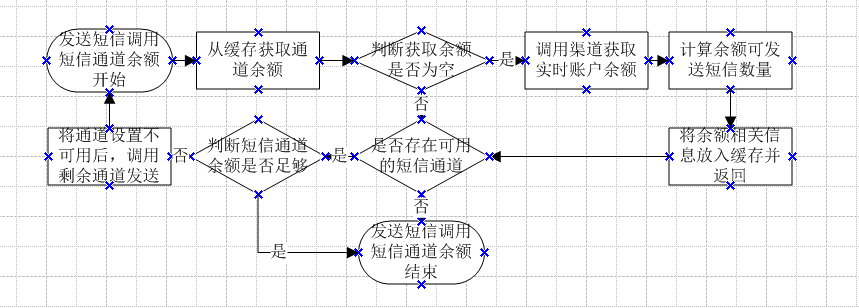
~~注意问题：1、余额是一定频率获取的，可能不是最新的余额，可能调用投递过程中还会出现余额不足的情况，还是的考虑通道余额不足的问题。~~

~~2、这里会涉及一些计算，根据通道余额转换成通道可发送短信条数，并根据试试投递短信内容计算短信数量~~

Web端查询账户余额：通道那边限制查询余额频率不能在一定时间内连续请求，否则会锁定账号。基于此前端请求查询余额时会根据通道设置限时访问，这样就会防止通道账户被锁定，也可以查询通道实时余额。

余额阈值预警：当通道余额达到低优先级阈值就修改筛选通道优先级别等级；当通道余额到达中优先级阈值会根据配置往指定账号发邮箱以及短信通知；当余额到更高级别的阈值，会根据配置将通道关闭等相关一系列操作。

### ~~发送短信判断通道账户余额流程~~



### 相关表结构设计

修改通道用户表，新增通道余额字段（钱单位用分）

## 状态报告设计方案

### 涉及应用场景

1. 后期做统计时可以用来做数据分析依据，计算通道的稳定性，成功到达率，成功接收短信时间分析，手机号接收短信统计等等。
2. 根据状态报告按通道时间节点或者手机号统计短信发送情况，设置短信通道优先级别指标
3. 后期做对账功能，会用到状态报告数据分析短信平台发送短信资费情况，以及统计供应商某个阶段发送短信产生费用，以及整个平台各个维度短信具体成功处理后的费用统计。

### 具体实现思路

状态报告的获取：后台会配置一个定时任务，在系统任务不是很繁忙的时间段去通道那边批量拉取状态报告数据解析入库，定时拉取可以按天去通道那边获取，时间节点暂时定位凌晨1点到凌晨5点。

通道稳定性以及到达率：通道发送后返回结果与状态报告接收短信成功的情况汇总统计，通道的到达率，以及通道稳定性。

通道优先级指标：根据阶段性统计每个通道的到达率和稳定性综合计算通道的优先级别，设置后的优先级别会在后续发送短信中通道筛选过程中会优先级更高。

对账功能：根据短信平台统计的短信发送成功的资费情况，与通道那边拉取的状态报告资费进行账务比对，进行财务账务汇总以及资金流计算。

### 相关表结构设计

主键，通道id，通道唯一标识，手机号，状态报告内容，响应时间，状态报告详情，创建时间（表加注释）

| 逻辑表名 | 状态报告表 | | 物理表名 | status\_report |
| --- | --- | --- | --- | --- |
| 主键 | id | | 索引 | channel\_id, response\_time |
| 字段名 | 数据类型 | 空值 | 默认值 | 描述 |
| id | bigint(20) | N |  | 状态报告ID |
| channel\_id | bigint(20) | N | # | 通道id |
| linkid | varchar(20) | N | 1 | 调用通道唯一id |
| phone\_number | char(20) | N |  | 手机号 |
| response\_time | timestamp | N |  | 状态报告响应时间 |
| content | varchar(100) | N | # | 状态报告内容 |
| detail\_result | varchar(1000) | Y | 1 | 调用通道返回状态报告详细结果 |
| create\_time | timestamp | N | CURRENT\_TIMESTAMP | 创建时间 |

## 自测方案

测试主要是从RPC接口接收任务、从MQ接收短信任务消息、对Redis队列、以及定时任务进行测试。

### 测试从MQ接收短信任务消息

生产平台的短信入口有一个是从MQ接收短信任务，需要对这个接收做测试。主要测试的内容是当MQ上有积压消息时如果处理，是继续发送还是丢弃。

消息头里面设置消息过期时间、消息模型

### 测试Redis队列

短信Redis队列的开发，主要包括队列初始化、入列数据、出列数据。首先要做的是这些基本功能的测试，要满足最基本的要求：根据业务线能生产正确的队列，入列和出列能按正确的优先级处理。

接下来在基本功能的上增加更多多功能的测试。

将生产平台、发送平台的功能包含上一起测试。其中通道发送短信可sleep上一些时间模拟发送。分另在和多线程下测试。除了测试基本功能时的要求，还要满足其它要求：

* 短信任务都要进入正确的队列；
* 生产平台产生的任务都会被发送平台处理，不能丢失，并能记录到数据库。
* 生产平台和发送平台的相同时间内的处理速度不能慢（需要定一个数值），处理量要大致相当，保证不积压消息。
* 当产生积压消息时，要做好后续措施。如果积压时间较长，失去了发送的意义，则废弃。

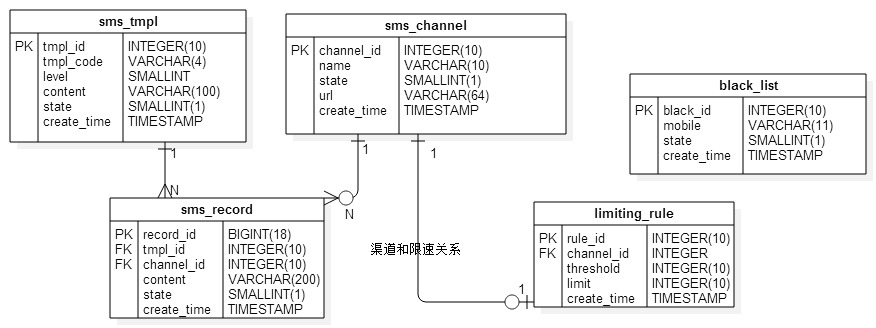
可从以下方式测试：

* 生产和发送都是单线程。
* 生产和发送都是多线程。
* 多台机器同时测试。
* 多个业务线、相同业务线的多个个优先级同时大量的测试。
* 发送执行中，生产先停止一段时间，再启动。

# 数据库设计

短信平台申请单独数据库资源，库名: sms

分表设计：在此只是简单提及一下，根据业务需求短信量会【】非常的庞大，但短信记录又不能丢失，建议采用时间分片机制实现sms\_record表的分片，前期就考虑进去避免后期再定期进行数据迁移。



| 逻辑表名 | 短信模板表 | | 物理表名 | sms\_tmpl |
| --- | --- | --- | --- | --- |
| 主键 | tmpl\_id | | 索引 | tmpl\_code |
| 字段名 | 数据类型 | 空值 | 默认值 | 描述 |
| tmpl\_id | int | N |  | 模板ID |
| tmpl\_code | varchar(11) | N | # | 模板编号 |
| level | tinyint(1) | N | 1 | 短信级别.1:低;2:中;3:高 |
| content | varchar(100) | N | # | 模板内容 |
| state | tinyint(1) | N | 1 | 是否可用.1:可用;0:不可用 |
| create\_time | timestamp | N | NULL | 创建时间 |

| 逻辑表名 | 短信通道表 | | 物理表名 | sms\_channel |
| --- | --- | --- | --- | --- |
| 主键 | channel\_id | | 索引 |  |
| 字段名 | 数据类型 | 空值 | 默认值 | 描述 |
| channel\_id | int | N |  | 通道ID |
| name | varchar(10) | N | NULL | 名称 |
| state | tinyint(1) | N | 1 | 状态.1:启用;2:禁用 |
| url | varchar(64) | N | NULL | 短信通道URL |
| create\_time | timestamp | N | NULL | 创建时间 |

| 逻辑表名 | 黑名单表 | | 物理表名 | black\_list |
| --- | --- | --- | --- | --- |
| 主键 | black\_id | | 索引 |  |
| 字段名 | 数据类型 | 空值 | 默认值 | 描述 |
| black\_id | int | N |  | 黑名单ID |
| mobile | varchar(4) | N | NULL | 手机号 |
| state | tinyint(1) | N | 1 | 是否黑名单.1:是;0:废弃 |
| create\_time | timestamp | N | NULL | 创建时间 |

| 逻辑表名 | 流控规则表 | | 物理表名 | limiting\_rule |
| --- | --- | --- | --- | --- |
| 主键 | rule\_id | | 索引 |  |
| 字段名 | 数据类型 | 空值 | 默认值 | 描述 |
| rule\_id | int(11) | N |  | 统计ID |
| channel\_id | int(11) | N | 0 | 通道ID |
| threshold | int(11) | N | 0 | 阈值,每天最大值 |
| limit | int(11) | N | 0 | 限速值,每1分钟限速最大值 |
| create\_time | timestamp | N | NULL | 创建时间 |

| 逻辑表名 | 短信记录表 | | 物理表名 | sms\_record |
| --- | --- | --- | --- | --- |
| 主键 | record\_id | | 索引 | inx\_tmpl\_id  inx\_channel\_id |
| 字段名 | 数据类型 | 空值 | 默认值 | 描述 |
| record\_id | bigint(20) | N |  | 记录ID |
| tmpl\_id | int | N | 0 | 模板ID |
| channel\_id | int(11) | N | 0 | 通道ID |
| content | varchar(200) | N | # | 短信内容 |
| state | tinyint(1) | N | 1 | 状态值.  1:未发送;2:已发送;3:失败 |
| create\_time | timestamp | N | NULL | 创建时间 |

| 逻辑表名 | 通道统计信息 | | 物理表名 | channel\_stat |
| --- | --- | --- | --- | --- |
| 主键 | stat\_id | | 索引 |  |
| 字段名 | 数据类型 | 空值 | 默认值 | 描述 |
| stat\_id | int(11) | N |  | 统计ID |
| channel\_id | int(11) | N | 0 | 通道ID |
| total\_num | int(11) | N | 0 | 发送总量 |
| ok\_num | int(11) | N | 0 | 发送成功量 |

# 安全性

软件系统安全从三个方面考虑：

1. 系统网络安全性

短信平台部署在内网环境，防火墙内部。对外提供的接口及消息做来源判断，只有分配了访问权限的业务端才可进行接口的调用。

服务器只开辟特定端口，并对端口流量做监控。

1. 数据安全性

数据存储在MySQL中，从MySQL服务的安全性、数据访问权限做控制。

1. 用户权限控制

短信管理平台属于技术内部平台，不需要完善的权限控制。只需要分配给几个人即可。

# 风险点

# 人员安排

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 生命周期 | 人员 | 开始时间 | 结束时间 | 备注 |
| 系统设计讨论 | 易红莹、马达  金梁、何攀 | 9/26 | 9/27 |  |
| 任务拆解/排期 | 易红莹、马达 | 9/28 | 9/30 |  |
| 软件开发阶段 | 易红莹、马达 |  |  |  |
| 联调测试阶段 | 易红莹、马达 |  |  |  |
| 系统发布阶段 | 易红莹、马达 |  |  |  |
| 后续通道接入 |  |  |  | 至少接入三家 |

# 规范化

严格按照技术中心规范执行，具体项如下：

基于SVN版本管理

集成Jenkins自动构建发布

项目结构、包命名、类命名、配置管理、日志规范

代码Review