概要设计说明书

版本 1.0

项目名称：小企业2.0配置框架系统分析与设计

**修订历史**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本号 | 作者 | 内容提要 | 核准人 | 发布日期 |
| 0.1 | 鲁友炳 |  |  | 2017/01/17 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**目 录**

1 项目功能 3

1.2 项目背景 3

1.3 业务功能点列表 3

1.4 业务流程及关键业务规则 3

1.5 技术方案 4

2 重要功能设计 4

2.1 warden agent动态配置采集 4

2.1.1 功能描述 4

2.1.2 系统交互边界 4

2.1.3 接口定义 5

2.1.4 接口内部流程 5

2.1.5 外部接口依赖 5

2.1.6 异常处理 5

3 数据库设计【如有，必写】 6

3.1 表结构设计 7

3.2 优化设计 7

4 性能方面设计【可选】 7

4.1 性能瓶颈 7

4.2 性能评估方法和指标 7

4.3 应对措施1 8

4.4 应对措施2 8

5 风险评估【可选】 8

5.2 系统评估 8

5.3 缓存评估 14

5.4 资损风险评估 15

5.5 安全风险评估 19

6 运维支撑设计 22

6.1 数据核对 22

6.2 监控设计 22

6.3 运维CheckList 23

7 尚未解决的问题【可选】 24

# 项目功能

## 项目背景

小企业2.0配置框架能够兼容多种应用场景,可以只使用本地配置，或者只适用远程环境。

## 业务功能点列表

*[注：列举本系统完整的、需实现的功能点列表，和功能设计章节对应。可有一人梳理出所有功能点，然后分人编写，编写完成后进行合并]*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | **名称** | **简述** | **PD** | **优先级** |
| 1 | 小企业2.0配置框架 | 支持本地获取配置、远程获取配置 |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |

## 业务流程及关键业务规则

*[注：列举和本系统相关的重要、关键业务规则说明，供下面设计予以参考]*

## 技术方案

小企业2.0配置框架支持从不同数据来源(本地和远程)获取**应用配置数据**，能够从JVM系统参数、环境变量、配置文件读取**系统配置数据**。



小企业2.0配置框架主要功能模块包括:

* 可重新加载配置文件数据,注入Spring bean模块
* 重写disconf-client模块，完整迁移至servyconf-client
* 开发小企业适配器client，能够切换从本地与远程获取配置数据

# 重要功能设计

## 可重新加载properties配置模块

### 功能描述

*[注：功能描述，UC图等]*

Spring配置模块包含两个非常重要的Class:

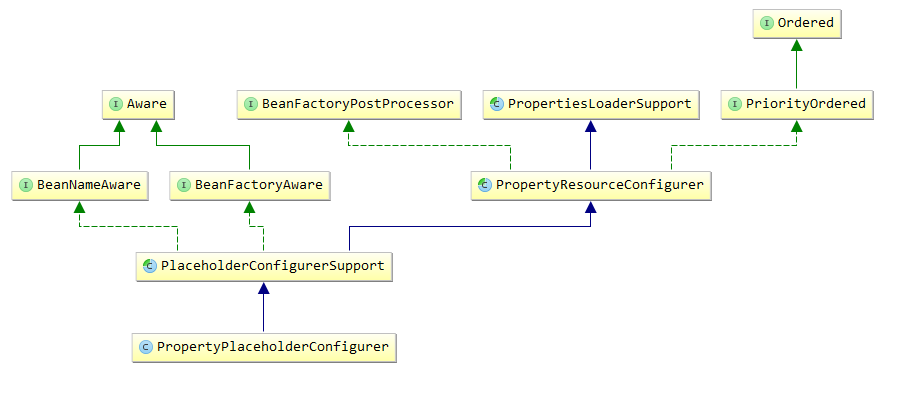
PropertiesFactoryBean和PropertyPlaceholderConfigurer

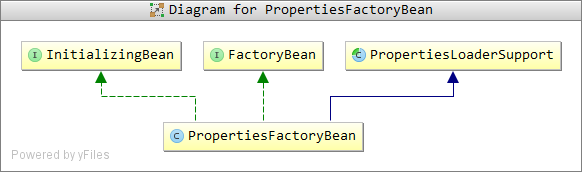
* PropertiesFactoryBean支持从多个properties文件读取配置信息
* PropertyPlaceholderConfigurer是Spring的资源属性配置器，主要用于解析Spring Bean中的占位符(placeholder),并将从properties文件获取的值注入对应的Spring Bean中。

使用Spring标准PropertyPlaceholderConfigurer,它读取properties属性文件,spring bean中属性的值${key}将被属性文件中的key值所替代。

可重新加载配置文件模块需实现以下功能: 当属性文件内容发生改变,属性里面内容要被重新读取并注入到bean对应的属性。

PropertyPlaceholderConfigurer继承了后置处理器**BeanFactoryPostProcessor。**





* 标准的Spring通过PropertyPlaceholderConfigurer提供对属性文件的支持:

<bean id="properties"  
 class="org.springframework.beans.factory.config.PropertiesFactoryBean">  
 <property name="location" value="classpath:config.properties"/>  
</bean>  
<bean id="propertyConfigurer"  
 class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">  
 <property name="properties" ref="properties"/>  
</bean>  
<bean id="mybean" class="cn.com.servyou.MyBean">  
 <property name="username" value="${xqy.db.username}"/>  
</bean>

* 可重新加载的PropertyPlaceholderConfigurer提供对属性文件的支持:

<bean id="properties"  
 class="cn.com.servyou.config.ReloadablePropertiesFactoryBean">  
 <property name="location" value="classpath:config.properties"/>  
</bean>  
<bean id="propertyConfigurer"  
 class=" cn.com.servyou.config.ReloadablePropertyPlaceholderConfigurer">  
 <property name="properties" ref="properties"/>  
</bean>  
<bean id="mybean" class="cn.com.servyou.MyBean">  
 <property name="username" value="${xqy.boss.username}"/>  
</bean>

配置文件发生修改时，通知ReloadablePropertyPlaceholderConfigurer重新将值注入Bean



注解分布式配置项(@Servyconf)和注解分布式配置文件(@ServyconfFile @ServyconfFileItem)

优点: **支持配置项**

**不需要编写Java Bean**

**缺点: 代码侵入**

**读取注解类配置都是通过Spring AOP拦截方式从配置仓库获取配置**

非注解基于XML分布式配置文件(**当前我们的使用方式**):

优点：无代码侵入

适合于旧项目的迁移

对于.properties配置文件，配置更新时，自动注入reload

缺点：缺点需要在xml中定义java bean

zk结点数据发生变更时(EventType.NodeDataChanged)，servyconf-client监听节点自动调用系统回调函数，做以下事情

disconfCenterFile.isTaggedWithNonAnnotationFile()

ReloadablePropertiesFactoryBean.this.reload

如果是XML配置，远程下载配置文件，注入至配置仓库，触发调用*ReloadConfigurationMonitor.reload*

比较两个新旧文件的修改时间，如果新文件时间大于旧文件，则调用mergeProperties()

通知ReloadablePropertyPlaceholderConfigurer，获取那些property受影响和获取受影响的beans，将受影响的bean属性值进行更新

### 系统交互边界

*[系统边界、交互时序图]*

Spring配置模块包含两个非常重要的Class:

PropertiesFactoryBean和PropertyPlaceholderConfigurer

* PropertiesFactoryBean支持从多个properties文件读取配置信息
* PropertyPlaceholderConfigurer是Spring的资源属性配置器，主要用于解析SpringBean中的占位符(placeholder),并将从properties文件获取的值注入对应的Spring Bean中。

1、原读取配置流程



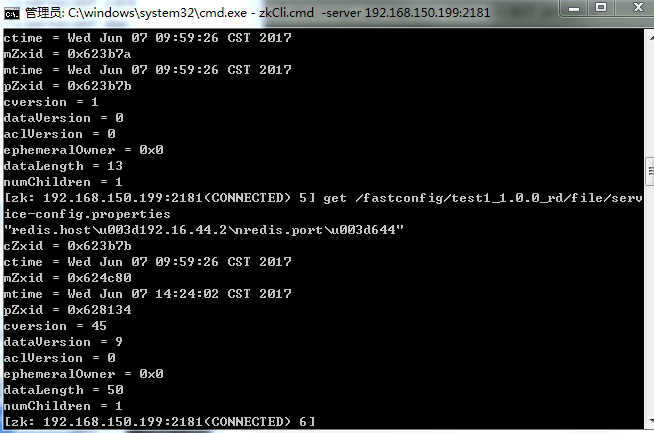
StringValueResolver对象是专门用来解析字符串的，spring解析bean的时候牵扯到字符串处理的有很多，这个的处理可以认为第一步最简单的处理，字符串到字符串的处理，spring bean的定义中所有的配置都是字符串都可以使用StringValueResolver这个对象处理。

在[spring](http://lib.csdn.net/base/javaee)的众多功能中，有一个帮助类PropertyPlaceholderHelper，专门负责解析路径中或者名字中含有占位符的字串，并负责填充上具体的值。

2、新读取配置流程

### 接口定义

*[注：描述对外提供服务的接口的输入输出、参数定义等]*



结点 data

/servyconf\_zk\_root/xqy\_iris\_demo\_1.0.0\_dev 127.0.0.1

*/file ip*

*/abc.properties 空*

*/[10.203.130.28]020121718-NB\_0\_c1987450-67a1-4b01-97a1-19e7265b7bc1 data*

### 接口内部流程

*【可以结合流程图详细描述内部的处理逻辑】*

### 外部接口依赖

*【描述此接口或功能对外部系统及对应接口依赖】*

### 异常处理

*[注：数据库异常、依赖系统问题、消息延迟/积压、二阶段延迟、重复请求等异常的处理逻辑]*

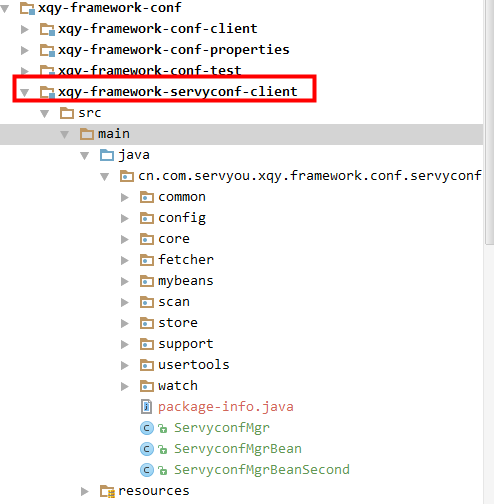
## servyconf-client功能模块

*[注：功能描述，UC图等]*

### 系统交互边界

*[系统边界、交互时序图]*

servyconf-client是对disconf-client模块进行重写，将对Properties可重新读取抽离出来



反射进制扫描注解

@ServyconfFile @ServyconfFileItem @ServyconfItem @ServyconfUpdateService

分析出配置文件与配置文件中的Field对应Method的Map关系



### 类定义

### 接口定义

*[注：描述对外提供服务的接口的输入输出、参数定义等]*

### 接口内部流程

*【可以结合流程图详细描述内部的处理逻辑】*

### 外部接口依赖

*【描述此接口或功能对外部系统及对应接口依赖】*

### 异常处理

*[注：数据库异常、依赖系统问题、消息延迟/积压、二阶段延迟、重复请求等异常的处理逻辑]*

## 小企业配置适配器功能模块

*[注：功能描述，UC图等]*

### 系统交互边界

*[系统边界、交互时序图]*

xqy-framework-conf-client是对本地或者远程配置模式进行适配的模块，它提供三种方式读取应用配置、获取系统配置数据,依次顺序为:

* 首先从（System.getProperty）-D JVM系统参数获取
* 从（System.getEnv）环境变量env读取
* 从classpath的servyconf.properties读取

### 类定义

### 接口定义

*[注：描述对外提供服务的接口的输入输出、参数定义等]*

### 接口内部流程

*【可以结合流程图详细描述内部的处理逻辑】*

### 外部接口依赖

*【描述此接口或功能对外部系统及对应接口依赖】*

1.disconf应用列表查询接口

2.disconf配置保存接口

3.es 索引file mapping新增

### 异常处理

*[注：数据库异常、依赖系统问题、消息延迟/积压、二阶段延迟、重复请求等异常的处理逻辑]*

# 数据库设计【如有，必写】

*数据库对象命名规则如下：*

|  |
| --- |
| *表名必须是英文单词,且长度小于26位* |
| *Sequence命名为:SEQ\_表名* |
| *Index命名为:表名的简写+字段的简写+\_IND* |
| *Check约束为:表名的简写+字段的简写+\_c* |
| *Unique约束为:表名的简写+字段的简写+\_u* |
| *PK约束为:表名+\_PK,主键尽量使用有意义的唯一标志字段。* |
| *字段名必须为oracle非关键字* |
| *用char(1)保存布尔型值或状态位* |
| *用char(8)保存8位的日期* |
| *非定长的字符存储需要使用varchar2* |
| *使用number代替integer,double和float* |
| *使用clob代替long,blob代替long raw* |
| *不使用数据库默认值,使用IBATIS的默认值* |
| *使用时间字段统一命名为gmt\_+字段含义* |
| *数据库函数的命名为:F\_+功能* |
| *数据库过程的命名为: SP\_+功能* |
| *数据库包的命名为: P\_+功能* |
| *表中字段长度和关联表不一致的* |

## 表结构设计

*[注：可用ER图加描述，如DO类图跟ER图近似可不画ER图。*

*表中字段长度要留有余量，每个字段都要有详细说明，慎用clob等大字段。*

*数据字典中要定义状态值描述（0、1代表xxx）*

*简称要有完整名称的说明*

*字段命名要遵循DBA的命名规范，中英文要能够望文生义，不用生僻字，实在不行用拼音*

*一个字段里面尽量不要存入不同的数据，也少用符号隔开的多个对象，如果必须使用，要有说明*

*如果有存储过程，存储过程的触发条件要详细说明*

*DBA要求每个表里面都要定义GMT\_CREATE、GMT\_MODIFIED这两个字段，如果记录可能删除，建议用DELETED表示是否删除（0正常，-1删除）]*

*【现有数据表有没有改变它的含义，或扩展它的内容？*

*现有数据字段有没有改变它的含义，或扩展它的内容？】*

*表结构设计示例，如有增加、修改、删除，请标红！*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *序号* | *字段名称* | *字段类型* | *字段说明* | *字段用途* |
| *1* | *CARD\_NO* | *varchar2(28)* | *贷记卡* | *非空* |
| *2* | *ID* |  |  |  |
|  |  |  | *交易号* | *非空* |

## 优化设计

*[注：数据冗余，优化查询方案等*

*表之间的关系要简单，涉及到多表的复杂查询和跨数据库的操作要合理使用冗余字段。*

*数据删除一般要采用逻辑删除，逻辑删除的数据要有清理机制。*

*合理利用事务，误操作后数据如何恢复。*

*根据数据流量，执行频率，制定分页方案。*

*要考虑数据库的设计方案对存储成本有多少影响。*

*过期数据如何清理。]*

# 性能方面设计【可选】

## 性能瓶颈

*【列出系统的性能瓶颈可能在那里】*

## 性能评估方法和指标

*【列出重要系统的性能问题的评估指标和方法，如果需要，可以咨询性能测试人员】*

## 应对措施1

*【列出具体的设计措施，*

*从那些方面改善系统性能*

*在什么情况下改善系统性能*

*】*

## 应对措施2

*【重复上述过程】*

# 风险评估【可选】

## 系统评估

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **类型** | **验证项** | **是否系分阶段考虑** | **说明** | **开发负责人** | **测试负责人** |
| **系统变更点** | 从系统层面描述这次变更的改动点 如： 提供服务的变更（如：服务删除、参数变更、错误码变更等） 与外部系统交互方式的变更（如：调用量、调用方式、频率变化等） 内部实现的重点变化（如：定时任务、调度任务、大数据量查询、主要业务逻辑、消息、缓存的） | **是** |  |  |  |
| 此次变更的重要风险点：包括**业务正确性、系统稳定性、数据兼容性方面**的风险 主要针对系统变更点，从被依赖系统、自身系统两个维度触发，评估可能的风险点。重点业务逻辑，即使没有风险，也要描述评估过程。 例1：ecctrlcore作为新系统，在系统稳定性等方面未经验证（每天有xxxx-xxxx调用量）。所提供的授信维护的服务如果出现稳定性和业务逻辑问题，将会影响申贷流程。 例2：ecbill修改了CAE代扣接口的调用，增加了支付工具参数，如支付工具控制错误，将会对不能扣余额宝的扣余额宝，产生投诉。或者对能扣余额宝的没扣余额宝，影响资金回收效率。 | **是** |  |  |  |
| 有哪些措施用于应对可能存在的风险点：比如开关、流量控制等 例：先打开票据还款的余额宝代扣（流量较小），验证扣款和支付工具选择逻辑没问题后，再开放业务 | **是** |  |  |  |
| **服务发布** | 是否新增或者修改过服务？ （有则列举具体服务，没有则填无） | **是** |  |  |  |
| 新增错误码（新接口或已符合规范的老接口）需要符合三代架构错误码规范 http://doc.alipay.net/pages/viewpage.action?pageId=69932847 http://doc.alipay.net/pages/viewpage.action?pageId=31567807 | **否** |  |  |  |
| 服务注释是否清晰，服务变更项是否同步注释？ 包括： 1. 具体功能描述 2. 使用场景 3. 同步实现或异步实现，若为异步，注明相关结果查询接口 4. 参数定义（1. 注明必填和选填；2. 格式化字符串参数注明格式要求；3. 若有集合参数，注明集合允许大小） 5. 返回值定义（若返回值为集合，注明排序策略和大小） 6. 错误码列举（必须包括：成功，失败可重试/失败不可重试，重复调用，未知异常四类） 7. 服务容量（包括TPS和每天最大调用量） 8. 是否有限流/降级措施及对应的触发场景和返回结果 9. 查询接口需要对返回结果中的敏感信息进行标注：账号、身份证、联系电话、地址等 具体格式参考：http://doc.alipay.net/pages/viewpage.action?pageId=69932849 | **否** |  |  |  |
| 服务参数、返回结果、错误码的变更是否兼容老接口 1. 不允许变更已有参数字段、返回结果字段、错误码的业务含义 2. 不允许删除已有参数字段、返回结果字段 | **是** |  |  |  |
| 服务变更是否通知到依赖系统 包括： 服务删除、参数变更、返回结果变更、错误码变更、数据处理变更（如数据顺序、数据集、数据时效性等）、配置变更 | **是** |  |  |  |
| 服务变更是否和上一版本兼容 | **是** |  |  |  |
| 服务返回结果是否有使用枚举类型 不允许服务接口返回结果使用枚举 | **是** |  |  |  |
| 是否有新增/变更稳定性控制措施，并通知给依赖系统 如果服务降级、限流等 | **是** |  |  |  |
| 是否有同名服务方法 不允许有同名服务方法 | **是** |  |  |  |
| 本地变量是否经过线程安全检查 1. 线程上下文，严格使用ThreadLocal，并在本地finally块中释放。ThreadLocal的使用参考RAII和RRID准则。 2. 不允许使用非线程安全的本地变量 | **否** |  |  |  |
| 服务粒度评估 查询服务： 1. 不允许批量返回大于5个不同用户的信息 2. 严格禁止万能查询接口，每个查询接口必须明确使用在一个业务场景 操作服务：不允许批量操作大于5个不同用户的信息 | **是** |  |  |  |
| **依赖变更** | 是否有新增强依赖 依赖等级定义：http://doc.alipay.net/pages/viewpage.action?pageId=24772667 | **是** |  |  |  |
| 新增的依赖服务是否也是此次新增的 如果是新增的，需要定义发布顺序 若同时有依赖同个系统新增的消息，要说明方案解决有顺序冲突的问题。（消息依赖先发订阅端，服务依赖先发服务提供者） | **是** |  |  |  |
| 是否存在系统启动依赖 只允许依赖数据库、应用服务器本地资源（如本地文件）、公共存储，不允许有其它基础技术服务、内部服务或外部服务依赖。如有特殊情况需说明。 | **是** |  |  |  |
| 是否有服务使用的变更 包括：调用次数、调用频率、入参集合大小、调用配置等 若有上述变更或新增服务依赖，需通知服务提供方 | **是** |  |  |  |
| 是否检查jar包冲突 | **否** |  |  |  |
| 是否有第三方jar包的变更（包括新增依赖和版本变化） 如果有，需要说明变更原因和第三方jar包的使用场景。评估第三方jar包安全性 | **否** |  |  |  |
| **数据服务** | **是否存在数据库结构/配置变更** 包括 1. 数据库表结构 2. 分库分表规则 3. 连接池 4. 超时时间 | **是** |  |  |  |
| **是否存在数据使用变更** 包括 1. 数据容量 2. 数据增长量 3. 数据访问量 4. 新增字段时，数据是否兼容 5. 新增业务表是否有全站唯一标识符（比如账户ID），为以后分库分表作考虑 6. 数据字段精度是否经过确认 | **是** |  |  |  |
| **批量操作：** 1. 批量查询，是否评估返回结果集合 2. 批量更新接口，是否评估批量条件，并验证更新数符合预期 3. IN条件操作，评估一次IN条件中的条数不大于200 | **是** |  |  |  |
| **是否涉及数据迁移/清洗** 若需要，说明是否迁移策略。如果需要停机维护，说明停机时间点、停机时长、停机时业务量。 | **是** |  |  |  |
| **是否有对其他数据系统的使用变更** 包括：其他数据存储中间件、本地文件系统、分布式文件系统、静态资源等 如果有，需要说明使用场景和使用规则 | **是** |  |  |  |
| **消息使用** | **消息配置变更** 1. 如果有新增消息，需要在发布计划中注明发布顺序：订阅端先发，发送端后发 2. 如果有消息下线，需要通知到相关系统，确定接收逻辑下线 3. 发送方如果有消息topic、eventCode等主体信息的变更，是否通知相关系统 4. 接收方如果有消息订阅关系的变更，如取消订阅、变更groupId等，是否有通知到消息中间件的owner | **是** |  |  |  |
| **消息发送量是否有变化** 如果有变化，需要通知：消息中心、消息订阅方 | **是** |  |  |  |
| **消息对象是否有变更** 1. 变更是否通知到消息依赖系统 2. 不允许删除/变更已有数据对象字段 3. 消息对象中不允许使用枚举类型 | **是** |  |  |  |
| **消息发送/失败场景评估** 1. 发送方对于可靠性要求高的消息，是否有考虑消息发送和业务逻辑的一致性。业务成功、消息发送失败时，是否有相应的识别及补偿机制。 2. 接收方对于可靠性要求高的消息，是否有考虑消息处理后的重试机制。 3. 接收方对于处理失败的消息，是否有区分失败场景。对于**不必**重试的错误场景，若仍然返回消息处理失败，可能会导致消息处理的死循环而积压 4. 异常处理逻辑是否已经测试覆盖 | **是** |  |  |  |
| **消息依赖等级是否有变化** 0级：消息确保快速可靠到达； 1级：消息确保可靠到达，有一定延迟； 2级：消息允许延迟和少量丢失，对业务无影响） 发送端和订阅端，任意一方有依赖等级的变化，需要通知到另一方 | **是** |  |  |  |
| **后台任务** | **并发策略评估** 对于任务并发处理的场景，需要有明确的锁机制，同一时间，同一任务只有一个线程在执行。 | **是** |  |  |  |
| **异常策略评估** 1. 任务执行结果包括：成功、失败不可重试、失败可重试。2. 对于必须成功的任务，要明确任务的重试机制和多次失败后的告警机制。 3. 对于失败任务进行重试，不能影响正常业务。保证重试逻辑正确。 4. 异常处理逻辑是否已经测试覆盖 | **是** |  |  |  |
| **执行顺序** 任务命令如果有执行顺序的要求，需要明确机制保证任务的执行顺序。 | **是** |  |  |  |
| **规则配置** | 规则配置：模型、折扣规则等。是否有线上验证机制。如新规则并行空跑、小数据量验证等 | **是** |  |  |  |

## 缓存评估

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **类型** | **确认项** | **说 明** | **开发确认人** | **测试确认人** |
| **本地缓存/tair** | 是否经过系统onwer和组长的review |  |  |  |
| 是否在正常业务逻辑中刷新缓存 |  |
| 缓存刷新失败是否影响业务逻辑 |  |
| 缓存获取失败有何补偿措施 |  |
| 是否存在多个缓存刷新入口 |  |
| 刷新频率 |  |
| 缓存命中率 评估缓存命中率，分析缓存穿透对DB的影响 |  |
| 缓存总大小以及预计增长趋势 |  |
| 缓存与持久化数据不一致有何影响 |  |
| 在缓存出现异常不能提供服务的时候，系统db参数设置（连接池连接数量等）和底层db容量可否支持正常业务 |  |
| **tair** | 读操作（日访问量，是否与tair系统owner确认） |  |
| 写操作（日访问量，是否与tair系统owner确认） |  |
| 线上group、namespace是否已经申请 |  |
| 单条记录大小 |  |
| **本地缓存** | 是否使用本地定时任务 |  |
| 缓存数据量评估，是否影响JVM内存空间的使用 |  |
| 集群缓存数据不一致有何影响 |  |

## 资损风险评估

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **确认项** | **说 明（此列必填）** | **开发确认结果** |
| **公共检查项** | **幂等性检查：** 本次是否有新开的服务接口，这些服务接口是否具备幂等性控制要求。 如果需要幂等性控制，则： 1，给出以什么单号做幂等性控制，确定调用方是否可以以此单号做幂等性控制。 2，给出接收到重复调用后，返回的错误码，确定调用方没有把此错误码当做失败处理。 3，给出幂等控制的底层机制，并确定底层机制确实存在。且抛出的异常被正确处理。 4、页面幂等性保证，重复刷新页面、回退页面不会导致金额等出错。 5、系统内部确保不会产生重复单号。 6、幂等控制是否考虑分库分表、历史数据迁移、failover因素。 **7、调用服务接口出现失败而重试时，使用的流水号是否一致。** | *案列：XXX服务，以XXX作为幂等控制单号。收到重复调用，会返回XXX错误码。以数据库XXX表XXX字段的唯一性约束做幂等性控制。违反约束后，会抛出XXX异常，会最终转换成XXX错误码。* | 开发确认人： 风险点及对应Action: |
| **并发更新检查：** 业务状态更新操作的单笔或批量SQL，是否遵循了： 1，单笔操作时：一锁，二判，三更新的规范 2，批量更新流水时，是否在SQL的where条件中增加了原有流水的状态，且检查了update方法返回的记录条数是否满足预期。  缓存（本地、分布式、线程变量）信息的更新，是否遵循了： 1、确保不出现脏数据，引起资金操作异常； 2、确保关联缓存的更新，设计了合理更新顺序、遵守数据一致性原则； |  |
| **支付数据防篡改检查：** 本次是否有支付相关的数据，经过外部网络传输（URL参数，表单等，和机构，商户交互等）。 如果有请给出防篡改机制。 | *XXX支付，XXX数据经过URL传递参数，经过了签名。* |
| **资金处理服务依赖检查：** 是否新增了对资金处理服务的依赖，如果有则检查： 1，对返回的结果，有至少四种情况的判断（成功，失败，重复调用，未知异常）。如果没有四种情况，请特殊说明。 2，对于资金处理的返回结果，是否检查了金额和状态等关键要素。 3，对于依赖的资金处理服务，系分中是否体现了该服务的资金流。 | *典型场景： ①时间维度临界值：例如产品帐分为日账、月账、年账；关注时间维度边界值，跨日（0点）、跨月（大小月） 需要明确是否业务上下链路的时间周期是同一时区。 ②额度计算客户模型：确定以卡或客户为维度计算额度 ③ 额度恢复：验证业务状态与额度状态保持一致：业务失败后需保证同步恢复额度；验证额度恢复计算公式：额度计算的逆向公式* |
| **兼容性分析：** 确保： 1，新系统处理新数据 2，新系统处理老数据 3，老系统处理老数据 4，老系统处理新数据 这四种情况，都经过分析和测试。 |  |
| **线上线下存在差别的关键配置检查：** 例如：确保账户类的配置，上线时都被替换为线上的配置。 |  |
| **额度：** ①额度计算时是否维度够全面？（业务维度、时间维度、客户模型维度、额度占用/恢复） ②额度超限的场景是否考虑到。 ③额度占用和**恢复**场景是否考虑周全。业务状态是否与额度状态保持一致。 |  |
| **资金核对检查：** 所有资金业务，是否部署t+1核对，核对逻辑是否通过测试，是否要修改完善。 |  |
| **应急预案：** 本次研发涉及的产品，如果发生了资损，是否有明确的应急预案，应急预案是否可执行。 |  |
| **机构、商户对接** | **接口约定检查：** 1，金额单位约定是否一致 2，返回的结果状态，是代表业务成功还是操作成功，或者这两种状态都有 3，成功失败是以实时返回结果为准，还是以后续对账文件为准 |  |
| **接口幂等性能力检查：** 机构或商户的接口是否具备幂等性控制能力。如果没有，请给出调用方的风险规避方案。 |  |
| **带有滞纳金的业务** | **防止用户产生滞纳金：** 1，失败的结果是否能有效和及时通知到用户 2，是否存在系统不可用的风险，有没有提前公告，并做好恢复的补偿机制 |  |
| **有时效性的商品** | **确保实效性：** 1，是否有超时控制，确保过期的业务单据不会被推进成功 2，是否将结果及时告知了用户 |  |
| **营销活动** | **防止负面影响：** 1，抽奖类活动，是否经过了压测，确保奖项抽取结果，满足预期。 2，抽奖类活动，给用户的文案（页面，短信，邮件），是否经过文案专员，法务专员的审核。 3，折扣类活动，配置项是否正确且配置完毕。 4，活动力度和范围是否经过检查，并做严格控制。（折扣范围，准入规则，活动人数等） |  |
| **微贷检查项** | **客户状态和资金风险** 1. 是否会对客户授信产生授信金额放大的风险 2. 是否会对客户状态产生风险，导致影响资金操作流程：如强扣标记、出催、入催等 |  |
| **客户类型和资金风险** 对客户进行限制或者功能开放时，是否考虑支付宝、淘宝的客户体系及可能的变更对资金操作的影响（如：主子账户、账户认证等级、店铺转让、支付宝账号换绑等对还款、放款、支付的影响） |  |
| **业务边界检查** 1、如当天放款当天到期 2、当天放款已逾期等场景 3、跨天处理：资金处理是否考虑跨天场景，如还款跨天、放款跨天、分润跨天、赔付跨天等 |  |
| **二阶段提交检查** 涉及xts分布式事务的，是否覆盖二阶段提交/回滚的验证 二阶段提交/回滚失败的，是否有监控告警 |  |
| **利率/费率计算，配置检查** 1. 利率、费率的配置，是否上线 2. 修改点是否影响利率、费率的计算流程 3. 利率，费率的修改入口是否经过数据验证。且线上有验证，核对，监控机制 |  |

## 安全风险评估

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **类型** | **确认项** | **说 明** | **开发确认人** | **测试确认人** |
| **公共检查项** | **权限控制检查：** 本次是否有直接面向终端用户的功能点或者外部接口（如json、mobilegw接口等），这些页面或者服务接口是否涉及到账户私有信息的查询或者操作，如有则需要权限控制检查： 1，确定业务功能的受众对象，对于不需要权限控制的功能点业务上明确原因； 2，面向前台用户的功能点，确定代码实现层面进行操作主体校验，确保A账户只能查询或者操作其账户关联私有信息； 3，面向后台用户功能点，不可依赖页面渲染逻辑进行权限控制，避免绕过页面的请求调用；确认server端权限控制至少不弱于页面渲染的权限要求； 4、MNG类系统功能页面，除公共出错页面、提示页面等之外，必须要进行权限控制，且禁止简单的role\_login、role\_login\_normal等简单权限； 5、涉及到多流程步骤的功能页面，代码层面需要进行强一致性校验，不允许直接绕过前置步骤直接提交后置步骤的请求，防止流程绕过； | XXX功能，以XXX作为条件查询信息。获取查询结果或者查询前会进行信息主体校验，A账户不可查询（或操作）B账户私有信息。 |  |  |
| **敏感信息检查：** 业务信息输出需要遵循： 1，高敏感度要求的卡号、身份证号、密码、短信校验码、动态口令禁止在任何途径明文展示（包括页面、页面源代码、日志等）； 2，对外敏感信息输出遵循数据安全输出规范，对输出对象(即合作伙伴)有同样的约束要求； 3、外部接口对外信息输出，能够支持信息敏感度级别定制化输出策略，如合作伙伴满足什么样的资质或者签署怎样的协议才能输出敏感度更高的信息（卡号、身份证号、真实姓名等）； 4、对于高敏感度的信息输出，应用层需要考虑支持加密传输（如身份证号加密对外输出）； **5、内部文件下载如果涉及到账务、用户私密信息等敏感数据，一定要经过ADC数据分发平台 6、与外部机构交互的密钥是否暴露在antx等配置文件中。所有密钥需要加密处理 7、数据存储（缓存、DB、本地文件、分布式文件系统等）中的敏感信息是否经过加密处理，包括： 银行卡号 证件号（包括身份证号、军官证等，工商注册号不在此列） 密钥** |  |
| **请求伪造检查：** 确认直接面向终端用户的功能是否进行充分的请求伪造防御设计： 1、数据防篡改：确认表单关键只读字段（金额、账户等）是否应用数据防篡改机制； 2、CSRF防御：确认表单、json等面向终端功能是否应用CSRF防御机制； | XXX支付，XXX数据经过URL传递参数，经过了签名。 |
| **客户端存储检查：** 主要检查范围为浏览器cookie、客户端缓存技术、手机客户端： 1、cookie不得存放敏感数据； 2、浏览器缓存技术应用不得存放敏感数据； 3、手机客户端：禁止本地明文存敏感数据，如密码、认证token； |  |
| **加密及防抵赖检查：** 主要应对与合作伙伴的信息共享、通知及协议应用： 1、支付宝全站（包括无线客户端）HTTPS或者应用层加密，不能使用弱算法； 2、外部接口调用及调用结果响应均需要采用签名机制防伪造、防抵赖； 3、对外通知(向合作伙伴系统的通知)需要采用签名机制防伪造、防抵赖； 4、手机客户端intent消息需要有可靠技术机制甄别可信来源，防消息伪造； |  |
| **跳转及重定向检查**： 确认系统中是否存在页面重定向功能点，若存在则需要进行如下检查： 1、原则上禁止向集团外域名的重定向； 2、若业务上一定需要向合作伙伴网站跳转，则一定需要进行与合作伙伴匹配的白名单校验； 3、手机客户端原则上不可直接打开或内嵌外部网站；若业务上需要，则一定需要进行与合作伙伴匹配的白名单校验； 4、手机客户端跳转外部网站，对于非集团内域名白名单，功能上需要给用户以显著的提示，避免钓鱼风险； |  |
| **第三方技术应用检查：** 主要检查项目或者系统中是否有新引入的第三方技术： 1、新引入的第三方资源（框架、jar、js/css、库文件等）必须经过安全团队确认； 2、严禁直接在线引用第三方资源（如直接通过外部url应用js、css等）； |  |
| **机构、商户对接** | **防伪造：** 1，对接双方约定一致的通讯、签名及加密机制，且技术层面需要能够提供进行有效来源甄别； |  |
| **防抵赖：** 2，对接双方约定一致的响应、通知的签名及加密机制，且对于调用双方传递的信息需要考虑较长的保存时效（例如关键日志存放三个月）； |  |
| **时效性：** 3、授权认证类接口调用技术层面需要考虑进行时效性校验或者提供相关要素（例如快捷登陆需要携带时间戳并签名跳转商户网站，时间戳可作为时效性校验要素）； |  |
| **客户端类软件** | 1、手机客户端在产品使用协议未明确说明得到用户授权情况下，不可上传用户手机 私有要素（通讯录、通话记录、短信等）； 2、无线客户端认证加密相关的关键技术实现代码，安全黑盒化（如放在SO中）； 3、 客户端：有限定的加载动态库(so、dll)文件，应用层采取可靠校验机制避免加载被篡改或者恶意库文件； |  |
| **攻防设计** | 1、所有注册口、登录口、外部收银台、未登录表单提交等入口，应接入防CC攻击产品（如图形验证码、hummock、RDS等） |  |
| **注入风险** | 1、XSS注入风险检查，关注涉及页面渲染且渲染数据可能由用户引入的点； 2、sql注入检查：对基于ibatis的应用，需要注意sql语句慎用$$进行变量引用，除非被引用变量不可能由外部注入； |  |

# 运维支撑设计

## 数据核对

*【数据表变更同步到ODPS，数据核对规则】*

## 监控设计

*【列举一下监控设计：*

*监控点设计*

*监控日志打点，*

*监控指标设计*

*报警机制设计和采用*

*监控异常之后处理机制设计*

*】*

## 运维CheckList

*系统发布和运行过程中，运维部门关注很多内容，运维CheckList作为概要设计的输出，可以产出专门的文件，也可以在这里说明：*

|  |  |
| --- | --- |
| **类型** | **确认项** |
| **数据库** | 是否需要新增数据库服务器？或使用现有的哪台数据库服务器？ |
| 数据库需要多大存储空间？ |
| 什么数据的量较大，半年或一年内会有多大的量，数据能否归档到历史库？ |
| 是否需要初始化数据？或数据迁移？其涉及的数据量有多大？ |
| **通知服务器** | 是否使用通知服务器？（例如，交易需要通知淘宝、外部商户） |
| 使用通知服务器产生的消息通知量约多少？ |
| **其它服务器** | 是否需要共享存储，请列出共享的主机列表，存储的内容，安全要求，共享目录名，共享的空间大小，月增长率，IO大小，增长率 |
| 需要的服务器数量，配置 |
| **网络** | 预计将新增多大的网络流量或PV？ |
| 是否有大量的邮件发送需求？或者可能造成某一小段时间内邮件发送接受量激增？ |
| 是否需要专线或者VPN还是公网服务支持 |
| 预计每秒将新增多大流量、PV或者session数 |
| 负载均衡提供哪些服务端口，是否需要做成80跳转443 |
| 是否需要session保持 |
| TCP服务是否为长连接，连接保持多长时间 |
| 是否需要做SSL卸载 |
| 是否有特殊的安全访问控制 |
| **安全** | 是否通过安全QA测试和代码review测试,VM XSS；session token 防CSRF |
| 是否有外部跳转存在，跳转是否给出白名单，参数传递清单? |
| 是否使用了加密机制-加密算法是什么？ |
| **域名** | 是否需要新域名？外部域名？内部域名？是否需要ssl证书？ |
| 是否存在由于全局域名所带来的版本不一致问题？ |
| 是否存在预发布、正式发布环境域名需要分别配制的情况 |
| **配制文件** | 预发布、正式发布环境是否存在不一致的情况？ |
| **支撑环境** | 应用系统是否需要新单独部署（即新增web服务器）？ |
| 是否需要对JDK，JBOSS，APACHE，Linux环境进行升级（如在etc目录下需要加入特殊的配置）？ |
| 预发布环境与生产环境是否存在差异？ |
| **权限** | 是否需要修改权限控制配置文件？ |

# 尚未解决的问题【可选】

*[注：说明在本程序的设计中尚未解决而设计者认为在软件完成之前应解决的问题。]*