上海巢聚网络科技有限公司

JEE 项目
概要设计

版本 **<0.1>**

修订历史记录

日期	版本	说明	作者
2009-02-25	0.1	草案	蔡源

目录

设计指南

1. 引言

1.1 背景

随着号百业务迅猛发展，上海号百从单个网站发展到七个之多，且各网站内容独立，页面风格不一，没有任何横向关联。为了提升业务使用体验，有效聚合用户，需要将号百现有门户网站有效整合在一起，形成业务集成门户，统一管理、统一接入、统一风格，以满足号百业务发展需求，提升号百品牌及知名度。

1.2 目的

本文档的目的是说明系统的整体方案架构，设计标准、约定和代码模式。

1.3 范围

本文档的适用人员包括项目开发组全体成员：技术管理人员、系统设计人员、系统开发人员、系统测试人员、系统维护人员、推广培训人员及其他相关人员。

1.4 术语定义

缩写	英	中
AJAX	Asynchronous JavaScript and XML	异步 JavaScript 和 XML
AOP	Aspect Oriented Programming	面向方面编程
IoC	Inversion of Control	反向控制
MVC	Model-View-Controller	模型-视图-控制器
ORM	Object/Relational Mapping	对象关系映射
REST	REpresentational State Transfer	具象状态传输
SOAP	Simple Object Access Protocol	简单对象访问协议
WSDL	Web Services Description Language	Web 服务描述语言

1.5 参考资料

2. 概述

本文档主要从系统最初的设计思想着眼，针对用户的应用环境，参考目前主流的 Java 解决方案，利用现有资源和经验积累，制定了这套严格的 J2EE 架构标准的设计方案。

文中首先介绍了系统开发背景，目的，总的设计思想；然后在这个基础上针对实际的业务需求进行系统架构。接着针对架构进行技术选型，选型原则：主流的、稳定的、资源丰富的实现。技术方案确定后，开始系统功能模型的设计；最后对系统运行期的数据处理、传递、安全方面的机制做一个设计。

3. 总体方案架构

3.1 基本设计概述

主流企业级应用系统解决方案主要包含以下需求：

- 持久性（persistence）：实现数据存储、处理，数据与对象映射，数据缓存（caching）。

- 事务（transaction）：确保一组关联操作正常、完整的执行。
- 安全性（security）：保证系统的通信安全、数据安全。
- 负载均衡（load balance）：在大量并发访问时，保持系统可用。
- 监控（system monitoring/management）：监控系统运行状况，设置系统参数。
- 日志（logging）：记录系统运行情况和异常，记录特定用户操作。
- 应用集成 （application integration）：与其他系统、应用程序集成。
- 认证/权限/组织角色管理（authentication/authorization）：管理系统用户、组织职权结构，限制特定用户对特定功能、特定数据的访问。
- 业务模型（domain model）：管理系统中业务对象的属性、字段。
- 业务逻辑（business logic/rules）：实现业务规则和业务逻辑。
- 工作流（work flow）：实现多用户、多环节之间的业务处理流程。
- 文件管理（file management）：管理文档，实现系统内部的文件传递。
- 报表/打印 （reporting/printing）：实现数据打印，实现报表的定制和输出。
- 门户/信息发布 （portal solution）：发布企业相关的信息、新闻，提供企业客户的访问入口。
- 通信（communication/messaging）：系统内部的消息、通知；系统与外部角色（比如企业客户）之间通过不同通信媒介（电话、网站、邮件等）的互动。

3.2 设计思想

基于系统的整体架构和目前的一些实现，结合原有的项目积累和 Open Source 的成果。系统将采用 Java 作为开发语言，利用 Java 强大的跨平台功能，兼容主流服务器和操作系统。基于 jdk 1.5 和 j2ee 1.4 的基本规范，兼容市面上各种主流应用服务器。BS 模式解决了原有应用程序发布、安装繁琐，维护更新困难的问题。技术方面采用主流 Java 开源框架 Spring 作为核心框架。各子系统间采用 WebService 标准交互数据，利用 WebService 良好的结构性、安全性、事务性保证数据准确、正确、完整（特殊时采用文本或 XML 格式发送）。

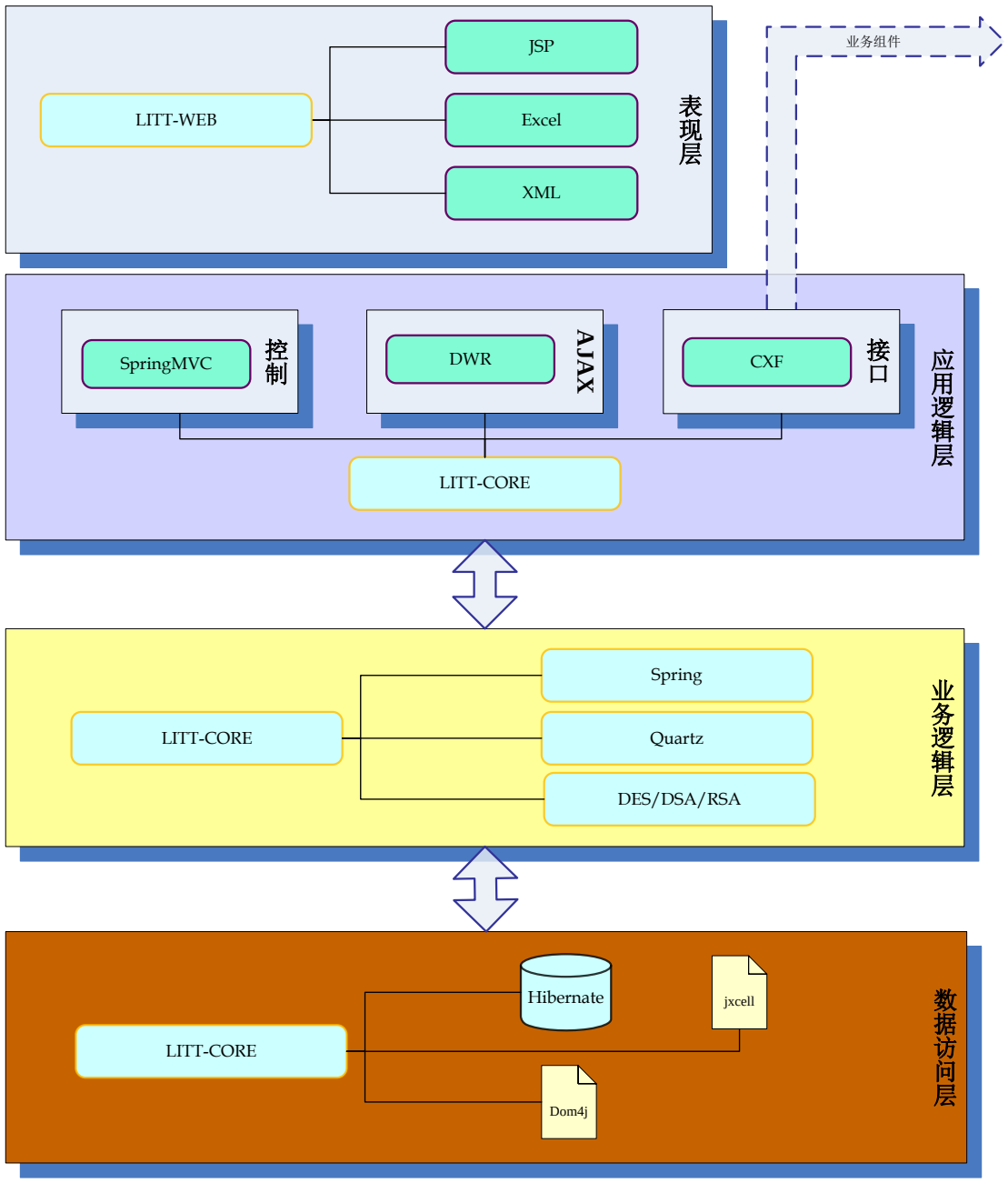
3.3 系统架构

3.3.1 系统层次

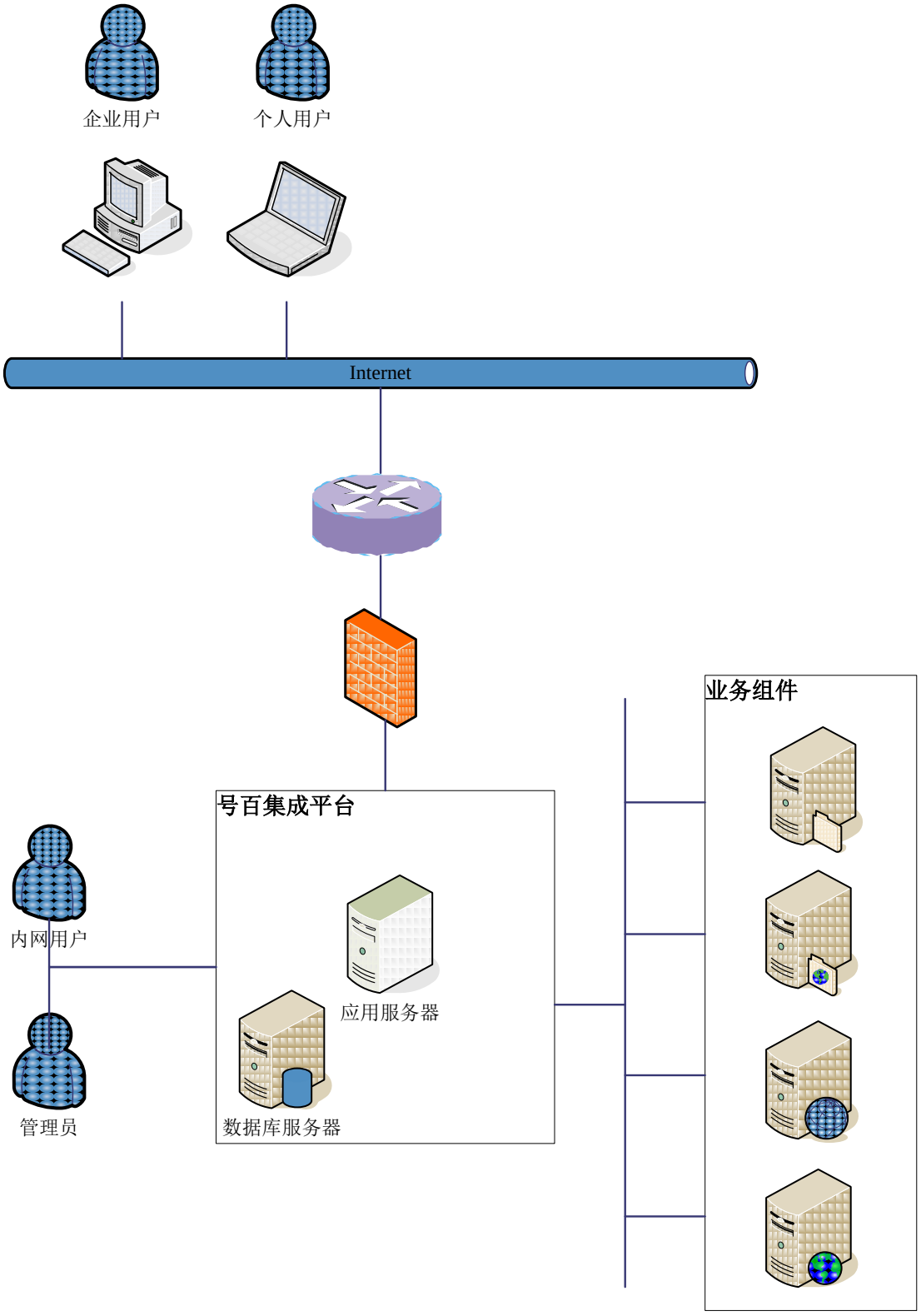
依照严格的 J2EE 架构标准，依据 J2EE 应用模式定义使用四层架构进行实现，它避免了传统两层结构的局限性，能够提供足够的可伸缩性、可配置型、可接入性和可管理性。
本系统共分为：**数据访问层**、**业务逻辑层**、**应用层(控制/AJAX/接口)**、**表现层**，共四个逻辑层次。术语表示为：DAO、Service、Action/AJAX/WebService、View。

- **数据访问层**：简单的数据库操作由 Hibernate 实现，复杂的业务逻辑如批处理等由 JDBC 调用 SQL 或存储过程实现，每个存储过程都是一个独立的完整的业务，事务管理由存储过程实现。
- **业务逻辑层**：该层提供基础的业务服务，处理相关业务逻辑，由 Spring 提供事务支持。Service 层调用底层抽取上的数据，并相互关联形成一个完成的业务操作；因此，开放给上层的方法都应该是一个完成的业务。
- **应用层**：根据上层需要调用相应业务逻辑。
  - **控制**：Action 采用 SpringMVC 的 MultiActionController 作为基类，同一模块共享一个控制类，每一个方法完成一个独立的完整的业务操作。一般情况下数据更新的操作由于受 Spring 事务管理，每个方法都只对应 Service 中的一个方法。
  - **AJAX**：通过有效的使用 AJAX，简化 Web 应用，并提高用户体验。
  - **接口**：通过开放接口，允许与外界系统进行交互。
- **表现层**：作为表现层，应该尽可能的简单，专注于数据展示，不依赖任何 Web 框架（如 Struts/WebWork/SpringMVC）。结合 WebWidgets,Ajax 简化 Web 应用。

3.3.2 技术架构图


号百集成门户 - 技术架构图
JSP
LITT-WEB
Excel
XML
表现层
业务组件
SpringMVC
控制
DWR
AJAX
CXF
接口
应用逻辑层
LITT-CORE
Spring
LITT-CORE
Quartz
DES/DSA/RSA
业务逻辑层
Hibernate
jxcell
LITT-CORE
Dom4j
数据访问层
页 1

3.3.3 网络拓扑图



3.4 数据库设计

根据对一般业务系统的分析，将数据库和程序系统统一进行整体描述，展示数据库的表之间以及与程序模块间的关系。

3.4.1 数据表分类

根据“处理特点”，将数据表进行分类如下：

- ✓ 业务数据表：记录业务发生的过程和结果。
- ✓ 基本编码表：描述业务实体的基本信息和编码。
- ✓ 辅助编码表：描述属性的列表值。
- ✓ 系统信息表：存放与系统操作、业务控制有关的参数。
- ✓ 累计数据表：存放业务的当前值和累计值。
- ✓ 结算数据表：存放各个时期末的结存数。
- ✓ 决策数据表：存放各个时期内发生的统计值。

3.5 开发环境

- ✓ JDK 版本：1.5.0 及以上
- ✓ J2EE 规范：1.4
- ✓ JSP 规范：2.0
- ✓ Servlet 规范：2.3
- ✓ Tomcat 版本：5.5 及以上

3.6 生产环境

3.6.1 软件环境

操作系统	Window/Linux/Unix...	
Java 虚拟机	JRE 1.5 及以上	
应用服务器	Tomcat 5.5 及以上 JBoss 4.0.0 及以上 Websphere 6.0 及以上 Weblogic 9.0 及以上	个别应用服务器可能会需要根据需要安装升级补丁
数据库	Oracle 9i2	
编程语言	Java 1.5、J2EE 1.4	
浏览器	IE 6 及以上	

3.6.2 硬件环境

CPU	P4 2.0 以上	
内存	1G 及以上(推荐 2G)	视具体的业务压力
硬盘	200M	

3.7 技术选型

- ■ Spring
  版本：2.0
  是一种全方位、多层次、轻量级的 J2EE 应用整合框架，它提供一种新的机制来管理业务对象其依赖关

系。

- Hibernate

  版本：3.2

  是 Java 对象持久层轻量级封装框架，是管理持续性数据问题的一种完整的解决方案。它既可以用来在 Java 应用程序中取代大部分 JDBC 代码，也可以整合到 J2EE 系统中作为持久层框架。

- Log4j

  版本：1.2.8

  Apache 的一个开放源代码项目，通过使用 Log4j，我们可以控制日志信息输送的目的地是控制台、文件、GUI 组件、甚至是套接口服务器、NT 的事件记录器、UNIX Syslog 守护进程等；我们也可以控制每一条日志的输出格式；通过定义每一条日志信息的级别，我们能够更加细致地控制日志的生成过程。

- XFire

  版本：1.2.6

  支持众多 WebService 标准的 Java WS 框架。XFire 对 Spring 提供了强大的支持，可以非常方便地在 Spring 中使用 XFire 实施 Web Service；此外 XFire 为客户端提供了多种访问 Web Service 的方式，支持基于 JVM 的本地测试。优点：支持 Spring、Pico、Plexus、Loom 等容器；灵活的 Binding 机制。XFire 比 Axis1.3 快 2-6 倍； XFire 的响应时间是 Axis1.3 的 1/2 到 1/5。

- DWR

  版本：2.0.1

  DWR(http://getahead.org/dwr/)是一种新的 Ajax 方式，有别于访问 URL 的传统模式，DWR 自己在服务端作了一个通用的 Service 层 Gateway(Proxy)，使得客户端的 Javascript 可以直接操作远程的服务器端的 Service 层，再配合 TrimPath JSTemplate 以 Template 而不是 Dom Builder 的方式将 Service 层返回的 POJO 渲染成复杂的页面，创造一种新的纯 Ajax 编程模式。

- Quartz

  版本：1.6.0

  Quartz 是一个开源的作业调度框架，它提供了巨大的灵活性而不牺牲简单性。你能够用它来为执行一个作业而创建简单的或复杂的调度。(由于 Quartz 使用独立线程并且经常为后台长时间调度，实际部署生产系统时建议应与业务系统分离，避免影响业务系统运行性能)

- MooTools

  版本：1.2

  MooTools 是一个简洁，模块化，面向对象的 JavaScript 框架。它能够帮助你更快，更简单地编写可扩展和兼容性强的 JavaScript 代码。 Mootools 从 Prototype.js 中汲取了许多有益的设计理念，语法也和其极其类似。但它提供的功能要比 Prototype.js 多，整体设计也比 Prototype.js 要相对完善，功能更强大，比如增加了动画特效、拖放操作等等。

3.8 开发模式

开发模式采用 XP(极限编程理论)，注重快速反馈，增量迭代，减轻开发风险。同时加强团队合作，提高生产效率。

4. 系统功能模型

4.1 模型说明

4.1.1 参数字典

参数字典分“**参数类型表**”和“**参数字典表**”。

参数类型表定义不同类型的业务参数，如用户类型、公告类型等。每个类型又可分为：1 – 不可修改，2 – 可增加，3 – 可修改，4 – 可删除 4 种。如公告类型，可被定义为“可增加”类型的，即公告类型可以增加，但不能被修改或删除。**参数类型表**另外用状态字段定义：1 – 正常，2 – 屏蔽，9 – 系统。对于某些业务需要，不需要用到的参数，可设置成屏蔽，即在业务系统中将无法使用该参数；对于状态为

系统的，则不能进行此项操作。

参数字典表为明细表，对参数类型定义业务参数，如公告类型可分为：1 – 公告，2 – 新闻，3 – 通知，4 – 紧急通知等，由于公告类型为可增加类型，故可在此基础上进行增加，但一旦使用过的公告，则不能进行修改或删除了。对某个业务参数细项，又有 3 种状态：1 – 正常，2 – 屏蔽，9 – 系统。由于业务需要可暂时屏蔽某些业务参数；但对于状态为系统的则不能进行此项操作

4.1.2 日志记录

日志按照类型分：操作日志、业务日志、系统日志。不同类型的日志有其相应的处理逻辑及具体实现，以下分别说明。

操作日志：记录操作员登录后执行的相关操作。（目前只对更新数据库的操作记日志，查询不记录）

业务日志：记录业务处理信息，如转账时金额的变动数额等。

操作日志同具体的业务应该是相互分离的，不在同一个事务中，及无论业务操作是否成功，都将记录用户操作。而业务日志记录业务的详细信息，应作为业务的一部分，与业务存在同一个事务中。因此操作日志一般在控制层编写，而业务日志一般在业务层编写。

注：可通过注解+拦截器模式提供非入侵的操作日志记录；而业务日志一般只能编码实现。

操作日志模型

属性	名称	类型	备注
busiType	业务类型	int	0 - 未指定 1 - 保存 2 - 更新 3 - 删除 4 - 查看 5 - 查询 6 - 审核 ...
moduleCode	模块编号	String	对所有功能模块，都有唯一对应的编号，如“系统配置”对应“SYS-001”
opId	操作员 ID	Long	
opName	操作员名称	String	冗余数据，这样就不需要关联操作员表了
opIp	执行操作的 IP	String	
opDatetime	操作时间	Date	数据库默认
content	操作内容	String	具体操作内容描述
status	操作状态	int	0 - 失败 1 - 成功

系统日志：记录系统日常运行时的行为日志，目前采用通用的日志框架，以手工编码的形式记录

4.1.3 License

程序通过 License 分发方式，对分发的系统进行使用授权。

1) 采用 DSA 算法先生成一个公钥和一个私钥。私钥存放于本地，公钥存放于程序当中。可对每个产品、每个客户生成独立的密钥对，再进行证书授权、鉴权。
2) 用私钥生成 license 签名，签名内容包括 license 关键属性（如客户名称，过期时间），随同 license 文件分发
3) 程序启动时读入证书文件和密钥文件，用户登陆时进行 license 校验，若为通过校验则无法继续使

用。未通过校验时自动跳转至证书更新页面，可输入新的证书信息来更新证书以便继续使用。

4) license 文件和密钥文件同程序一同分发，位于类路径，系统启动时复制到 HOME_PATH 下（不覆盖）。

5) 证书更新采用 DES 算法（密钥即 licenseId）对内容进行加密，加密后的内容可通过 IM、邮件等形式分发给客户，由于加密后的内容需要通过相应的密钥进行解密方能获得原始内容，故不用担心传出过程中被盗取。客户收到新证书后，可直接在界面上进行更新操作，系统将自动解密并更新证书文件。

4.1.3.1 License 数据结构

licenseId	证书 ID	FDDF2C35-FE5E-F068-A1E6-2A68056147EC	根据服务器唯一信息加 MD5 算法生成的唯一 ID
licenseType	证书类型	DEMO - 演示版 TRIAL - 试用版 STANDARD - 标准版 ENTERPRISE - 企业版 DEVELOPMENT - 开发版	不同类型之前对应不同的校验策略
productName	产品名称	teamwork	授权产品名称
companyName	公司名称	littcore	本公司名称
customerName	客户名称	chinamobile	授权客户名称
version	版本	1.0	分发版本
createDate	创建时间	61178947200000(2008-8-8)	
expiredDate	过期时间	61223097600000(2010-1-1)	
signature	签名	302C02141AD644D09C5E5DD8921367CA8D4AB69A0B 1C866002145594FDADD13493405DA21CE6373FC2B5 176AA38C	采用 DSA 算法

4.1.3.2 License 分发策略

- DEMO - 演示版
  - 基本功能
  - 固定使用期限 30 天
  - 不能升级和更新 license
  - 主要用于推广
- TRIAL - 试用版
  - 标准功能版本
  - 根据用户需要可设定试用期限、超过期限则提示用户升级证书
  - 可通过更新 license 的方式升级为标准版或企业版
- STANDARD - 标准版
  - 标准功能版本.
  - 根据用够需要设定使用期限、超过期限则无法继续使用，提示用户延长使用时间.
  - 使用期限内可升级.
  - 可通过更新 license 的方式升级为企业版
- ENTERPRISE - 企业版
  - 全部功能版本.
  - 根据用够需要设定使用期限、超过期限则无法继续使用，提示用户延长使用时间.
  - 使用期限内可升级.

- 可通过更新 license 的方式降级为标准版

➢ DEVELOPMENT - 开发版
- 全部功能版本.
- 无使用期限.
- 可升级.
- 提供二次开发 API

5. 系统功能模块

系统中功能模块按业务类别进行分解，通常分为 2 层。

➢ 第一层为域，按大业务组合具体的业务模块，通常不包含实际的业务。如：系统管理、系统辅助、个人门户等。

➢ 第二层为业务模块，每个模块通常代表一类业务。如“系统管理”中包含：用户管理、角色管理等。

业务模块采用统一格式的编号（域编号+模块编号）进行唯一性定义，同时该编号又可作为模块菜单权限的标识。

业务模块下面为具体的功能点，即具体的业务操作功能项。为各业务模块特有，不具有通用性。

（具体业务模块设计请查看各模块设计文档）

6. 系统权限模型

6.1 基本模型

权限基于 RBAC0(Core RBAC)模型设计。基本元素：访问主体（用户、角色），访问客体（资源、操作），许可权。

在本系统中资源指代业务模块，操作指代模块所具有的功能点（即某个操作项）

由于各模块功能点差异性比较大，其权限定义上采用数字编号的方式，由各模块独立定义。如 1 在模块 1 中代表新建，到模块 2 中可能代表审核。

功能项按对数据的操作类型又分为：增、删、改、查，每个功能点对应一个类型。可以通过该类型限制用户只能做查询操作或者只能增加操作

PS:功能点的设计唯一的缺点是不能按功能点对多个模块进行操作日志查询。

6.2 权限域

权限根据其业务特性，按适用范围在本系统中进一步被细分为如下 3 层：“**全局权限**”、“**模块权限**”、“**业务权限**”。

➢ 全局权限：即用户天生就具有的权限。该类权限通常对应用户个人可访问和使用的资源，如：通讯录、日记本、日报月报等。

➢ 模块权限：即通常意义的资源权限。该类权限需管理员授予该用户方能使用。

➢ 业务权限：即受业务依赖的权限。该类权限除了需要授予外，通常还受限于特定的业务逻辑，如“某个项目下的文档访问权限”，首先需要具有访问文档的权限，同时还需要有访问这个项目的权限。

“全局权限”和“业务权限”其实都是“模块权限”的特例，全局权限是为了简化那些用户固有权限的设置，使得无需对每个用户设定一个特殊的角色，同时简化了这些权限在变更时可能产生的角色信息变更。“业务权限”则是较为复杂的模块权限，由于涉及到业务数据，无法简单的由 RBAC 模型来解决。

6.3 资源分级

所有的可访问资源大致分为如下三类：开放的（Public）、受保护的（Protected）、私有的（Private）。

➢ Public：表示完全公开的资源，任何人均能访问，且访问结果一致。

➢ Protected：表示受权限保护的资源，只有授权用户方能访问，且不同的权限级别访问的结果也可能不一样。

- Private：表示个人私有资源，只有创建资源的用户个人方能访问。

6.4 权限分级

权限分配采用分级制度，即超级管理员具有所有权限；系统管理员的权限由超级管理员分配，系统管理员再建角色的话，可分配权限只能是其本身拥有权限的子集，依次类推。

7. 持续集成指南

7.1 持续构建服务器

持续构建服务器采用轻量级的 hudson，直接部署到 TOMCAT 中。

8. 机制指南

8.1 系统使用证书校验机制

未避免系统遭人非法复制和恶意传播，故设计该认证体制。采用 XML 格式定义加密前数据，采用 BASE64+DES 双重加密算法。

8.2 用户安全校验机制

1. 用户在进行登录时，一个会话(SESSION)连续输错 3 次则锁定该会话 30 秒，以防止有人恶意尝试。一个帐号的密码连续输错 8 次则强制锁定该帐号，以防止有人恶意破解密码（只能联系系统管理员解锁）。
2. 同一用户同时只能在一个地方进行登录，用户在另一机器登录时即强制注销前一个登录。
3. 采用登录时绑定机器 IP 或唯一索引号的方式，通过维护在线用户列表，在 SESSION 监听器中进行注销操作。
4. 采用过滤器限制重复提交和提交压力。

8.3 自动登录机制

通过 cookie 实现操作员的自动登录，cookie 中只保存操作员的 ID 号，而不保存密码，以保障数据的安全性；同时操作员修改密码后仍能自动登录。

8.4 数据传递机制

采用 WebService 作为各系统数据交互的机制，同时支持 Text/XML。

8.5 异常处理机制

由于采用了松散架构，各子系统可能部署在不同环境下，数据通信时的异常处理就尤为重要。

8.6 数据清理、更新机制

通过一个后台的定时任务，在每日凌晨对数据库中的过期数据进行清理和更新。

9. 开发、测试、生产

9.1 数据库

根据开发、调试、生产环境，数据库也分为 3 个。

- 开发：项目组使用统一的**开发库**，以便在一个统一的环境下确保系统的完整性。
- 测试：根据不同需求可使用统一的**测试库**或是各自使用独立的**测试库**，测试库的要求为：数据库结构与开发库一致，测试数据则由外部准备。同时另备一个**模拟生产库**模拟一个真实的应用环境，数据库

结构随系统版本的发布同步更新，但是数据将保留。

- 生产：通常双机热备，备机为主机昨日的数据。系统版本更新应首先在备机上测试通过才能更新主机。

9.2 服务器

根据开发、调试、生产环境，服务器也分为 3 个。

- 开发：通常为本机，应用服务器使用轻量级的 Tomcat。
- 测试：通常为独立主机，可以是个人电脑，条件允许下使用与生产环境相同的配置以便进行压力测试。应用服务器需与生产环境相同以防止个别服务器本身的不兼容问题。
- 生产：通常为独立主机，操作系统以 Linux 为宜，应用服务器以当前稳定版为宜，同时双机热备以保障 7 X 24 小时正常运行。