**广州烽燧信息科技有限公司**

**技术部版本管理规范**

**技术部内部文档**

**2017年2月**

**版本控制信息**

| 版本 | 日期 | 拟稿和修改 | 说明 |
| --- | --- | --- | --- |
| 1.0 | 2017-02-19 | Bill | 拟稿发布 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**目录**

[文档类别使用对象 5](#_Toc475315711)

[1．引言 5](#_Toc475315712)

[1.1目的 5](#_Toc475315713)

[1.2范围 5](#_Toc475315714)

[1.3术语定义 5](#_Toc475315715)

[2．版本管理 7](#_Toc475315716)

[2．1版本标识方法 7](#_Toc475315717)

[2．1．1版本标识说明 7](#_Toc475315718)

[2．2目录结构 7](#_Toc475315719)

[2.2.1 trunk 8](#_Toc475315720)

[2.2.2 branches 8](#_Toc475315721)

[2.2.3 tags 8](#_Toc475315722)

[2.2.4 files 8](#_Toc475315723)

[2.2.5 script 8](#_Toc475315724)

[2.2.6 sql 8](#_Toc475315725)

[2．3权限控制管理 9](#_Toc475315726)

[3．版本管理流程 10](#_Toc475315727)

[3.1 基本流程图 10](#_Toc475315728)

[3.2版本创建 10](#_Toc475315729)

[3.3新版本发布 11](#_Toc475315730)

[3.4版本升级原则 11](#_Toc475315731)

[3.5紧急变更方案 11](#_Toc475315732)

[3.6 角色定位说明 12](#_Toc475315733)

[4．备份管理 12](#_Toc475315734)

[5．SVN使用规范 13](#_Toc475315735)

[5.1先更新，再提交 13](#_Toc475315736)

[5.2一个提交尽量对应一个逻辑问题 13](#_Toc475315737)

[5.3多提交 13](#_Toc475315738)

[5.4不要提交不能通过编译的代码 13](#_Toc475315739)

[5.6每次提交必须书写明晰的标注 14](#_Toc475315740)

[5.7提交时注意不要提交本地自动生成的文件 14](#_Toc475315741)

[5.8合并代码注意先后顺序 14](#_Toc475315742)

# 文档类别使用对象

文档类别

该文档是为技术部开发提供的版本管理规范性文件。

使用对象

该文档使用对象为技术部管理人员和开发人员，以及其他相关人员。未经许可，该文档不得提供给上述规定对象以外的人员阅读或使用。

# 1．引言

## 1.1目的

本文档是为规范代码和文档版本管理而制定的。

## 1.2范围

本文档为技术部各人员提供有关版本管理规范的相关内容，包括：

1. 版本标识方法
2. 版本管理流程
3. 角色定位
4. 版本管理规范

## 1.3术语定义

**SVN**

Svn是一个开源的版本控制系统Subversion的简称

文档

上线所需的相关文档，包括部署手册，源码修改清单列表等

脚本

上线所需的相关脚本，包括编译脚本等

SQL语句

上线所需的相关SQL语句，包括建表语句等

配置管理

标识和确定系统中配置项的过程，在系统整个生存周期内控制这些项的投放和更动，记录并报告配置的状态和更动要求，验证配置项的完整性和正确性。

软件配置

软件的具体形态在某时刻的瞬时影像。

配置项

软件配置管理的对象称为配置项，如：源码。

基线

软件生存周期中各开发阶段末尾的标记，它的作用是把各阶段工作的划分更加明确化，使本来连续的工作在这些点上断开，使之便于检验和肯定阶段成果。

版本控制

通过svn co把分支文件夹拷贝到开发环境进行开发，并进行版本控制

版本管理

根据需求，创建开发所需的分支

标签管理

为测试版本，上线版本创建标签

版本更新

通过svn ci定期备份修改内容，或通过svn update更新当前所开发的源码，或通过svn merge把主干新增内容更新至分支

版本测试

通过svn export校验源码，进行源码的比对，测试

版本修复

对当前测试或上线版本出现的问题进行修复

版本冲突

由于修改了同一个文件，所以svn ci，svn merge以及svn up时会报错，造成了版本冲突问题。

# 2．版本管理

## 2．1版本标识方法

为了使工作规范化、统一化，各系统实行的版本标识管理方法分为：上线版本，测试版本，修复版本，文档版本，脚本版本以及sql语句版本。

### 2．1．1版本标识说明

Live版本：在生产环境上运行的正式版本。

Staging(测试)版本：在UAT（用户验收）环境上运行的测试版本。

修复版本：在生产环境上用于修复当前版本的补丁版本。

Live版本号为tag\_release\_加版本号，　版本号分2节：主版本号从1开始，次版本号从0开始， 比如tag\_release\_1.0，说明该版本为已发布到生产环境的1.0版本，如果需要修复某个发布版本，分支的版本标识为dev\_版本号\_bugfix, 比如dev\_1.0\_bugfix, 说明该版本为tag\_release\_1.0的修复版本，如果此修复版本发布到生产环境后，需要生成版本标签，比如tag\_release\_1.1。

文档版本：上线版本对应的相关文档。

以“file”开头，版本号放后。

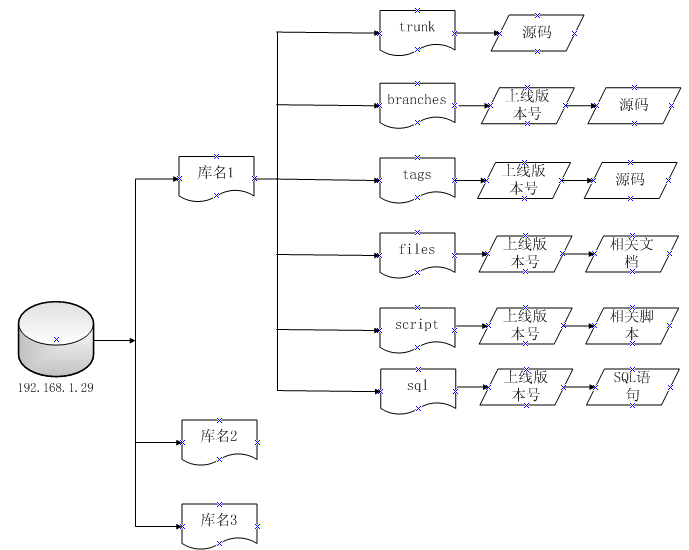
脚本版本：上线版本对应的相关脚本。

以”spt”为开头，版本号放后

sql语句版本：上线版本对应的sql语句。

以“sql”为开头，版本号放后。

## 2．2目录结构





### 2.2.1 trunk

主干文件夹，存放的是当前系统的最新源码

### 2.2.2 branches

分支文件夹，存放的是当前开发和历史开发的分支文件夹的源码。

### 2.2.3 tags

标签文件夹，存放的是当前上线版本和历史版本的源码。

### 2.2.4 files

文档文件夹，存放的是当前上线版本和历史版本的相关文档。

### 2.2.5 script

脚本文件夹，存放的是当前上线版本和历史版本的相关脚本。

### 2.2.6 sql

sql语句文件夹，存放的是当前上线版本和历史版本的相关sql语句。

## 2．3权限控制管理

为保障版本的安全性，一致性，以及防止意外修改，必须对不同的文件夹设置不同的访问权限。

文件夹权限类别：只读权限，读写权限。

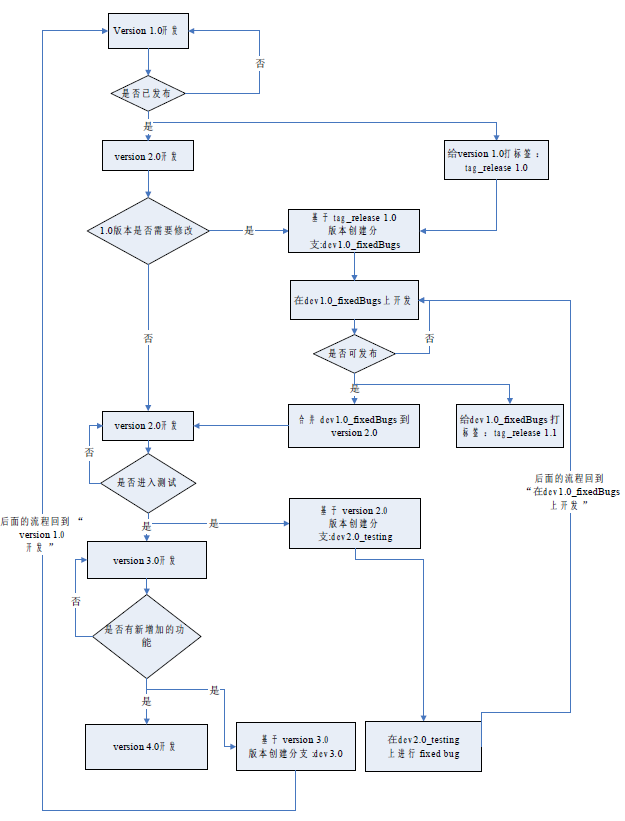
用户类别：开发人员、QA、项目经理等。

为了控制不同的使用权限，根据要求在服务器上分别建立不同的用户，针对不同的配置项所在目录分配不同的权限。

为了便于管理，应以表格的形式列出人员与管理对象的访问关系（用户权限清单）。

# 3．版本管理流程

## 3.1 基本流程图



## 3.2版本创建

1. 项目创建时，根据项目分析确定开发环境及发布环境结构，开发负责人在版本管理库里创建库名及相关目录，并记录在项目管理工具的wiki上或发邮件通知开发人员, 首个版本一般保存在/库名/trunk下。
2. 开发人员根据开发负责人提供的版本控制代码路径从版本库的trunk文件夹内checkout到开发服务器建立版本控制，进行程序开发。

## 3.3新版本发布

1. 根据项目需求，开发人员开发完成后，提交回版本库，如果当前为主干，将了解是否有分支需要合并一起发布，如果有，需要将最新的分支文件合并至主干的工作拷贝，合并完成后，进行diff的比对，确认没问题后，最后把主干的工作拷贝提交至版本库。
2. 开发人员告知开发负责人当前开发完成，开发人员需要准备部署手册，源码修改清单等相关文档，编译脚本，SQL语句。开发负责人同意后发布到测试环境，通知测试人员测试
3. 测试通过后，测试人员通知开发人员和开发负责人，准备上线。
4. 开发人员上线完成后，通知测试人员做回归测试，测试通过后，通知开发负责人
5. 开发负责人当前主干版本创建标签文件夹。

6. 测试未通过，开发人员对代码进行二次开发，待开发完成后，直至上线

## 3.4版本升级原则

版本升级应严格纳入版本管理的控制之下。应当谨慎地控制版本的升级，保障高版本的向下兼容性，或提供严格定义的升级方法。

在下面几种情况下，进行版本演化和升级：

1. 当系统有重大的需求，需要较大的改进或修改时，主版本号为新版本+1。
2. 当系统有重大的BUG问题时，次版本+1。
3. 对于改动量比较少的，如修复小问题之类的，可以从当前正在开发分支支中，进行改进或修改，和下一个新版本一起上线。

## 3.5紧急变更方案

触发条件：下一个上线版本已经并入了主干，需要在下一个上线版本前插入一个补丁版本。

1. 变更需求确认后，开发负责人或开发人员从对应的版本库找到对应下一个上线版本，如果没有分支，需要从tags里找到相应版本创建分支文件夹到branches，并将分支信息更新到wiki。
2. 开发人员根据提供的分支文件名从版本库的分支文件夹内checkout到开发服务器建立版本控制，进行程序开发。
3. 开发人员告知开发负责人分支已经完成，开发人员需要准备部署手册，源码修改清单等相关文档，编译脚本，SQL语句。并了解是否有不同的分支需要发布，如果有需要协商合并流程，开发负责人同意后发布到测试环境，通知测试人员测试。
4. 测试通过后，测试人员通知开发人员和开发负责人，准备上线。
5. 开发负责人可以根据当前修改情况考虑是否需要将修改复分支合并回最新主干，也可以在最新版本发布前再进行合并。
6. 开发人员上线完成后，通知测试人员做回归测试，测试通过后，通知开发负责人
7. 开发负责人当前发布版本创建标签文件夹。

## 3.6 角色定位说明

开发人员需要做：

[版本控制](#_1.3术语定义)，[配置项](#_1.3术语定义)，[文档](#_1.3术语定义)，[脚本](#_1.3术语定义)，[SQL语句](#_1.3术语定义)，[版本更新](#_1.3术语定义)，[版本修复](#_1.3术语定义)，[版本冲突](#_1.3术语定义)

测试人员需要做：

[版本测试](#_1.3术语定义)

开发负责人员需要做：

[配置管理](#_1.3术语定义)，[基线](#_1.3术语定义)，[版本管理](#_1.3术语定义)，[标签管理](#_1.3术语定义)

# 4．备份管理

为了保证文档的最大可恢复性，要随时及定期地进行备份工作。

1. 随时备份：
2. 开发人员如果当天不能提交代码，需要将自已当日修改的源文件在公共服务器自己的目录下进行备份。
3. 建议备份采用循环备份。
4. 定期备份
5. 备份形式为硬盘备份和光盘备份。硬盘备份时，要备份在独立的硬盘上；光盘备份时，要将光盘存放在可靠的地方。
6. 备份周期视各系统的具体情况而定。如果处于开发阶段，每日应对待开发项进行备份，一般为每晚２点至5点之间；如果处于其它阶段，根据具体情况而定，但周期不能超过两周。
7. 备份要由版本管理员负责，备份原则应是保证文档的最大可恢复性。
8. 对于历史版本或某用户的特殊版本，如果无特殊原因不再进行修改的话，建议用光盘进行备份，而且应有备份盘说明文件BACKUP.TXT。该文件应该记录以下内容：本次备份时间，备份内容，执行人。
9. 版本管理员每周一自检一次自动备份是否正常，技术部负责人每月１号检查一次。

# 5．SVN使用规范

## 5.1先更新，再提交

SVN更新的原则是要随时更新，随时提交。当完成了一个小功能，能够通过编译并且自己测试之后，谨慎地提交。   
 如果在修改的期间别人也更改了svn的对应文件，那么commit就可能会失败。如果别人和自 己更改的是同一个文件，那么update时会自动进行合并，如果修改的是同一行，那么合并时会产生冲突，这种情况就需要同之前的开发人员联系，两个人一起协商解决冲突，解决冲突之后，需要两人一起测试保证解决冲突之后，程序不会影响其他功能。   
 在更新时注意所更新文件的列表，如果提交过程中产生了更新，则也是需要重新编译并且完成自己的一些必要测试，再进行提交。

## 5.2一个提交尽量对应一个逻辑问题

一次提交多个问题的修改或者一次提交多个模块的不同改动是不被允许的，这样也就失去了版本管理的意义。我们要尽量做到一次提交对应一个问题，涉及一个模块。

如果某次修改必须涉及到多个模块，请谨慎检查代码和库上的代码情况，确保不会因为你的上传导致无法回退。

## 5.3多提交

每次提交的间歇尽可能地短，以几个小时的开发工作为宜。例如在更改UI界面的时候，可以每完成一个UI界面的修改或者设计，就提交一次。在开发功能模块的时候，可以每完成一个小细节功能的测试，就提交一次，在修改bug的时候，每修改掉一个bug并且确认修改了这个bug，也就提交一次。我们提倡多提交，也就能多为代码添加上保险。

## 5.4不要提交不能通过编译的代码

代码在提交之前，首先要确认自己能够在本地编译。如果在代码中使用了第三方类库，要考虑到项目组成员中有些成员可能没有安装相应的第三方类库。项目经理在准备项目工作区域的时候，需要考虑到这样的情况，确保开发小组成员在签出代码之后能够在统一的环境中进行编译。

## 5.6每次提交必须书写明晰的标注

在一个项目组中使用SVN，如果提交空的标注或者不确切的标注将会让项目组中其他的成员感到很无奈，项目经理无法很清晰的掌握工作进度，无法清晰的把握此次提交的概要信息。在发现错误后也无法准确的定位引起错误的文件。所以，在提交工作时，要填写明晰的标注，能够概要的描述所提交文件的信息，让项目组其他成员在看到标注后不用详细看代码就能了解你所做的修改。

## 5.7提交时注意不要提交本地自动生成的文件

例如eclipse中的.classpath文件，Windows生成的缩略图Thumbs.db，项目编译生成的临时文件.obj, .class等等。如果项目中没有进行这方面的配置来强行禁止提交这样的文件，请自觉不要提交这样的文件。提交了这样的文件后，别人在更新后就可能与本地的环境冲突从而影响大家的工作。

## 5.8合并代码注意先后顺序

完成单元测试后，进行集成测试时，才可合并入主干，如果要自己做集成测试，则可以把主干合并入分支进行测试。

在多个分支开发的情况下，后上线的分支必须等前上线的分支合并入主干后测试通过了，再可并入主干。