|  |
| --- |
| 上海与德通讯技术有限公司 |
| 代码修改规范以及提交流程 |
| 代码提交规范和流程 |

|  |
| --- |
|  |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Issue** | **Description** | **Author** |
| 2014/10/11 | 1.0 | 初始版本 | 冯松 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[1. Abstract 4](#_Toc400982311)

[2. 代码修改规范 4](#_Toc400982312)

[2.1. 注释的添加 4](#_Toc400982313)

[2.1.1. 代码的修改 4](#_Toc400982314)

[2.1.2. 注释的添加 4](#_Toc400982315)

[2.1.3. 宏控的添加以及使用 5](#_Toc400982316)

[2.2. 代码提交流程 6](#_Toc400982317)

[2.2.1. 获得修改文件列表 6](#_Toc400982318)

[2.2.2. 添加需要上传的文件 7](#_Toc400982319)

[2.2.3. 提交需要上传的文件 8](#_Toc400982320)

[2.2.4. 上传代码到远端服务器 11](#_Toc400982321)

[2.3. 代码合入服务器 12](#_Toc400982322)

[2.3.1. 添加reviewer 12](#_Toc400982323)

[2.3.2. 代码重新上传 13](#_Toc400982324)

[2.3.3. 回退代码节点 13](#_Toc400982325)

# Abstract

此文档介绍软件工程师修改代码规范以及提交流程。

# 代码修改规范

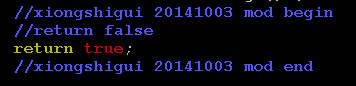
## 注释的添加

因为同一套代码，包含的项目很多，对应的需求很多，所以为了便于代码维护以及管理，新添加的代码以及修改的代码，均需要添加自己相应的注释。

### 代码的修改

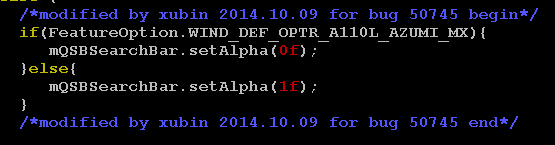
因新增需求或着平台bug，经常会有代码的修改，代码的修改除了要保持代码的可读性以及代码的整洁性外，还需要符合以下几点要求：

1、修改原有代码时，需要保留原有内容，以便后续检测。如图一



图一

2、对于项目定制需求，需要添加宏控，来管控新增代码，以免影响其他项目的代码流程。如图二



图二

### 注释的添加

对于修改的代码，新增的代码以及新增的文件，均需要添加自己的注释，以便查询和阅读。

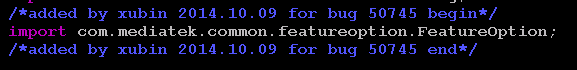
1、修改代码后，需要使用自己的注释把修改的代码上下包裹起来，格式可参照图一，图二。

注：上下注释中的内容，可根据自己习惯编写。

2、新添加的代码同样需要使用自己的注释把新增的代码上下包裹起来，格式可参照图一，图二。

注：上下注释中的内容，可根据自己习惯编写。

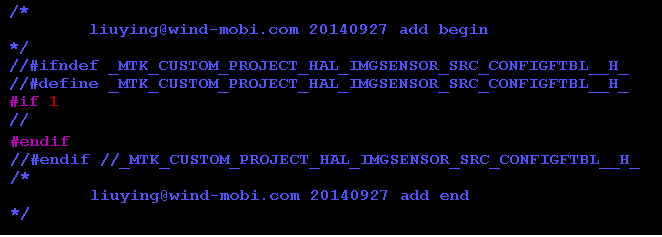
3、因宏控需要或调用函数需要，经常需要import其他的类，新添加的import同样需要使用自己的注释把新增的import上下包裹起来，格式可参照图三。



图三

注：上下注释中的内容，可根据自己习惯编写。

4、因为新调试设备或者新添加功能，经常会伴随着新的文件的引入，但是因为大版本升级的时候，会不时的对文件系统做出调整，有很多文件的新增和删除，这样就要求我们在自己新增文件的时候，也是需要在文件开头与文件结尾处添加自己的注释，以便文件不会被删除掉，格式如图四。



图四

注：上下注释中的内容，可根据自己习惯编写。

### 宏控的添加以及使用

需求的定制以及功能的实现，会有各种宏控的新增与使用，当一个项目建立时，会添加相应的一个项目宏控，如图二中的WIND\_DEF\_OPTR\_A110L\_AZUMI\_MX就是A110L项目中的一个项目宏控。

对于宏控的添加以及使用，有以下几点需要注意：

1、在Java文件中使用项目宏控，需要保证该文件中已import图三中所示的类名，否则会报错。

2、在mediatek/build/tools/javaoption.pm文件中包含的宏控名称，才可以在Java文件中使用。

3、在mediatek/config/xxx/ProjectConfig.mk文件变量AUTO\_ADD\_GLOBAL\_DEFINE\_BY\_NAME中的宏控，才可以在.cpp文件中调用。

4、新增宏控需要在以下几个文件中添加：

（1）mediatek/config/common/ProjectConfig.mk 宏控置为no

（2）mediatek/config/xxx/ProjectConfig.mk 宏控置为yes

（3）如果需要在.cpp文件中使用，需要在mediatek/config/xxx/ProjectConfig.mk文件中的变量AUTO\_ADD\_GLOBAL\_DEFINE\_BY\_NAME后添加

（4）如果需要在java文件中使用，需要在mediatek/build/tools/javaoption.pm文件中添加

5、对于一个独立的功能，尽量添加一个功能宏控，以便代码的独立性以及功能的可移植性。

6、当ProjectConfig.mk中的宏控发生修改或者新增一个宏控后，整个工程必须new后，才能生效。

7、mk文件，cpp文件，h文件，java文件中宏控使用的方式有所不同，需要注意相应的语法规则。

## 代码提交流程

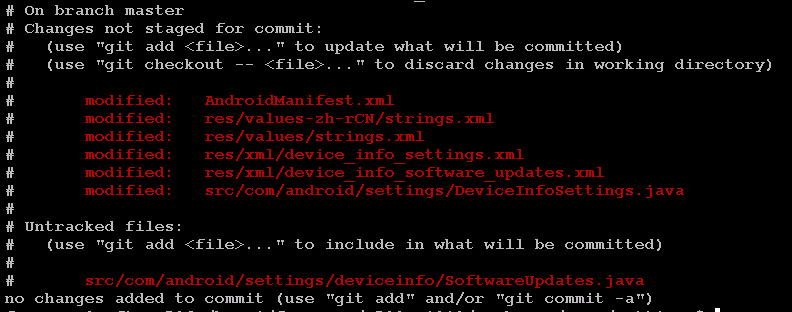
代码修改并验证完成后，需要提交远端服务器。

### 获得修改文件列表

在修改的模块目录下，执行命令：

$git status .

会列出发生变动的文件列表，如图五：



图五

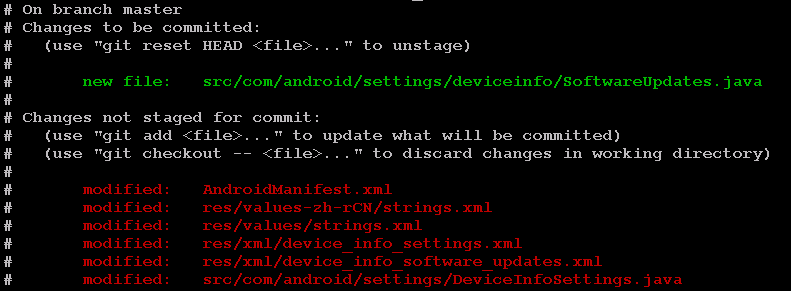
其中Untracked files:一栏为新添加的文件。

### 添加需要上传的文件

对于新添加的文件必须执行以下命令：

$git add src/com/android/settings/deviceinfo/SoftwareUpdates.java

执行好后，新增加的文件的字体会变为绿色，如图六：



图六

然后执行以下命令，添加需要上传的改动的文件：

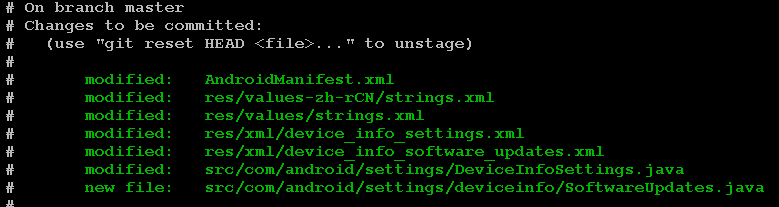
$git add src/com/android/settings/deviceinfo/SoftwareUpdates.java AndroidManifest.xml

res/values-zh-rCN/strings.xml

以上命令中add后添加的是需要上传的文件名，多个文件直接使用空格隔开，如果当前目录下有改动的文件均需要提交的话，可使用以下命令，一次提交。

$git add .

add后，所以改动的文件均会变为绿色，如图七：



图七

### 提交需要上传的文件

如果仅需要提交改动的文件，即在图五步骤时，只需要执行以下命令上传对应的文件即可：

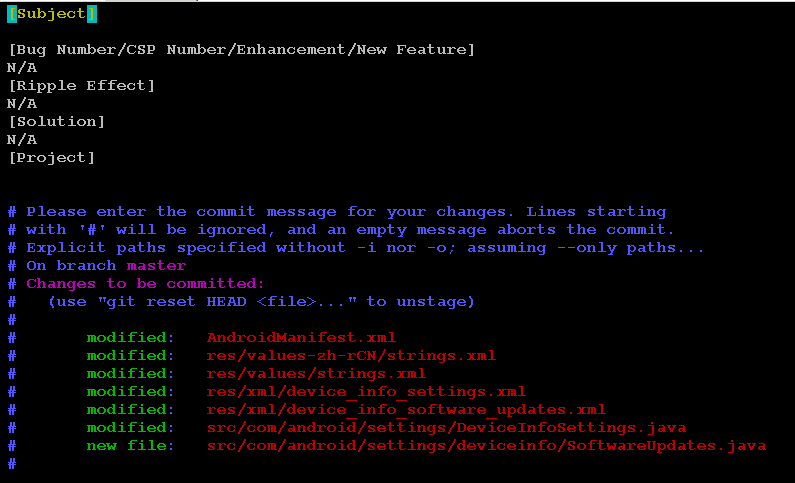
$git commit src/com/android/settings/deviceinfo/SoftwareUpdates.java AndroidManifest.xml

res/values-zh-rCN/strings.xml

如果当前目录仅有改动的文件，不包含新增的文件，或者是执行到了图七步骤时，也可执行如下命令来上传当前目录下的所有改动：

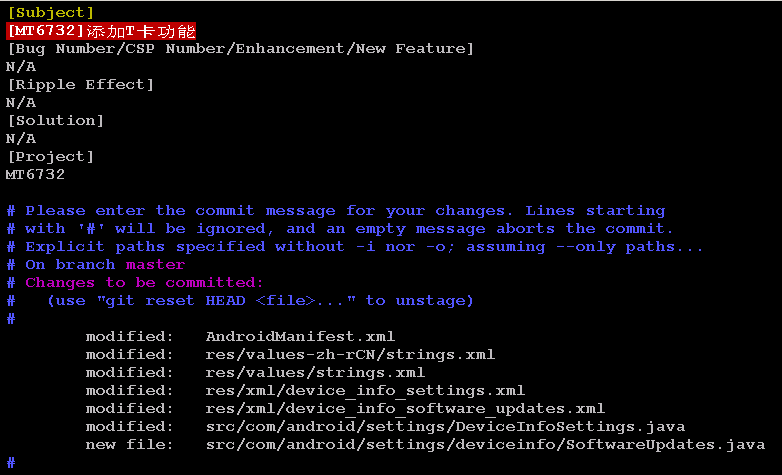
$git commit .

执行命令后，会打开vi界面，来填写改动的相应信息，如图八：



图八

填写完对应信息后，如图九，然后保存退出。



图九

图九中每一栏需对应添加的内容：

[Subject]-----项目名称以及修改内容

[Bug Number/CSP Number/Enhancement/New Feature] ----修改问题对应的bug号或patch号

[Ripple Effect]-----修改代码对应的波及分析

[Solution]--------问题的解决版本

[Project]--------项目名

以上各项填写完成后，需要把图中修改文件名称前面的#号去掉。

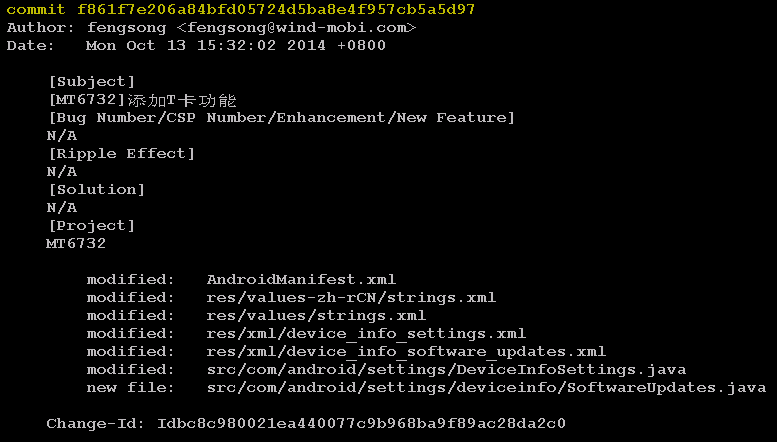
退出后，修改就已经上传成功。如图十：



图十

这时可以通过以下命令查看新提交代码的log，如图十一：

$git log .



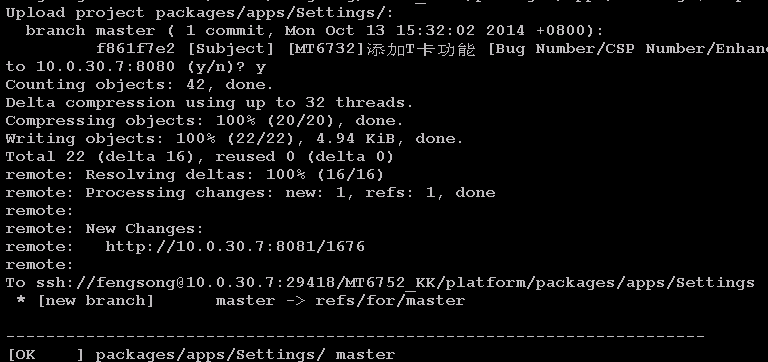
图十一

### 上传代码到远端服务器

在代码目录下执行以下命令，即可把之前步骤中提交的修改，上传到远端服务器，

$repo upload .

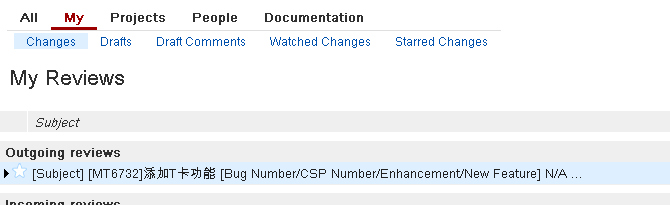
代码提交后，待会有图十二中所示的提示后，才代表提交成功。



图十二

## 代码合入服务器

上述步骤完成后，在网页端<http://10.0.30.7:8080>中即可看到刚才的提交记录，如图十三

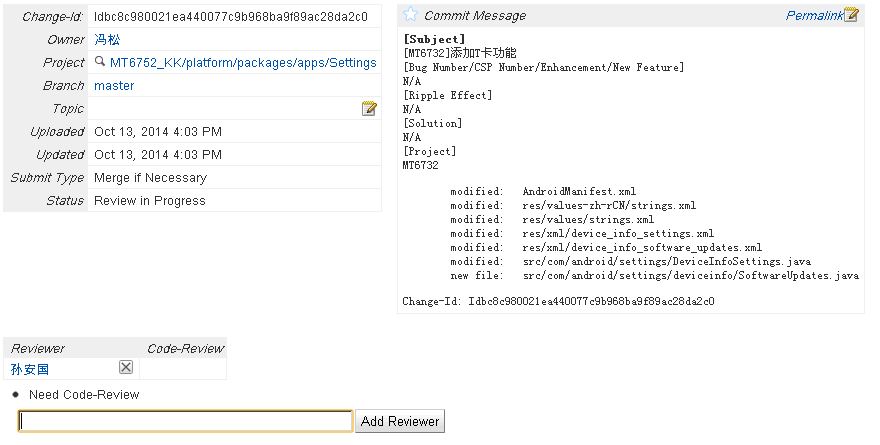


图十三

注：目前代码服务器有30.6与30.7两台，所以网站有所不同，因MT6372代码在30.7上，所以以30.7为例，操作方式与规则相同。

### 添加reviewer

因并非所有人都有权限上传代码，所以需要添加reviewer来检测代码以及上传代码，如图十四，



图十四

如果没有问题，则代码上传步骤结束，后续工作为SCM上传代码。

### 代码重新上传

Review时发现错误，或者格式有问题，或者代码上传有冲突，就需要abandon掉提交，重新上传。

点击Abandon Change按钮，忽略掉提交，如图十五中所示按钮：



图十五

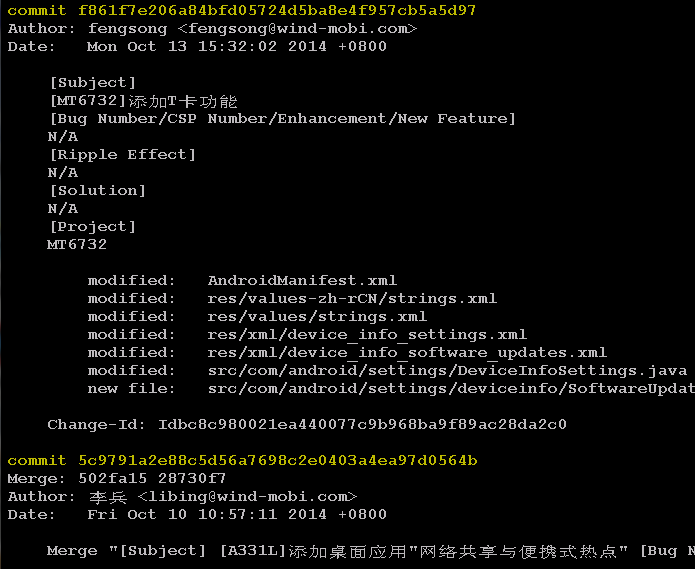
### 回退代码节点

在网页中abandon掉代码提交后，还需要在本地代码中回退节点，来重新提交。

在代码目录下，执行以下命令来查询log信息：

$git log .

查询出有问题的提交的前一次提交，如图十六：



图十六

执行以下命令，把代码回退到之前的节点：

$git reset –hard 5c971a2e88c5d56a7698c2e0403a4ea97d0564b

--hard后添加的参数为需要回到的节点的commit ID

回退成功后，提示如图十七：



图十七

然后重新合入修改，待修改好后，重新按照之前的步骤，提交修改的代码。