目录

[第1章 概述 2](#_Toc41309544)

[1.1 版本类型 2](#_Toc41309545)

[1.2 版本号 2](#_Toc41309546)

[1.3 版本文件 2](#_Toc41309547)

[第2章 管理员维护 3](#_Toc41309548)

[2.1 修订 3](#_Toc41309549)

[2.2 发布 3](#_Toc41309550)

[第3章 用户操作指南 4](#_Toc41309551)

[3.1 配置 4](#_Toc41309552)

[3.2 新建分支 4](#_Toc41309553)

[3.3 升级分支 4](#_Toc41309554)

[3.4 安装包 4](#_Toc41309555)

[第4章 附录 5](#_Toc41309556)

[4.1 Git默认编辑器VIM 5](#_Toc41309557)

[4.2 Tag 5](#_Toc41309558)

[4.3 合并到工作区 5](#_Toc41309559)

[4.4 通过变基将多个提交合并 6](#_Toc41309560)

[4.5 VMS提交失败的处理 6](#_Toc41309561)

# 概述

## 版本类型

版本分为标准版、内测版和定制版。

标准版即Tag，由管理员维护，公开但只读。

内测版即0.0.0分支，由管理员维护，不公开。

定制版为针对用户的特殊定制分支，公开且可读写。可以基于标准版或定制版创建新的独立定制版分支。

## 版本号

主版本号

修订号

次版本号

内建号

1xx.2xx.1xx.3xx

主版本号—Major，内测版进行非兼容性升级时手动递增此版本号。

次版本号—Minor，内测版进行兼容性升级时手动递增此版本号。

内建号—Build，定制版专用编号，标准版和内测版固定为0。

修订号—Revision，标准版固定为0。内测版和定制版修订后，VMS提交时自动递增。

## 版本文件

### 库版本号

Git管理的顶层文件夹称为版本库，版本号保存在顶层文件夹Version.json的VersionNow字段。版本库下管理多个源程序工程，任意一个工程升级时，库版本号均会递增。

### 工程版本号

Git版本库下可管理多个工程，每个工程均有独立的版本号。

1. C#工程，版本号定义于工程目录 \Properties\AssemblyInfo.cs文件中的AssemblyFileVersion字段；
2. C工程，版本号定义于工程目录\Inc \Version.h文件中的VERSION字符串。

# 管理员维护

## 修订

1. 使用VMS检出0.0.0分支。
2. 修改源文件维护升级此版本，升级完成或切换到其它分支前，使用VMS提交此版本。
3. 向下兼容性升级直接提交即可，VMS自动处理版本号。
4. 非向下兼容性升级，须手动更改相应[**工程**](#_工程版本号)**次版本号**，并将修订号改为-1，然后使用VMS提交。

## 发布

当内测版修订到特定成熟版本后，使用VMS的“发布”按钮发布新标准版本。

填写提交信息时，应综合此前未发布的升级内容，提交后VMS自动发布标准版。标准版仅用于通过Tag提供新版本号，不得在此分支上进行任何修改。

# 用户操作指南

## 配置

1. 首次使用时，在设置界面，输入汉字姓名作为用户名。
2. 按需配置其它设置

## 新建分支

1. 在基础分支所在行，点击“新建分支”按钮。
2. 填写订单信息和版本说明，点“提交”创建新分支。
3. 在新分支完成更改后，点“提交”上传更改。

## 升级分支

1. 在VMS版本列表选中需升级的版本，点击“检出版本”按钮。
2. 点击“打开”按钮打开Visual Studio工程。如包括多个工程，可通过Visual Studio🡪团队资源管理器🡪主页🡪解决方案列表🡪打开工程。
3. 通过”浏览”按钮，打开库目录，可手动打开非Visual Studio工程。
4. 修改完成后，使用VMS提交更改。

## 安装包

1. 删除安装包目录旧文件.
2. 点击VMS界面“安装包”按钮，自动生成安装文件，并整合到安装包目录。

# 附录

## Git默认编辑器VIM

### 两种工作模式：

在编辑模式下按下ESC键，回退到命令模式；在命令模式下按i，进入编辑模式。

1. 命令模式：接受、执行 vim操作命令的模式，打开文件后的默认模式；
2. 编辑模式：对打开的文件内容进行增、删、改操作的模式；

### 创建、打开文件：

1.输入 touch 文件名，可创建文件。

2.使用 vim 加文件路径（或文件名）的模式打开文件，如果文件存在则打开现有文件，如果文件不存在则新建文件。

### 保存文件：

1.在编辑模式下编辑文件

2.按下ESC键，退出编辑模式，切换到命令模式。

3.在命令模式下键入"ZZ"或者":wq"保存修改并且退出 vim。

4.如果只想保存文件，则键入":w"，回车后底行会提示写入操作结果，并保持停留在命令模式。

### 放弃所有文件修改：

1.放弃所有文件修改：按下ESC键进入命令模式，键入":q!"回车后放弃修改并退出vim。

2.放弃所有文件修改，但不退出 vi，即回退到文件打开后最后一次保存操作的状态，继续进行文件操作。按下ESC键进入命令模式，键入":e!"，回车后回到命令模式。

## Tag

### 新建

$ git tag 1.2.0

### 删除

$ git tag -d 1.0.0

$ git push origin :refs/tags/1.0.0

### 推送

$ git push origin –tags

### 删除所有远程标签

必须在删除本地标签之前运行

$ git tag -l | xargs -n 1 git push --delete origin

### 删除所有本地副本

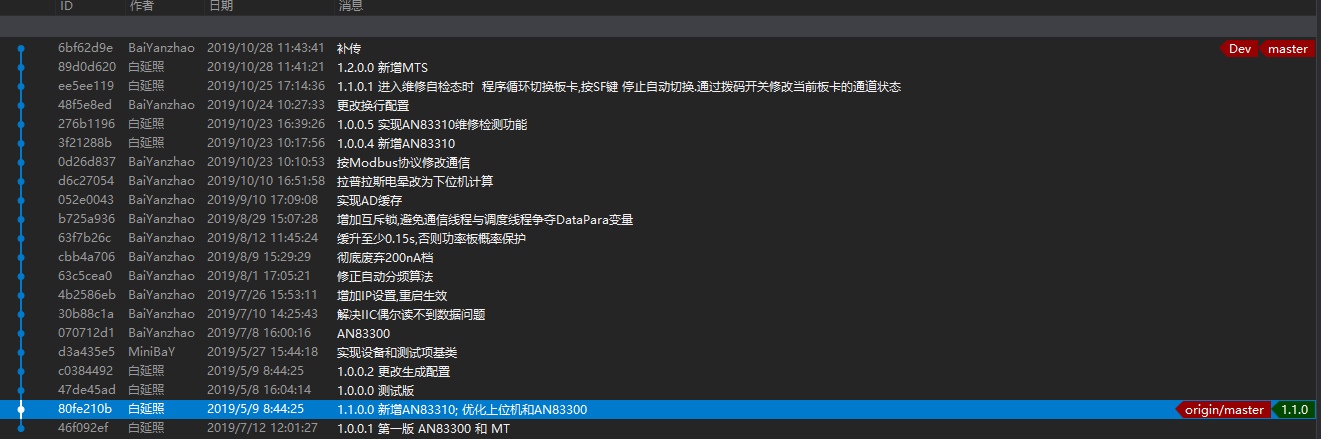
$ git tag | xargs git tag -d

## 合并到工作区

$ git checkout master

$ git merge --squash Dev

## 通过变基将多个提交合并



1. 打开git-bash，将图中无Tag的提交合并（ID80fe210b–66f62d9e），输入以下命令，进入[VIM](#_Git默认编辑器VIM)变基编辑界面。

$ cd "C:\Users\BaiYanzhao\source\repos\MTx"

$ git rebase -i 80fe

**注:此处80fe为origin/master行ID的简写.**

1. 在编辑界面，点键盘“I”键进入输入状态，最后第一行为“pick”或“r”，其余行根据说明改为“d”或“s”。
2. 修改完成后，点键盘“ESC”，然后输入“:wq”后回车，完成保存退出。
3. 在接下来的[VIM](#_Git默认编辑器VIM)提交信息编辑界面，修改新的提交信息，修改完成后保存退出。

将其他分支

## Git误操作恢复

$ git reflog -n8

$ git branch recover\_branch[新分支] [commit\_id]

## VMS提交失败的处理

1. 点“浏览”按钮，打开工程文件夹。以文件夹名Software为例。

2. 关闭VMS，将文件夹顶层目录的隐藏文件夹“.git”删除，并将工程文件夹重命名为Bak。

3. 打开VMS，如果没有所需分支，则新建分支；如存在所需分支，则点"检出"按钮，检出分支，此时VMS自动生成Software文件夹。

4. 将Software文件夹下除隐藏文件夹“.git”外的文件全部删除；将Bak文件夹内全部文件剪切到Software文件夹；删除空文件夹Bak。

5. VMS界面。点“提交”，重新提交。