

版本管理系统



白延照

2019-7-12

目录

[第1章 概述 3](#_Toc23864757)

[1.1 版本类型 3](#_Toc23864758)

[1.2 版本号 3](#_Toc23864759)

[1.3 版本文件 4](#_Toc23864760)

[第2章 内测版 5](#_Toc23864761)

[2.1 修订 5](#_Toc23864762)

[2.2 发布 5](#_Toc23864763)

[第3章 标准版 6](#_Toc23864764)

[3.1 合并内测版 6](#_Toc23864765)

[3.2 同步内测版 7](#_Toc23864766)

[第4章 定制版 8](#_Toc23864767)

[4.1 配置 8](#_Toc23864768)

[4.2 使用 8](#_Toc23864769)

[4.3 安装包 8](#_Toc23864770)

[第5章 附录 9](#_Toc23864771)

[5.1 Git默认编辑器VIM 9](#_Toc23864772)

[5.2 新建Tag 9](#_Toc23864773)

[5.3 删除Tag 10](#_Toc23864774)

[5.4 通过变基合并提交 10](#_Toc23864775)

[5.5 VMS提交失败的处理 11](#_Toc23864776)

# 概述

## 版本类型

版本分为标准版、内测版和定制版。标准版由管理员维护，对外公开但只读。内测版由管理员维护，对外不公开。定制版公开并可读写。

### 标准版—master分支

管理员通过Visual Studio手动切换到此分支，通过内测版合并升级此版本。

通过设置Tag对VMS界面开放版本，通过删除Tag禁用过时的版本。

### 内测版—0.0.0分支

管理员通过VMS管理提交此版本。

### 定制版—以定制号命名的分支

定制版为针对用户的特殊定制分支，如 1.1.64.0，使用VMS创建和升级此版本。另外，还可以基于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

标准版进行非兼容性升级时递增此版本号。

### 次版本号—Minor

标准版进行兼容性升级时递增此版本号。

### 内建号—Build

标准版和内测版固定为0。每个新定制使用递增的内建号，作为定制编号。

### 修订号—Revision

标准版固定为0。内测版和定制版修订后，VMS软件提交时自动进行递增。

## 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。

# 内测版

## 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。
5. ~~新增工程或工程间互不兼容类升级，须手动~~**~~递增~~**[**~~库~~**](#_库版本号)**~~主版本号~~**~~"\_Major"，并将库次版本号"\_Minor"改为0，修订号"\_Revision"改为-1，然后使用VMS提交。~~

## 发布

当内测版升级测试稳定后，~~须手动递增~~[**~~库~~**](#_库版本号)**~~次版本号~~**~~"\_Minor"，并将修订号"\_Revision"改为-1，然后使用VMS提交，提交信息整合最近的升级内容。然后~~参考[合并内测版](#_合并内测版)发布新版本。

~~注意：发布前内测版只需修改Version.json文件，提交信息应综合此前未发布的升级内容。发布时master分支的提交信息与此信息完全相同。~~

# 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。当内测版修订到特定成熟版本后，合并到master分支，发布新版本。

完成master合并后，须同步更新内测版，保证内测版历史记录中包含所有Tag。

## 合并内测版

~~在~~[~~内测版提交~~](#_发布)~~后，按以下步骤发布更新。~~

~~除3.1.1必须使用git-bash外，其它亦可使用Visual Studio实现。~~

### 使用git-bash，合并0.0.0分支

BaiYanzhao@baiyanzhao MINGW64 /

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

BaiYanzhao@baiyanzhao MINGW64 ~/source/repos/MTx (0.0.0)

$ git checkout master

BaiYanzhao@baiyanzhao MINGW64 ~/source/repos/MTx (master)

$ git merge --squash 0.0.0

### 提交并推送master

使用VMS，填写提交信息并提交版本。提交信息应综合此前未发布的升级内容。

~~使用以下命令提交master，提交信息内容与0.0.0分支最近提交完全一致。~~

~~BaiYanzhao@baiyanzhao MINGW64 ~/source/repos/MTx (master)~~

~~$ git commit -m '1.2.0.0 升级XX功能'~~

~~BaiYanzhao@baiyanzhao MINGW64 ~/source/repos/MTx (master)~~

~~$ git push~~

### ~~新建并推送Tag~~

~~在bash命令行输入以下命令，新建以~~**~~库版本号~~**~~命名的Tag。~~

~~BaiYanzhao@baiyanzhao MINGW64 ~/source/repos/MTx (master)~~

~~$ git tag 1.2.0~~

~~BaiYanzhao@baiyanzhao MINGW64 ~/source/repos/MTx (master)~~

~~$ git push origin --tags~~

### 完成合并

最终master历史如下图所示。master分支历史记录按Tag顺序排列。

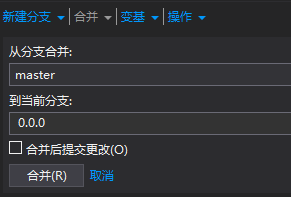
合并完成后，按[3.2](#_同步内测版)更新0.0.0分支。



## 同步内测版

为保证下次升级时，不重复解决master冲突，需从0.0.0分支合并master分支，确保0.0.0分支的历史记录中存在所有Tag。

1. 使用Visual Studio打开工程，切换到0.0.0分支，在分支界面选择合并。



1. 解决冲突后，使用VMS提交合并。

# 定制版

## 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。

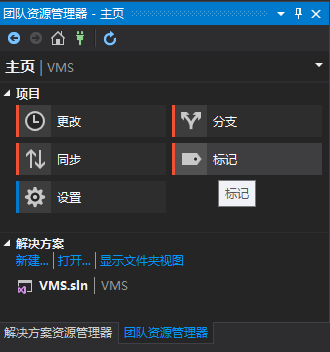
2.放弃所有文件修改，但不退出 vi，即回退到文件打开后最后一次保存操作的状态，继续进行文件操作：按下ESC键进入命令模式，键入":e!"，回车后回到命令模式。

## 新建Tag

1. 打开历史记录界面，在需要常见Tag的行右键，选择“创建标记”。



1. 输入指定的版本号作为新标记名称
2. 打开团队资源管理器主页，选择“标记”按钮，进入标记管理界面。



1. 点“全部推送”按钮，或在新建的标记处右键“推送”，将新标记推送到服务器

## 删除Tag

BaiYanzhao@baiyanzhao MINGW64 /c/users

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

BaiYanzhao@baiyanzhao MINGW64 ~/source/repos/MTx (0.0.0)

$ git tag -d 1.0.0

Deleted tag '1.0.0' (was 46f092e)

BaiYanzhao@baiyanzhao MINGW64 ~/source/repos/MTx (0.0.0)

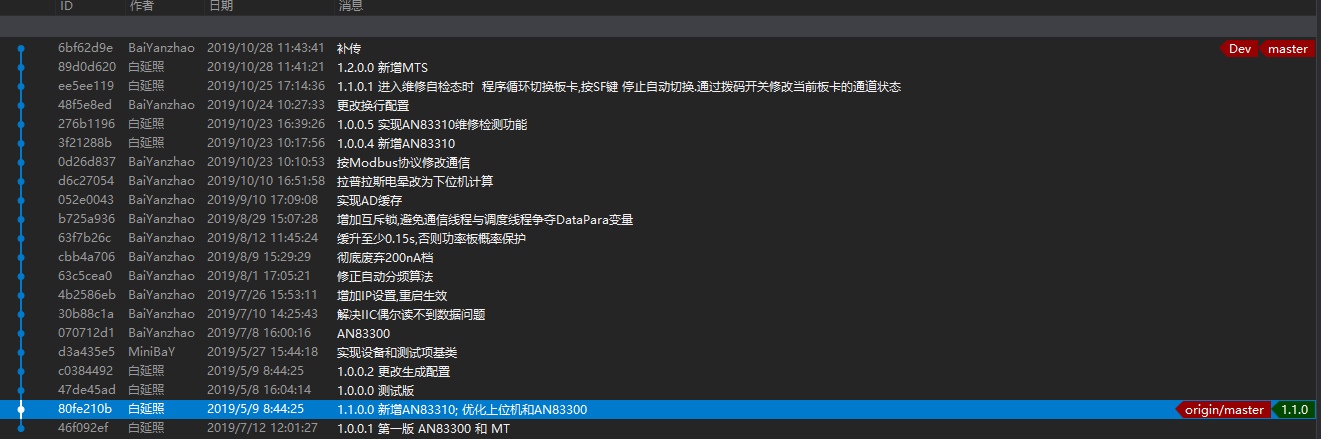
$ git push origin :refs/tags/1.0.0

remote: Updating references: 100% (1/1)

To http://192.168.1.49:2507/r/MT.git

- [deleted] 1.0.0

## 通过变基合并提交



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。

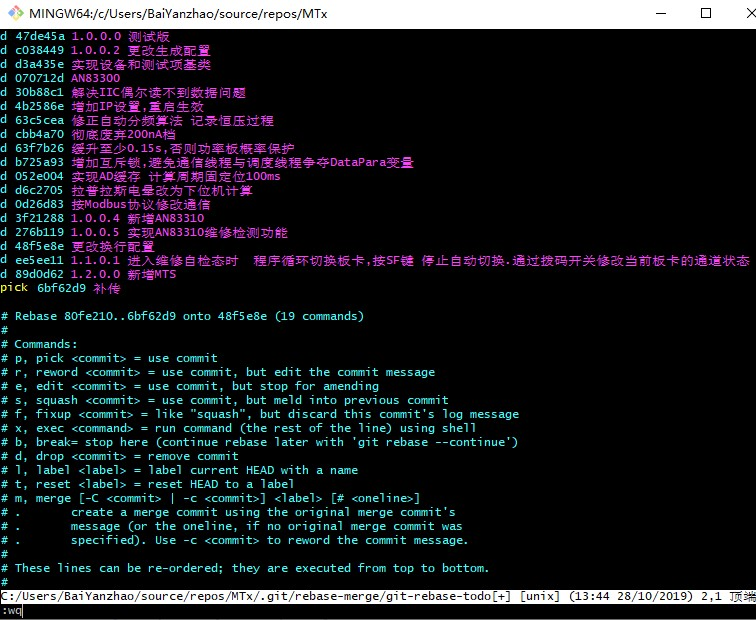
$ cd "C:\Users\BaiYanzhao\source\repos\MTx"

BaiYanzhao@baiyanzhao MINGW64 ~/source/repos/MTx (master)

$ 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”。点键盘“ESC”，然后输入“:wq”回车，完成保存退出。



1. 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。

## 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。