**Training for GitHub**

Revision 49

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Name** | **Date** | **Reason For Changes** |
| Draft |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**NOTICE**

This document contains confidential information, which is proprietary to BSM Wireless Inc. No part of its contents may be used, copied, disclosed or conveyed to any party in any manner whatsoever without prior written permission from BSM Wireless Inc.

[1. Introduction 5](#_Toc439670435)

[2. About GitHub 6](#_Toc439670436)

[3. Getting Started 6](#_Toc439670437)

[4. Branches 6](#_Toc439670438)

[4.2 Feature Branches 7](#_Toc439670439)

[Creating a Feature Branch 8](#_Toc439670440)

[Incorporating a finished feature on develop 8](#_Toc439670441)

[4.2 Production Branch 8](#_Toc439670442)

[4.3 Release Branch 8](#_Toc439670443)

[Creating a release branch 9](#_Toc439670444)

[Finishing a release branch 10](#_Toc439670445)

[4.4 Service Pack or Hotfix Branch 11](#_Toc439670446)

[Creating the hotfix branch 11](#_Toc439670447)

[Finishing a hotfix branch 12](#_Toc439670448)

[4.5 Third Party Collaboration 13](#_Toc439670449)

[5. Code Review 13](#_Toc439670450)

[6. Conflict Resolution on Merge 13](#_Toc439670451)

[Appendix I 14](#_Toc439670452)

[Learning Resources 14](#_Toc439670453)

# Introduction

This document provides guidelines to the BSM Wireless Engineering team on the process to use GitHub as the unified version control system. The team is expected to follow the guidelines for branching, merging, releases, bug fixes and service pack and any other changes that may impact BSM Wireless products. This document provides guidelines on how to use GitHub but is not considered a training document. The team need to utilize alternative training resources to become familiar on how to use GitHub.

# About GitHub

GitHub is a Web-based Git repository hosting service. It offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features. Unlike Git, which is strictly a command-line tool, GitHub provides a Web-based graphical interface and desktop as well as mobile integration. GitHub allows both Public and Private Repositories. At BSM, we mostly work on private repositories. [Reference](https://en.wikipedia.org/wiki/GitHub)

# Assumpation

This document assumes the repositories have been setup in GitHub and the authorized users have access to the repositories they need to access to.

# Branching Policy

See reference doc

# Git Client Installation

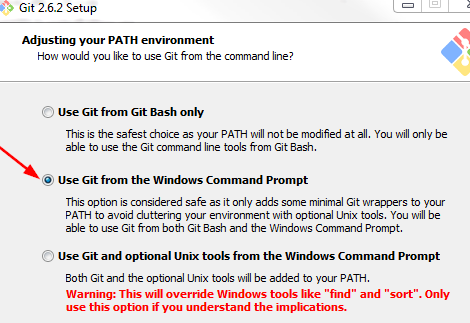
## 5.1 Windows

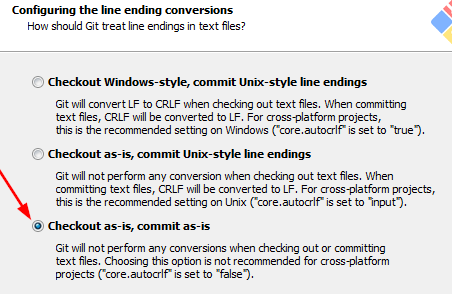
### Git for Windows

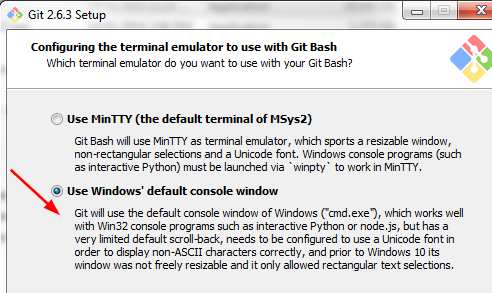
* Download Git for Windows (<https://git-for-windows.github.io/>)
* Install “Git for Windows”

Choose to install as “Administrator” ( so it will be installed under “Program File” folder, otherwise Git will be installed under your account )

Following option must be followed during installation







* Verify the Installation
  + Start “Command Prompt” in Windows
  + Run command “git --version”



### TortoiseGit

* MUST has “Git for Windows” installed
* Installation is pretty straightforward. Just follow the installation steps

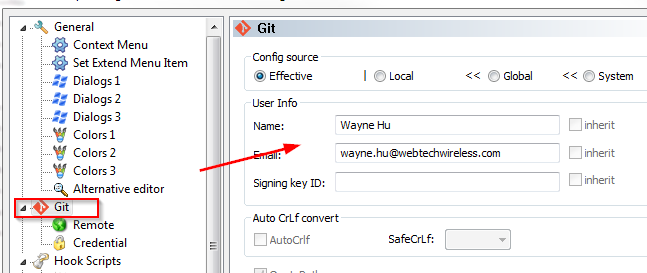
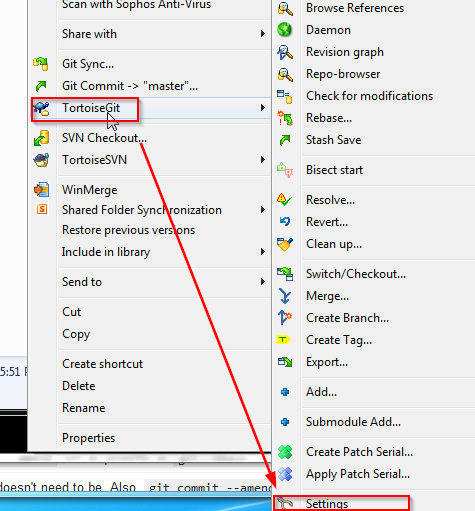
## 5.2 Linux/Mac

All samples will use “git command” and “TortoriseGit” to illustrate.

# Git Client Setup

## 6.1 Assign Name & Email

### TortoiseGit



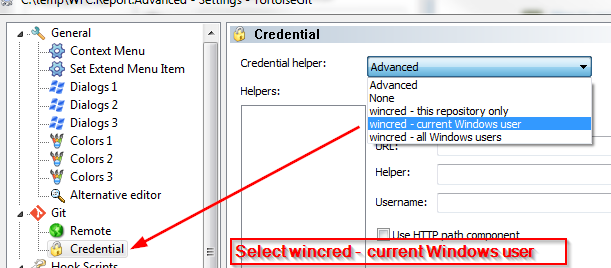
### Command Line

git config –-global user.name “John Test”

git config –-global user.email “john.test@bsmwireless.com”

## 6.2 Remember Password

### TortoiseGit



### Command Line

Below command will store the password for 24 hours ( 86400 seconds )

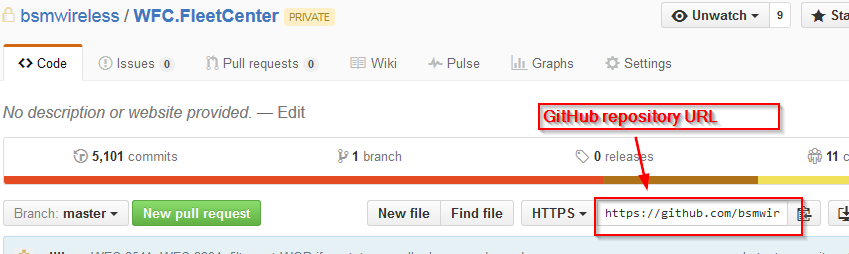
git config --global credential.helper 'cache --timeout=86400'

# General Use Case

## 7.1 Checkout From GitHub

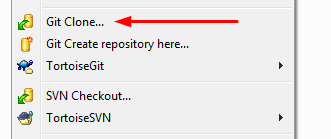
Get repository URL

* login GitHub
* click the repository

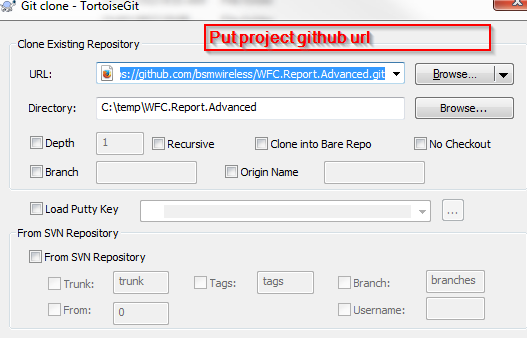


### TortoiseGit

* open context menu of TortoiseGit, select “Git Clone”



* Input the URL and destination directory



### Command Line

cd <your working directory>

git clone [https://github.com/bsmwireless/<RepositoryName>.git](https://github.com/bsmwireless/%3cRepositoryName%3e.git)

Cloning into 'WFC.Report.Advanced'...

Username for 'https://github.com': JohnTest

Password for 'https://johntest@github.com':

remote: Counting objects: 2173, done.

remote: Compressing objects: 100% (577/577), done.

d R0 ei

Receiving objects: 100% (2173/2173), 2.11 MiB | 1.68 MiB/s, done.

Resolving deltas: 100% (1537/1537), done.

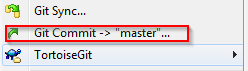
Checking connectivity... done.

## 7.2 Commit Changes

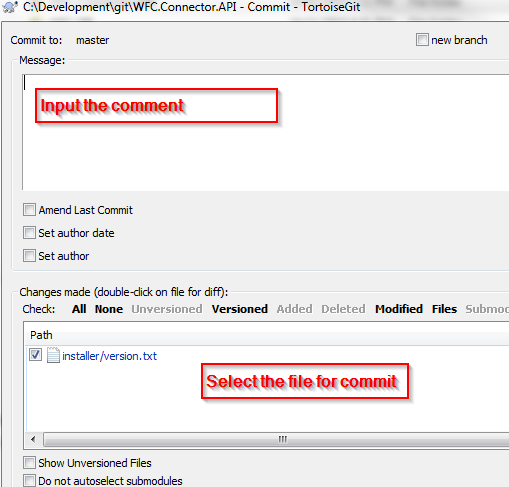
Made changes on tracked files ( the changes are committed to local repository, not GitHub)

### TortoiseGit

* Goto the working directory, select “Git Commit” from context menu



* Select the files that want to commit



### Command Line

* check status first

git status

On branch master

Your branch is up-to-date with 'origin/master'.

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

modified: installer/version.txt

Untracked files:

(use "git add <file>..." to include in what will be committed)

.flattened-pom.xml

RESTApiInstallerLibs/

test/ruby-cucumber/restapi/results.html

no changes added to commit (use "git add" and/or "git commit -a")

* Add modified files to staging area

git add <modified file>

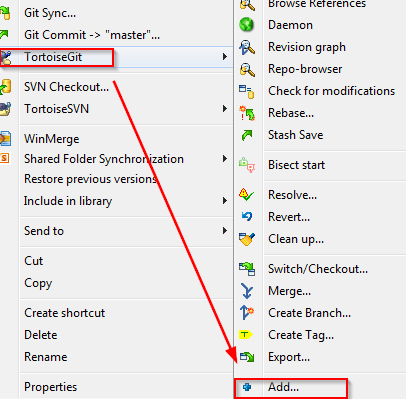
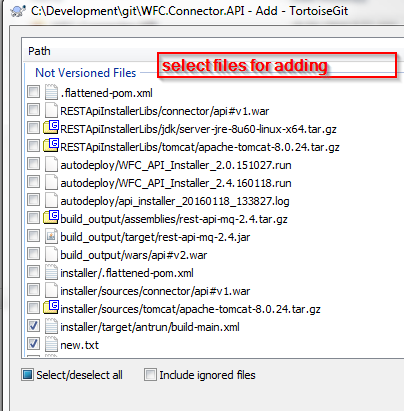
* Commit to local repository

git commit –m “comment for the commit”

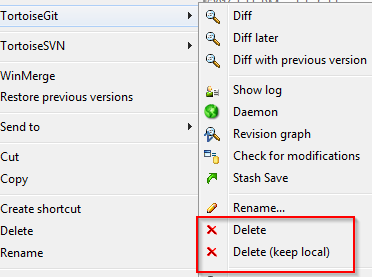
## 7.3 Add/Remove Files

### TortoiseGit

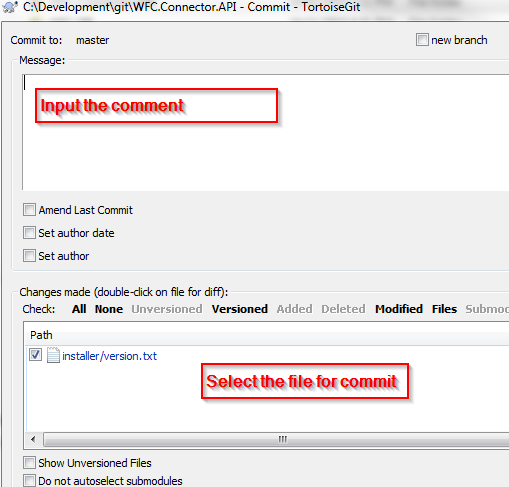
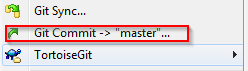
* Add new files

* Remove files



* Commit to local repository



### Command Line

* Add untracked files to staging area

git add <untracked file>

* Remove files

git rm <file name>

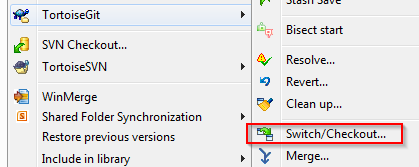
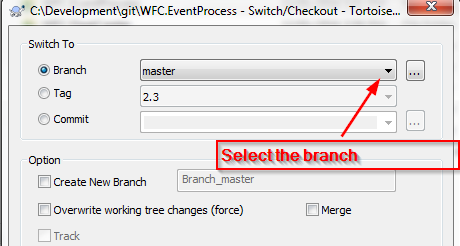
* Commit to local repository

git commit –m “comment for the commit”

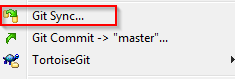
## 5.3 Push Changes to GitHub

### TortoiseGit

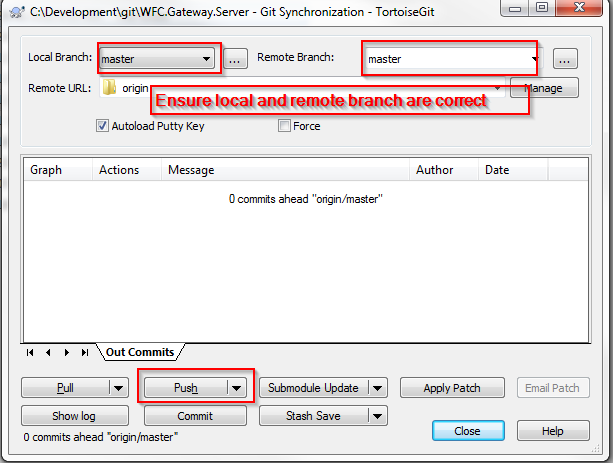
* Switch to branch that need to push to GiHub

* Select “Git Sync”



* Ensure “Local Branch” and “Remote Branch” are correct, then click “Push” button



### Command Line

* Switch to the branch you want to upload to GitHub ( \* attached to current branch )

>git branch

hotfix-2.3.1

release-2.3

\* master

>git checkout hotfix-2.3.1

* Push to GitHub

git push origin <branch\_name>

Example:

git push origin master

git push origin hotfix-2.3.1

## 5.3 Pull Changes from GitHub

## 5.3 View Change History

## 5.3 Working with Branch

create

switch

commit changes

push

merge

## 5.3 Tag

## 5.3 Undo

## 5.3 Resolve Conflicts

# Using Git in VisualStudio

## 4.2 Feature Branches

May branch off from:

develop

Must merge back into:

develop

Branch naming convention:

anything except master, develop, release-\*, or hotfix-\*

use Jira id e.g. SFM-5234

Feature branches are used to develop new features for the upcoming or a future release. The essence of a feature branch is that it exists as long as the feature is in development, but will eventually be merged back into develop (to definitely add the new feature to the upcoming release) or discarded (in case of a disappointing experiment). Feature branch only represents changes to one feature. Do not mix multiple features in one feature branch.

### Creating a Feature Branch

When starting work on a new feature, branch off from the develop branch.

$ git checkout -b myfeature develop

# Using Git in Eclipse

# Using Git in Xamarin

## 10.1 Setup

<place holder>

## 10.2 <Place Holder>

<place holder>

# Appendix I

## Learning Resources

<https://help.github.com/>

<https://www.youtube.com/watch?v=0fKg7e37bQE>

<http://lifehacker.com/5983680/how-the-heck-do-i-use-github>

<https://guides.github.com/activities/hello-world/>

<http://readwrite.com/2013/09/30/understanding-github-a-journey-for-beginners-part-1>

<https://www.atlassian.com/git/tutorials/advanced-overview>

References & Credits

<http://nvie.com/posts/a-successful-git-branching-model/>

<https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>