Building Fabric Engine From Source

Fabric Engine Version 1.2.0-beta

Table of Contents

Chapter 1. Introduction

This document will outline how to set up your development environment in order to build Fabric Engine Core under Mac OS X, Linux, and Windows. The builds under all three systems are generally done from the command prompt, using Mingw32 under Windows for more UNIX-like functionality. The Fabric Engine Core is used by all Fabric applications whether in the browser or in a language binding module.

Fabric supports the following platforms:

- *Windows*: Vista or Windows 7.
- *Mac OS X*: 10.6 (Snow Leopard) or above
- *Linux*: Tested on Ubuntu 10.04 and above (32-bit or 64-bit). Should work with any modern distro.

Chapter 2. Core Setup

2.1. Prerequisites

Before getting started you will need several packages installed in order to be able to retrieve and compile the Fabric source code. Packages vary depending on build operating system.

2.1.1. Windows

2.1.1.1. Install software

The following software is required to build Fabric on Windows:

- Microsoft Visual C++ Express 2010 [http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-cpp-express] (or any superior Visual Studio version). Use standard settings.
- Latest Python 2.x 32-bit [http://www.python.org/download/]. Use standard settings.
- Latest Scons 2.x [http://sourceforge.net/projects/scons/files/scons/]. Use standard settings.

You'll also need to add the python and python/scripts folders to your `PATH` environment variable. To do so click on the Windows Menu, choose “Control Panel”. Now click on “System and Security”, followed by “System”. Finally, click on “Advanced System Settings”, followed by the “Environment Variables” button. If the variable “PATH” already exists, append “;c:\Python27;c:\Python27\Scripts” to its value, else create it and choose “c:\Python27;c:\Python27\Scripts” for its value.

2.1.1.2. Install Git and Configure SSH

The Fabric Engine source code is hosted in a GitHub repositroy, which uses SSH for security. You will need to set up Git and SSH on your windows machine to be able to clone the repository.

Install Git for Windows and (optionally) set up your github ssh key by following the instructions found here: `Set Up Git` [http://help.github.com/win-set-up-git/]

If you want the system to remember your git passphrase, you can follow the instructions found here: `Why git can't remember my passphrase under Windows` [http://stackoverflow.com/questions/370030/why-git-cant-remember-my-passphrase-under-windows]

2.1.2. Mac OS X

2.1.2.1. Install Xcode

Install the latest version of Xcode (4.1 as of this writing). This should automatically get you the `git` binary so you can check out the source.

2.1.2.2. Install MacPorts

Follow the instructions to install MacPorts on your system. [http://www.macports.org/install.php] Use the OS X .pkg installer (the recommended choice). Once the installer has finished, bring MacPorts up-to-date by opening a Terminal and running:

```
sudo port -v selfupdate
```

You need to make sure that the directory `/opt/local/bin` is in your `PATH` environment variable so you can run the MacPorts tools easily. To the end of the file `~/.profile`, add the line:

```
export PATH="$PATH":/opt/local/bin
```

Then restart the Terminal application.

2.1.2.3. Install required applications

There is a version of bison (the GNU yacc-compatible parser generator) that ships with Xcode but it is too old for us; you need to install the version of bison from MacPorts. To do so, run:

```
sudo port install bison
```

The build system we use is scons, and it too needs to be installed from MacPorts. To do so, run:

```
sudo port install scons
```

2.1.2.4. Install Git

Follow the instructions on GitHub for setting up Git on OsX. `mac-set-up-git` [http://help.github.com/mac-set-up-git/]

2.1.3. Linux

A few packages are necessary to build under Linux, namely the compiler, bison, and scons. Assuming you are running Ubuntu, you may run:

```
sudo apt-get install build-essential scons bison
```

If you are running a different distribution use your appropriate package manager to install the required application.

2.1.3.1. Install Git

Follow the instructions on GitHub for setting up Git on Linux. `linux-set-up-git` [http://help.github.com/linux-set-up-git/]

2.2. Build Folder

You first need to choose a directory where the Fabric source code will live. Assuming that `myuser` is your local username, the recommended paths are:

- Windows: `c:\Users\myuser\Fabric`
- Mac OS X: `/Users/myuser/Fabric`
- Linux: `/home/myuser/Fabric`

You will then need to set an environment variable called `FABRIC_CORE_PATH` that points to this newly-created directory. Under Mac OS X and Linux, this can be done by running:

```
export FABRIC_CORE_PATH=~/Fabric
```


You may also want to add this to your `~/.bash_profile` or `~/.profile` file.

Under Windows you'll want to set it as a permanent environment variable by going to Control Panel -> System -> Advanced System Settings -> Advanced -> Environment Variables. You can create a new variable named `FABRIC_CORE_PATH` and set it to `c:\Users\myuser\Fabric` (or whatever directory you chose).

2.3. Clone the PublicDev Repository

If read-only access is enough, you can clone the PublicDev repository from the command line without setting up an ssh key by doing:

```
git clone https://github.com/fabric-engine/PublicDev.git $FABRIC_CORE_PATH
```

Where `$FABRIC_CORE_PATH` is the folder set up above.

If you will require write access then you will need to:

- Ensure that you have uploaded your public SSH key to your github.com account.
- Test that your SSH keys are properly set up for github.com by running: `ssh -T git@github.com`. You should see a message like the following: `Hi myuser! You've successfully authenticated, but GitHub does not provide shell access.`
- If your keys have been set up and you have been granted write access, you can check out the repository by running: `git clone ssh://github.com/fabric-engine/PublicDev.git $FABRIC_CORE_PATH`

2.4. Install Pre-built Third-Party Binaries

Certain additional custom-compiled binaries are needed to compile Fabric. These are platform-specific and available from the PublicDev github page.

1. From the downloads page for the PublicDev repository [https://github.com/fabric-engine/PublicDev/downloads], download the latest set of pre-built third-party binaries for you platform.
2. Unpack the archive into `$FABRIC_CORE_PATH`.

2.5. Build

Before running the build, you'll need to set up some necessary environment variables that the build process will use. These are set by simply running a script found in the PublicDev repository.

Source the `$FABRIC_CORE_PATH`/Util/fabric-build-env.sh script with the following args (Windows: use Git bash):

```
source ~Fabric/Util/fabric-build-env.sh -n Release
```

The `-n` is optional and adds a symlink for the NPAPI plugin to be automatically loaded when your browser runs. You can pass either `Debug` or `Release` to specify either a Debug build or a Release build.

This can be added to your `~/.bash_profile` or `~/.profile` file so that it will always run on the command line if you want.

Important note: If you are building on Windows, you will need to copy two resource headers from your Visual Studio Professional install into Fabric before compiling. In your VS install path, under `VC\atlm-`

`fc\include\` there should be files named `afxres.h` and `winres.h`, these will need to be copied to your `$FABRIC_CORE_PATH\Native\Clients\NPAPI\Windows\` folder.

The build can now be run:

```
cd $FABRIC_CORE_PATH
```

```
scons all
```

Note: Building the documentation guides (`scons guides`) require additional setup steps which have not been detailed yet. Required additional software include `Short-Attention-Span Docbook` [http://napcs.com/products/docbook/], Java, Haml and Ruby.

