Generic SWGEmu Launcher Design

*The design specification and usage document for an implementation of a generic too to install, update and launch a client for a custom SWGEmu based server.*

*Version 0.09.000*

*March 7, 2016*

# Contents

[0 Contents 3](#_Toc447834479)

[1 Purpose 5](#_Toc447834480)

Generic SWGEmu Launcher Design

# Background

The SWGEmu team currently does not provide an open source version of their Installer/Launcher. For servers that do not want to add any in game items, the launcher that the SWGEmu team does provide allows the player to add entries for other servers and launch the game to play on them. The SWGEmu team has implied that their launcher will eventually be made open source by the time that the project reaches its Version 1.00 goal.

The problem that the community of SWGEmu bases servers currently has is that they want to keep the current alpha version of the server and expand the content available on each of the servers. In the commercial version of Star Wars Galaxies, Sony Online Entertainment used a launcher to not only complete the game install, but to also check to see if the game client files were up to date. If the client files were not up to date and in sync with the server the client was updated before it would launch the game. Since the SWGEmu team does not need this update functionality even if the launcher source were made open source today it would be lacking this update functionality. So the community has a need for an open source launcher to support their players and to facilitate sharing between server development communities.

When I was working with the SWGChoice development team I did a very quick and dirty launcher because the team knew that they would be adding items regularly during early development and had experienced the problems that the Bloodfin SWGEmu server had when the game client files were out of date. This launcher did work but it lacked the performance and functionality that the team felt was reasonable in a production version of the launcher.

When the server shut down I had the ownership of the code that I had written for the launcher, considerable experience from creating the launcher and a laundry list of desired changes to improve the performance and functionality of the launcher. I eventually planned to rewrite the launcher code to be easy to adapt for use by other servers and post it to the [Mod The Galaxy](http://www.modthegalaxy.com/) website. This was delayed due to some real life issues but I was eventually able to return to the project. To facilitate the work I started with a design document and as work was being done on the rewrite, mostly from scratch, to add to the design document a user manual section.

# Development Environment

## Development Environment Cost

To make it easier to develop server code the SWGEmu team chose a philosophy of creating a development environment that could be very powerful while costing nothing. This philosophy should be retained as part of the scope for the Launcher development environment. The goal will be to try to find tools that can be legally obtained and used at no or low cost.

## Target Environment

While it is possible to run the SWG Client on Linux and Macintosh systems, the distributed binaries are designed to run in a Microsoft Windows environment. Most of the people wanting to play on a SWGEmu server will be trying to do so using a Microsoft Window System. So producing binaries for other operating systems or using other operating systems to develop the launcher will not be a part of the consideration.

While Microsoft is no longer supporting the Microsoft Windows XP operating system, there are plenty of SWG gamers that are still using systems with that operating system to play the game. So the development environment must be able to produce binaries that will work on that version of the Microsoft operating system. The primary concern will be if the compiler produced code that needs libraries that are not part of the default operating system. If the compiler required that the target systems install redistributable libraries, then they must still supply these files for the Microsoft Windows XP operating system.

Ideally the development tools should be part of an integrated development environment with a robust debugging facility.

## Language Selection

The SWGEmu server code is written primarily in C++ and Lua with some scripting tools. So it would have been logical to use C++ to write the launcher in. This is the language that it appears that the SWGEmu team used to create their launcher. One thing that I relearned with the SWGChoice development team was that there are different levels of experience and skill when it comes to enthusiasts that want to work on a SWGEmu based server. For this reason C++ was rejected. While there is lots of training materials out there for this language, those training materials target people with a good general programming foundation. This opens up considering other languages.

Java is a very popular language that like C++ has all of the tools to create a launcher. It also has the ability to produce operating system independent executables. This feature would allow the future modification of the launcher to run native on Linux and the Macintosh operating system and then to invoke an emulator to run the SOE client files under. The primary provider of the Java language also provides a very good integrated development environment with debug and unit test functionality built in at no cost. This is very enticing but like C++ all of the documentation and the language itself targets people who have a good general programming foundation.

Basic was created as a language that would be a good introduction to programming language. Over time companies like Microsoft enhanced the language so that it could be used for some very powerful application development while it retained the ease of learning. Microsoft through Visual Studio 2015 has a very powerful integrated development environment for their Visual Basic. Their .NET framework also provides tools to create and use a very clean and useful GUI based installer. It also includes tools for getting files over the internet. Currently Microsoft makes available Visual Studio 2015 – Community edition at no cost. It does not have built into it tools to facilitate co-operative development on a large scale or an application installer tool like the commercial versions. Lacking these components does not make the free version of Visual Studio 2015 unsuitable for the project.

Upon further investigation the latest .NET framework, version 4.6.1, includes a redistributable install package for Windows XP even though it was release nearly a year after support for Windows XP officially ended. Also starting with Visual Studio 2015 the [GitHub](https://visualstudio.github.com/) web site now provides an extension to Visual Studio that allows a seamless integration with your GitHub account and repositories.

In the end Visual Basic and .NET 4.6.1 was selected for the language and Visual Studio 2015 – Community edition for the development environment was chosen. Using C++ in Visual Studio was an option but the higher skill set to come up to speed with it was taken as a detriment. Also the code base for the launcher is small enough that if anyone wanted to port it to Visual C++ it should not be a difficult task.

## Installer Selection

The number of files for the installer are reasonably small. It would have been possible to simply distribute the files associated with the installer in a zip file and require people to run a .bat file to do the initial setup of the registry entries.

The equivalent 2013 version of Visual Studio 2015 – Community edition did include a free InstallshieldLE 2015 version for integration into Visual Studio. Flexera decided that they would no longer support InstallshieldLE integration into Visual Studio 2015 – Community. Developers had the option to upgrade to a paid version of Visual Studio 2015 which Flexera does support with InstallshieldLE, purchase a full version of Installshield or migrate to another install package.

Given the goal of no or low cost for the environment this automatically rules out Installshield as an option. A simple search of the web produces some alternatives.

The first is that Microsoft included installer code in Visual Studio through Visual Studio 2010. In Visual Studio they switched to an integrated version of InstallshieldLE and no longer shipped their integrated installer. The do still make it available [here](https://visualstudiogallery.msdn.microsoft.com/f1cc3f3e-c300-40a7-8797-c509fb8933b9) as a free plug in. It is not being updated with newer install technologies so while it is an option it is low on the list.

The InstallAware product is another installer that integrates with Visual Studio. A quick check of their web site shows that they have a free version that integrates into Visual Studio and supports the newer installer abilities. A problem quickly arises on their download page. They ask for a valid email address to send the key to unlock the installer package. However, they reject my email address since it is @gmail.com. I sent them a message asking if there was any way around this. So while others may be able to use this it looks like it will not be an option for me to develop with.

**Update: The InstallAware sales team did respond to my message with the password needed to run the installer for their product. While this does help their cause since I can now evaluate their product the final decision will be made based on ease of use. The top two candidates so have the needed functionality so ease of use will most likely break the tie.**

This exhausts the installers that can integrate with Visual Studio. The next step is to look at good standalone products that have a free option.

The first option that popped up was [Advanced Installer](http://www.advancedinstaller.com/). It does not have a truly free version. If you are a certified Microsoft Technician, regularly review and blog about software or are an open source project they can give you a license for free. I guess that they see these people as advertising their product and are willing to give these people free licenses. While this is an open source project and I may be able to get a free license for my use in the project, I doubt that they would give a free license to every SWGEmu based server a free license. So this probably rules out this product.

Another product that jumped out is [Inno Setup from jrsoftware](http://www.jrsoftware.org/isinfo.php). This product is capable of doing everything that would be needed by the installer package and is free. This is what was used for the SWGChoice launcher install. It supports setting up registry keys, software dependencies, digital signing and of course install/uninstall. So this is the installer that will be used for this project.

## Update Mechanism

Updating the client over the internet is a fundamental premise of the project. Transferring files over the internet can be done several ways and using different protocols.