**EVE Influence Map V6.0  
(Codename: *Landlord*)**

**System Requirements:**

A fast processor and 1GB+ of RAM

**Software Requirements:**

Java SDK & JRE

MySQL Server

MySQL Workbench

MySQL Connector for Java

Eclipse IDE for Java

C++ Compiler of choice

cURL

PNGcrush

advPNG

**Preface**:

This will be a set of instructions on how to install the software and build the EVE Influence Map. An understanding of software engineering is required, knowledge of Java, C++, and MySQL is highly recommended.

1. **MySQL**
   1. The first piece of software required is MySQL, which is used as the data backend to store the data required to build the map and is from which Java pulls the data it needs.
   2. The tables are based on the MySQL conversions of CCP’s Static Data Dump, with additional columns added as necessary
   3. The default install options for MySQL (installing it configured for a developer machine) is ideal. Assuming you’re on Windows, whether you want to install this as a service or not is up to you. However, it’s recommended if you’ll be generating daily maps.
   4. Once MySQL is installed, go in to the console and we’ll need to create the EVE database, this is done with the following command:  
      *create database eve;*
   5. Once that is created, go in to the EVE database via the *use eve;* command to make sure everything has worked up to this point.
   6. Use the MySQL Workbench to log in to your database. Once logged in, open the *buildTables.sql* script and execute it to build all of the tables the EVE Influence Map uses. The table will undoubtedly be out of date, but we’ll take care of that later. For now, we’re done with MySQL
2. **Java/Eclipse**
   1. Install both as you’d normally do so
   2. Unpack EVEMap.zip, this is the source code. It’s exported from an Eclipse project, so import it back in to Eclipse as an existing project and you should be set.
   3. Install the MySQL connector for Java, please see the instructions that come with it for how to install it.
3. **Test Run**
   1. At this point you are able to build a copy of the EVE Influence Map based upon the old data we imported in to the table. Please try to build the map now to make sure that everything up to this point has worked correctly. You’ll need to execute the Java VM with following switches, please see autobot.bat for an example
      1. –Xss64m switch, otherwise the Goons will probably blow the stack.
      2. -Xmx256m switch, otherwise Fraternity will probably blow the heap.
      3. –user <username>, the user name for your MySQL account. Usually root.
      4. -password <password>, the password for the above account
   2. The EVE Influence Map program requires no intervention once it has started, it will run and exit on its own. When it completes it will leave the *influence.png* file at the location the program was run from, this is the final map.
4. **Build the Parsing Tools**
   1. 5 simple C++ tools are included in the package for manipulating and preparing data:
      1. **Sovereignty Sanitizer ESI2**: Takes the raw CCP ESI structures.json data and writes two new files from it: a SQL statement for updating the sovereignty DB, and another well parsed file for use in comparing sov data to figure out where sov changed.  
         Inputs: *structures.json*  
         Outputs: *updateSov.sql, sov.txt*
      2. **Alliance Fetch Builder ESI**: Takes the raw CCP allianceids list and generates further ESI calls to collect the name for each alliance, since those require separate calls. You will need to call *fetchAlliances.bat* after this tool completes  
         Inputs: *allianceids.json*  
         Outputs: *fetchAlliances.bat*
      3. **Fetch Alliances.bat**:Not a program, but rather a generated batch of ESI calls. Looks up the alliance name for each alliance.  
         Inputs: none  
         Outputs: *alliancenames#.json*
      4. **Alliance Sanitizer ESI**: Takes the output of *fetchAlliances.bat* and generates a final SQL script to add each alliance to the DB and update their name with the current name.  
         Inputs: *alliancenames#.json*  
         Outputs: *updateAl.sql*
      5. **Sov Change Finder ESI2**: Takes the refined structure info from today and yesterday and builds a SQL script file for updating the sovchangelog table with any changes in sovereignty (for drawing the little circles and making the on-map list).  
         Inputs: *sov.txt*, *oldsov.txt*(yesterday’s sov.txt renamed)  
         Outputs: *updateSovChange.sql*
      6. **Autobot Builder ESI2**: Tool to build a Windows batch file to fully build the map. It’s based on how my Windows server is setup, so you’ll probably need to rewrite most of it.  
         Inputs: none  
         Outputs: *autobot.bat*
   2. Once the tools are compiled, feel free to move them to whatever directory you will be using to build the map. Presumably you’re going to export a JAR file of the map generator to this directory.
   3. You’ll also need to get copies of cURL, advPNG, and PNGcrush, and put them in this directory.
5. **Anatomy of A Daily Map Build**
   1. Backup the DB and save it somewhere safe
   2. Build and run the autobot batch file, all that follows is from that
   3. Delete all of the old, outdated files
   4. Copy over yesterday’s sov.txt as oldsov.txt for use with today’s map run
   5. Use cURL to get today’s ESI data from the EVE ESI server
   6. Sanitize & parse the sovereignty structure and alliance data.
   7. Copy over generated SQL scripts to SQLoutput folder
   8. Have the MySQL server execute the new scripts
   9. Run the map generator
   10. Do a better job than Java of packing the map as a PNG using PNGcrush and advPNG (see autobot code for parameters). pngCRUSH will remove the unneeded alpha channel, and advPNG will get us the best compression possible (it’s a bit of a hog to run, but it’s one optimization for a file downloaded thousands of times per day)
   11. Upload the map to verite.space using cURL
   12. Also copy over the resulting map to the Oldmaps folder for archive
6. **Doing A Full Test Run**
   1. Note: You won’t have any data for comparison purposes to find sovereignty changes until day 2. For now, stop the autobot script after it sanitizes the sovereignty data for the day and copy today’s data over to oldsov.txt, so that the rest of the tools will execute correctly. Then continue the script from where you left off.
   2. Attempt to run a generated autobot script. If you have done everything correctly here you should have a completed map and have it uploaded in as little as 90 seconds (or your money back!)
   3. Once you can confirm that the script runs correctly, you can schedule it to run every day or modify it as required.

**Various Notes**

1. **Alliance Colors**
   1. Alliance colors are dictated by the *color* field in *evealliances*. They are in a RGB hexadecimal form. Colors need to be intense here to carry after factoring in alpha transparency, and the influence map application may change colors if they don’t meet certain guidelines such as intensity (I’m not completely sure what the logic is behind it)
   2. To change an alliance’s color, figure out what new color you want (in hex) and then do:*update evealliances set color = “[newcolor]” where name = “[alliance name]”;*
2. **Autobot**
   1. Autobot is a simple/stupid batch file. If CCP has somehow screwed things up, it will probably fail and bad things will happen. (But CCP has been pretty good about things the last few years)
   2. It’s advisable to not generate the map until a few hours after DT, so that if it runs long you don’t try to query the ESI server while it’s unavailable.
3. **EVE Influence Map Application**
   1. The algorithm that places alliance names is highly recursive, the really big alliances (BoB, RA, GoonSwarm) can potentially cause the application to run out of stack **and** heap space (yeah, it’s that bad). If this occurs, run the JAR file with the –Xss64m and -Xmx264m options.
   2. No, I don’t know how everything works either. Paladin Vent wrote the original logic; I’m just standing on the shoulder of giants (and better coders than I)
4. **EVE Universe**
   1. The EVE universe is technically upside-down; so Branch would be drawn down below and vice-versa.
   2. Rather than handling this quirk at every last step of the code, I just flip the Z data in the *mapsolarsystems* and *mapregions* tables so that everything is right-side up.
   3. So if you need to import *mapsolarsystems* fresh from the CCP static data export, you’ll need to flip it.
      1. And if you do that, you’ll want to re-add the missing custom columns.

ALTER TABLE `eve`.`mapsolarsystems`

ADD COLUMN `allianceID` INT(11) NULL DEFAULT NULL AFTER `securityClass`,

ADD COLUMN `sovereigntyLevel` INT(11) NULL DEFAULT NULL AFTER `allianceID`,

ADD COLUMN `sovereigntyDateTime` DATETIME NULL DEFAULT NULL AFTER `sovereigntyLevel`,

ADD COLUMN `stantion` INT(11) NULL DEFAULT 0 AFTER `sovereigntyDateTime`,

ADD COLUMN `constellationSov` INT(11) NULL DEFAULT 0 AFTER `stantion`,

ADD COLUMN `gridRef` CHAR(10) NOT NULL DEFAULT 'A0' AFTER `constellationSov`,

ADD COLUMN `ADM` DECIMAL(2,1) NOT NULL DEFAULT 1.0 AFTER `gridRef`;

* + 1. Most are self-explanatory. A few like *stantion*, *constellationSov*, and *gridRef* are no longer used, but removing them would screw up the parameter count in the Java code, and I am lazy

1. **NPC Alliances**
   1. NPC influence is held in the *npcalliances* table on a per system basis. This allows you to alter NPC influence as necessary
   2. The default is 10, but some systems go as high as 130 (Blood Raider) in order to balance out the immense scores of player influence
2. **Influence Propagation**
   1. Influence is emanated from solar systems and spread both via gates and via space (pixels). Gate influence can travel a few jumps, and one further jump if it emanates from Activity Defense Multiplier 6.0 (max-ADM) systems
   2. You will get weird edge cases now and then where space far from any system will change hands. Math is weird like that
   3. Sov ownership is sorted out by the pre-processing tools. Specifically, sov is determined by looking who owns the Infrastructure Hub (IHub) in a given system. Reinforced structures do not report their ADM, so we look at both the IHub and TCU to see if we can derive the official ADM value from either of them.
3. **Triglavian Font**
   1. The Triglavian font is included for fun, in case you find the need to have the map labeled in the Triglavian alphabet
4. **Performance**
   1. The map code is multi-threaded, and by default will use all the threads available on a system. Though you do get diminishing returns with additional threads (I assume DB locks become an issue at some point)
   2. The map also eats a fair bit of memory (upwards of a GB). This was far more of a problem in 2007 than it is in 2003. But may it be explicitly known that this map has not been extensively optimized beyond refactoring the original code and adding multi-threading
5. **Other**
   1. The source code to the map is offered on what’s essentially a BSD/MIT basis, to ensure its continuation and to share in knowledge. That said, I would greatly prefer it not being used to publish a “competing” map, so that there remains a singular, canonical influence map going back to 2007.
   2. If I drop off the face of the earth, all the historical maps are on verite.space and are easily copied. If I don’t drop off the face of the earth, then I’ll find someone to take over the project should I decide to retire.
      1. If *EVE Online* drops off the face of the earth, then we’ve all truly and finally won EVE…
   3. Why is the map generator in Java but all the processing tools in C++?
      1. When Paladin Vent retired, he only left behind the Java generator, and not his processing tools. So those were all (re)created by me, and I prefer C++ and the Unix philosophy of chaining together simpler tools (it makes debugging so much easier).
   4. The full copyright notice for the Java generator program is below

/\*

\* Copyright (c) 2007, [AEGL, UNL] Paladin Vent

\* All rights reserved.

\*

\* Additional authors:

\* [CE] Mirida (Multithreading)

\* [FREEE] Verite Rendition (Various, General Upkeep: Aug 2007 to Present)

\* [MOP] Calistra "Draekas" Darkwater (Multiple-Instance Name placement)

\*

\* Redistribution and use in source and binary forms, with or without

\* modification, are permitted provided that the following conditions are met:

\* \* Redistributions of source code must retain the above copyright

\* notice, this list of conditions and the following disclaimer.

\* \* Redistributions in binary form must reproduce the above copyright

\* notice, this list of conditions and the following disclaimer in the

\* documentation and/or other materials provided with the distribution.

\* \* Neither the name of the United Legion nor the

\* names of its contributors may be used to endorse or promote products

\* derived from this software without specific prior written permission.

\*

\* THIS SOFTWARE IS PROVIDED BY [AEGL, UNL] Paladin Vent ``AS IS'' AND ANY

\* EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED

\* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE

\* DISCLAIMED. IN NO EVENT SHALL [AEGL, UNL] Paladin Vent BE LIABLE FOR ANY

\* DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES

\* (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;

\* LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND

\* ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT

\* (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS

\* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

\*/