**Multiplayer Assembly Package  
Tutorials  
Stats and Stats Recording**

This is more of a class documentation for the new stats container class included in MAP 1.4, To install it, you will find a stats system folder in the tutorial files. Place both the C++ and C++ Header file in the PGD/Solutions/ folder, then follow the instructions of the Adding a New MAP Class tutorial to add this new class to the engine.

This stats container uses a linked list to store stats inside the engine. The stat field holds a name and a value, the name is respectively, what you want to name your stat and the value is the number held within the class, which could be something like “kills”, “deaths”, ect.

The information is saved in an XML file which is then encrypted using AES-256 CBC. \

The following is all of the TS definitions created by this class:

*addStat(string n) – Adds a stat with name “n” to the stat list  
getStatID(string n) – Returns the ID of a stat by its name  
getStatInfo(int id) – Returns a tabulated string containing the name and value of a stat by the ID value.  
getStatValue(int id) – Returns just the stored value of a stat by ID.  
updateStat(int id, F32 value) – Add or subtract a value from a stat by ID.  
saveClientData(string guid, string path) – Save all stats to a path. GUID info of client is saved.  
loadClientData(string path) – Load a saved encrypted XML file into the stats system.*

To use this system, you need to create a client-side file that is loaded before the client’s info is. Basically create and execute a file somewhere in scripts/client/ that contains a list of all of the stats in your game, YOU CAN NOT CHANGE THE ORDER OF THESE ONCE THE FIRST FILE IS SAVED, OTHERWISE YOU WILL CRASH YOUR GAME. Here is an example of a file:

//stats file  
addStat(“kills”); //ID: 0  
addStat(“deaths”); //ID: 1  
addStat(“exp”); //ID: 2

Then once this is done, update your login script (guiWindows.cs) to make the following calls:

%guid = $ConnStore::guid;

%file = GetUserDataPath() @"save/"@%guid@"/data.xmle";

if(!isFile(%file)) {

echo("\* Creating Client Save File Now.");

SaveClientData(%guid, %file);

}

else {

echo("\* Loading Client Save Data.");

LoadClientData(%file);

}

This will ensure that the client’s file is created or loaded once they log-in to your game. From here out, the system is relatively straight-forward. To update a stat’s value, you only need to get the ID which can either be done by storing all of the values (see the comments in my addStat() calls) in the file, or by using getStatID() of the stored name. From there, simply call updateStat(id, amount); to update it.

As for saving, I don’t recommend constantly saving the file every time the stat is updated, as that may cause some lag issues. Instead, do it every ‘x’ number of calls or even better, update function disconnect() in client.cs and call the save function when the player disconnects from a server.

For GUI screens that need the value of the stat, you can either use getStatInfo, or getStatValue, both of which use the stat ID. For statInfo, you will need to make use of getField (see the source code for more info), whereas statValue will simply push the number.

For further explanation, ask on the forums, or see the PGD Survival source code (which uses a slightly modified version of this system (survivalControl.cpp))