**Multiplayer Assembly Package  
Tutorials  
Creating a Server-Side Rank/Progression System**

This tutorial will teach you how to create a server-side Rank/Progression system using the Multiplayer Assembly Package.

**What is it?**A Server-Side Rank-Progression system is a widely used system in games that involve leaderboards, competitive multiplayer and other silly things like reaching prestige level… never-mind, you got my reference. ServerSide RP systems can be useful, but at the same time they have their own set of problems compared to the Client-Side model. A major one deals with large user-bases and server load. For the indie developer, having your web host scream at you for large bandwidth overloads is probably not the best of ideas, but at least your user data is safe on your server… or is it?

**Tutorial**Start by following the installation tutorial of the client side RP container to create the C++ class which will store our data, the only true key difference between these systems is what you ACTUALLY do with the data once you have it.

Fire up ye olde php editor (a text editor will do fine) and add something along the lines of this to it:

<?php

require("path/to/sql.php"); //modify this to route to sql.php

$guid = $\_POST["guid"];

$cert = $\_POST["cert"];

$data = $\_POST["data"];

//change this line... yada yada yada

$sql = sqlLogin("host\_authUsr", "sqlPassowrd", "authSystem");

//check the GUID & signature to see if they are who they say they are

$guid = $sql->real\_escape\_string($guid);

$cert = $sql->real\_escape\_string($cert);

$data = $sql->real\_escape\_string($data);

$stmt = mysqli\_prepare($sql, ‘SELECT \* FROM Accounts WHERE guid=? AND signature=?’);

mysqli\_stmt\_bind\_param($stmt, ‘ss’, $guid, $cert);

$stmt->execute();

$result = $stmt->get\_result();

if($result->fetch\_array(MYSQLI\_ASSOC) == NULL) {

//naughty naughty hacker...

die("$MAP"."$INVALID\_ACCOUNT");

}

$sql->close();

$sql = sqlLogin("host\_othrUsr", "otherSQLPassword", "rankSystem");

//DATA == GET - Send Down Rank, Else: Save Info

if(strcmp($data, "GET")) {

$stmt2 = mysqli\_prepare($sql, ‘SELECT \* FROM Accounts WHERE guid=?’);

mysqli\_stmt\_bind\_param($stmt2, ‘s’, $guid);

$stmt2->execute();

$result = $stmt2->get\_result();

if($result->fetch\_array(MYSQLI\_ASSOC) == NULL) {

die("$MAP"."$NO\_DATA\_ON\_SERVER");

}

else {

$row = $result->fetch\_array(MYSQLI\_ASSOC);

die("$MAP"."$DOWNLOAD ".$row[rankdata]);

}

}

else {

$stmt2 = mysqli\_prepare($sql, ‘SELECT \* FROM Accounts WHERE guid=?’);

mysqli\_stmt\_bind\_param($stmt2, ‘s’, $guid);

$stmt2->execute();

$result = $stmt2->get\_result();

if($result->fetch\_array(MYSQLI\_ASSOC) == NULL) {

$stmt3 = mysqli\_prepare($sql, ‘INSERT INTO RankTable (guid, rankdata) VALUES (‘?’, ‘?’)’);

mysqli\_stmt\_bind\_param($stmt3, ‘ss’, $guid, $data);

$stmt3->execute();

die("$MAP"."$UPLOAD\_SUCCESS");

}

else {

$stmt3 = mysqli\_prepare($sql, ‘UPDATE RankTable SET rankdata=? WHERE guid=?’);

mysqli\_stmt\_bind\_param($stmt3, ‘ss’, $data, $guid);

$stmt3->execute();

die("$MAP"."$UPLOAD\_SUCCESS");

}

}

?>

Save this somewhere on your server under the name rankUpload.php. You will notice if you read the script that we will not only be using the rank system SQL system, but we will need a new MySQL database and table. So, follow the setup route shown in the setup guide to create a new database, user with full access, and table with the following info:

Table Name: RankTable

Value Set:

guid – int(11)

rankdata – text()

Modify the php code above to reflect these changes (the SQL login stuff).

This will store your player’s rank information. The information will be an AES encrypted file with the player’s guid as the file key. Also, to establish the connection, the player needs to send to the server their GUID, signature, and a message with either the information to save, or GET to download their rank file. So, let’s write some new TS code to handle these transmissions.

//rankTransfer.cs

$Rank::Host = "www.somewebsite.com";

$Rank::Path = "/path/to/rankUpload.php";

function downloadRank() {

$Rank::Buffer = "";

if(isObject(TCPRankDownload)) {

TCPRankDownload.delete();

}

%t = new TCPObject(TCPRankDownload);

%separator = getRandomSeperator(16);

%header = assembleHTTP1\_1Header($Rank::Host, $Rank::Path, "POST", "PGD XXZ568 Client", "Content-Type: multipart/form-data; boundary="@%separator@"\r\n");

//GET request to server

%dispo = makeDisposition(%separator, guid, $ConnStore::guid);

%dispo = %dispo @ makeDisposition(%separator, cert, $ConnStore::sig);

%dispo = %dispo @ makeDisposition(%separator, data, "GET", 1);

%header = %header @ "Content-Length: " @ strLen(%dispo) @ "\r\n\r\n";

%request = %header @ %dispo;

//Send.

%t.request = %request;

%t.connect(%host @":80");

}

function uploadRank() {

$Rank::Buffer = "";

%guid = $ConnStore::guid;

%file = GetUserDataPath() @"save/"@%guid@"/data.xmle"; //where you save your rank files at

AESEncFile(%file, %guid, "save.xmle");

%fo = new FileObject();

%fo.openForRead("save.xmle");

while(!%fo.isEof()) {

%fCont = %fCont @ %fo.readLine();

}

%fo.close();

%fo.delete();

deleteFile("save.xmle");

if(isObject(TCPRankDownload)) {

TCPRankDownload.delete();

}

%t = new TCPObject(TCPRankDownload);

%separator = getRandomSeperator(16);

%header = assembleHTTP1\_1Header($Rank::Host, $Rank::Path, "POST", "PGD XXZ568 Client", "Content-Type: multipart/form-data; boundary="@%separator@"\r\n");

//GET request to server

%dispo = makeDisposition(%separator, guid, %guid);

%dispo = %dispo @ makeDisposition(%separator, cert, $ConnStore::sig);

%dispo = %dispo @ makeDisposition(%separator, data, %fCont, 1);

%header = %header @ "Content-Length: " @ strLen(%dispo) @ "\r\n\r\n";

%request = %header @ %dispo;

//Send.

%t.request = %request;

%t.connect(%host @":80");

}

function TCPRankDownload::onConnected(%this) {

//delay it shortly so C++ can finish up the needed methods.

%this.schedule(500, send, %this.request);

}

function TCPRankDownload::onLine(%this, %line) {

if(trim(%line) !$= "") {

%this.canRead = true;

}

if(%this.canRead) {

if(strstr(%line, "$") != -1) {

$Rank::Buffer = $Rank::Buffer @ "\n" @ %line;

}

}

}

function TCPRankDownload::onDisconnect(%this) {

%this.delete();

if(firstWord($Rank::Buffer) $= "$MAP$INTERNAL\_ERROR") {

echo("Rank Download: The server encountered an internal error.");

}

else if(firstWord($Rank::Buffer) $= "$MAP$INVALID\_ACCOUNT") {

echo("Rank Download: Invalid account certificate sent.");

}

else if(firstWord($Rank::Buffer) $= "$MAP$NO\_DATA\_ON\_SERVER") {

echo("Rank Download: No Rank Data.");

}

else if(firstWord($Rank::Buffer) $= "$MAP$UPLOAD\_FAIL") {

echo("Rank Download: Rank Upload Failed, contact server admin.");

}

else if(firstWord($Rank::Buffer) $= "$MAP$UPLOAD\_SUCCESS") {

echo("Rank Download: Rank Upload Success.");

}

else if(firstWord($Rank::Buffer) $= "$MAP$DOWNLOAD") {

//save the new file.

%data = strReplace($Rank::Buffer, "$MAP$DOWNLOAD ", "");

%guid = $ConnStore::guid;

%file = GetUserDataPath() @"save/"@%guid@"/data.xmle"; //where you save your rank files at

%fo = new FileObject();

%fo.openForWrite("save.xmle");

%fo.writeLine(%data);

%fo.close();

%fo.delete();

AESDecFile("save.xmle", %guid, %file);

deleteFile("save.xmle");

echo("Rank Download: Rank Download Success.");

}

}

Now, when your client logs in, call downloadRank() to connect to the server. Perform a file exists check after the call, and if the download succeeds, proceed with the CS Rank system calls. Then, when you want to upload, first call the CS Rank System save file call, and follow it immediately with the uploadRank(); command.

NOTE! The transmission maybe too large such that the TCPObject crashes upon upload. If this is the case, then you will want to replace the transaction with a cURL based one. See cryptoPackage.cpp or the tutorial on web access using cURL to see how this is handled.

And that about covers it. You should now be able to store your player’s rank data on the server. Just ensure you follow proper security measures when using the same MySQL info for other tasks, the worst thing you could do is write an unsecured PHP file that gives unauthorized access to the data table, hence the “or is it” mentioned earlier. Just keep your scripts safe and secure, and you should have no problems.