# Multiplayer Assembly Package
# Startup Guide and Pack Installation

**By: Robert C. Fritzen**
**Phantom Games Development**

**Built to run on Torque 3D MIT 3.5**

First off, I would like to thank you for purchasing my first kit for Torque 3D. I'm glad to have the wonderful community at GarageGames behind me when moving my own projects forward and after coming across many problems and successes, I have decided to share one of my greatest successes with you, the community.

Enter the Multiplayer Assembly Package (MAP). As you are likely aware (because you bought this pack), this package will give your game an advanced level of cryptography to protect your user accounts with, and it will also allow you to encrypt client save data to protect it against evil hackers. I have also incorporated the cURL library into Torque 3D to address an issue with the TCPObject inside the engine when creating or recovering user account certificates. You and your users will appreciate the fact that you are no longer crashing upon account registration ;)

This document file is your complete guide to installing this package into Torque 3D. I highly recommend you read through everything carefully. If you have any questions or need help setting this up, hop on the Phantom Games Development forums (http://forums.phantomdev.net) and post a message in the MAP section. If you do not have access to this board, contact me at rfritzen@phantomdev.net and I will sort this issue out. If this is your first time reading this document, don't just hop on the forums and expect to have access, we will get there soon enough.

**If you experience and issues or problems with the pack that seem to be related to the pack's code please report those to me immediately so I can address the problem. Being a paid product, I want to ensure it works as intended and that the needed support for it is present.**

# I. Needed Stuff, Pack Stuff

The following are the needed prerequisites for using this pack.

1) Torque 3D MIT v3.5 or Higher (Built without errors)
2) A Web Enabled server with PHP 5 and MySQL
3) Basic PHP Editing Knowledge (Text Editing, Mainly)
4) Basic MySQL Knowledge (Creating Databases and Tables)
5) Microsoft Visual Studio 2008 (Working on 2010, some other work may be needed)
6) Time

**Use Allowance:**

*This pack has been downloaded for use for your projects and your projects alone. Do not re-distribute this pack in any means without first contacting Phantom Games Development and attaining proper permission (rfritzen@phantomdev.net).*

*Limits: One Pack per machine, and you are permitted to keep a backup copy on another machine. No web backups of this pack are permitted as a registered copy will allow you to re-download the pack if you need to.*

*Legal Stuff: This pack contains strong cryptography code. Usage of this pack is prohibited under federal law to any countries in which the United States has embargoed goods to, if you are a resident of or are herein affected by this clause, please remove this pack from your machine.*

*License: Without the technical jargon, I am not responsible for any damage this may cause to your machine, I'm using it on my lower end laptop without any issues so it should work fine for you. Also, the above statements apply. If you need a more serious license, you may see the included EULA file, but like I said, it's pretty much just this stuff here.*

## II. Pre-Install Work

As of version 1.3, the C++ section of the pack install no longer requires you to manually install the libraries into the engine, I have included some new changes to allow the project generator for T3D MIT to do this step for you. **If you have an existing project, the torque script files have been moved to the template folders. Just copy the files from there.**

As of version 2.0, the master server replacement has been included. I recommend re-installing the pack unless you're good at catching the changes. I have included a separate tutorial for upgrading from 1.5 to 2.0.

As of version 2.1, the pack is now requiring installation to be done on T3D MIT v3.5 and above, if you are unable to upgrade to this version, please refer to prior versions of MAP or seek help on the forums for proper pack installation.

As of version 2.2, you no longer need to use http://www.phantomdev.net/certAuthorityCreate.php, and a large amount of the installation process regarding MySQL and PHP has been removed by the insertion of an install.php script for the Authentication side of the pack.

Before we start, we need to add the new MAP files to Torque 3D's module system so it detects the changes. Copy the contents of the tools and templates folders to your Torque 3D folder, select the option to merge the folders and replace all existing files.

Inside the main MAP folder, you will see an *engine* folder. Copy the contents into your Torque3D's engine folder, this will install all of the new C++ code and the two new libraries to your copy of Torque 3D.

Before we begin the installation, you will need to generate a RSA keypair to serve as your certificate authority. This CA keypair will be used in future steps, so keep it handy once generated.

Navigate to the following page: http://www.phantomdev.net/certAuthorityCreate.php (If this page is down, send me an email at phantom139@phantomdev.net). At the bottom of the page there will be a few bolded fields, you will need those later, so copy the bolded fields (and what they are) to a text document or a word document (formatting is key, so don't mess with it). Once you have those, we can begin the installation. We will start with the engine modifications, so make sure you have a copy of T3D MIT that builds and runs fine before proceeding (You will need to run the project generator).

## III. Installation: Torque 3D Work

I recommend you do this on a clean copy of T3D MIT. Please remove any existing installation of the XXZ568 authentication package before continuing.

First off, we need to make some additions to Torque 3D to detect our add-ons. The way I coded MAP makes adding new modules easy enough (but I wrote a tutorial on that for your reading pleasures later on).

Open the *engine/app/mainLoop.cpp* file, and scroll down to the end of the #include statements. Now we're going to add a block of code there.

```
#define _LOADAUTH 1
#define _LOADSTORE 1
#define _LOAD_MYGAME 1
#define _LOADPGDPCON 1
#define _LOADPGDTCPCON 1
#include "PGD/Control/PGDMain.h"
```

This will be used in a bit. PGDMain.h contains a lot of definitions that will toggle these flags off and on as needed by your project, but again, a whole tutorial on that is available.

Next scroll down to `void StandardMainLoop::init()` and after the block with Con::init(); add this block of code:

```
   #ifdef _PGDAUTH
      Con::printf("PGD: xxz568 package activating...");
      xxz568::create();
   #endif
   #ifdef _LOADSTORE
      PGDStore::create();
   #endif
   #ifdef _LOADPGDPCON
      portControl::create();
   #endif
   #ifdef _LOADPGDTCPCON
      tcpCurl::create();
   #endif
```

Now with this section of code installed the engine will be able to use the new MAP code. Next, we need to add some code to the gameConnection class to use the authentication methods when players join servers.

Open *engine/T3D/gameBase/gameConnection.h* and at the end of the includes add these two lines

```
#define _LOADAUTH 1
```

```
#include "PGD/Control/PGDMain.h"
```

Now scroll down towards the bottom of the file. Eventually you will see methods for setting and getting whiteout and blackout. After those definitions add the following:

```
   #ifdef _PGDAUTH
      bool verifyClientSig(const char * signature, const char * details);
   #endif
```

Now open gameConnection.cpp and at the bottom of the file add this:

```
/**
PGD Authentiaction (xxz568)
**/
#ifndef _PGDAUTH
DefineConsoleMethod(GameConnection, verifyClientSignature, void, (),,
"(Disabled)") {
   Con::printf("This executable is running on a non-auth based game, this
function has been disabled for that purpose.");
}
#endif

#ifdef _PGDAUTH
DefineConsoleMethod(GameConnection, verifyClientSignature, void, (const char
*sig, const char *det),, "verifys the client signature on the server-side") {
   bool result = object->verifyClientSig(sig, det);
   if(result == true) {

   }
   else {
      Con::printf("Client Disconnected, invalid account signature.");
      object->setDisconnectReason("The server has denied your account
certificate");
      object->deleteObject();
   }
}

bool GameConnection::verifyClientSig(const char * signature, const char *
details) {
   std::string toWhrl = details;
   std::string toWhrl_final = toWhrl;
   toWhrl_final = cryptoPackage->pgdHash(toWhrl);
   std::string hexSig;
   hexSig.assign(std::string(signature));
   //
   bool result = cryptoPackage->caVerify(toWhrl_final, hexSig);
   return result;
}
#endif
```

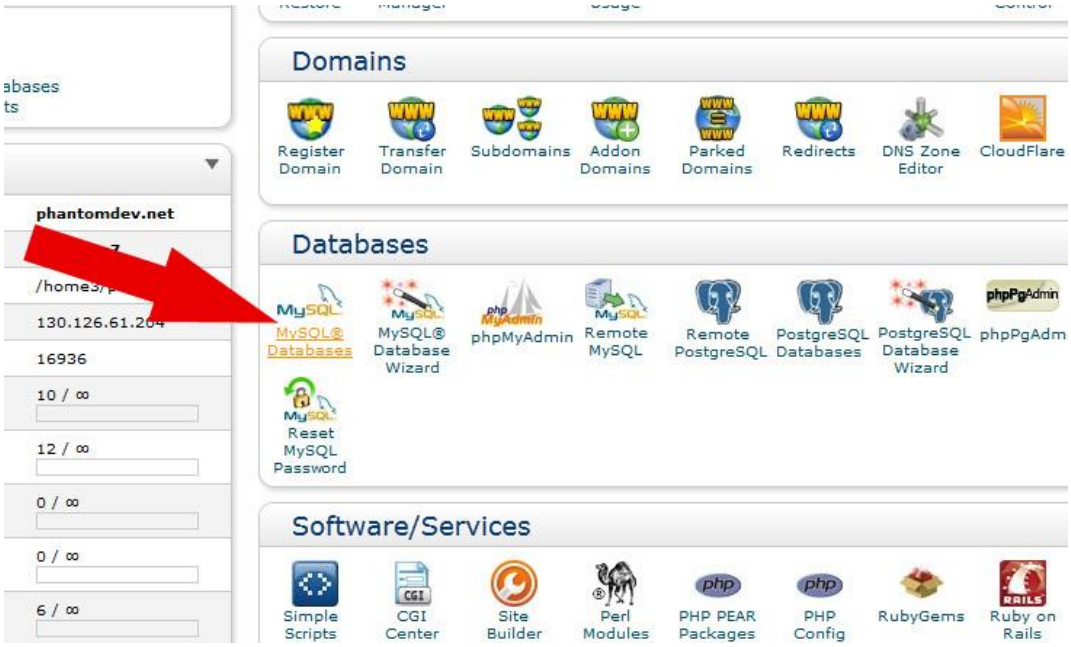
Save everything and build your solution. Make sure there are no errors due to linking problems, if there are, check over your library installs and try again. Once built, you may proceed to the next part of the installation, a last C++ addition will be made later on, but you will not need to do so until install.php instructs you to do so.

## IV. Installation: MySQL Databases

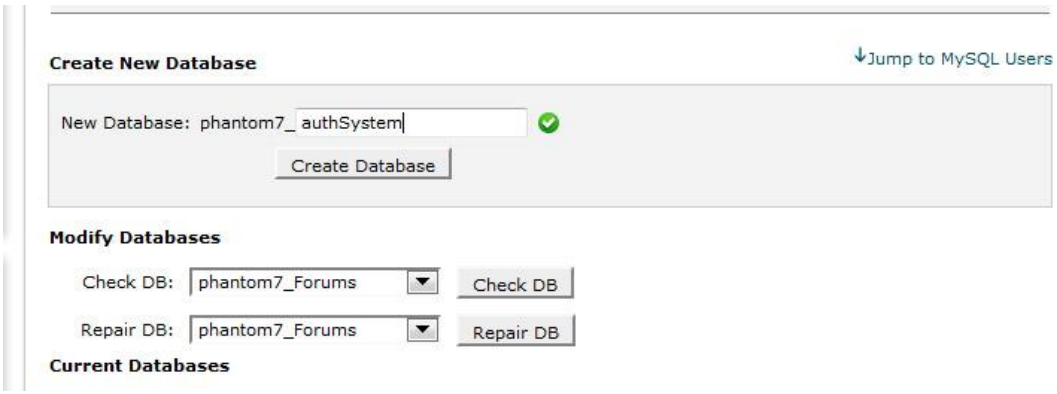
This section will vary from server to server. This is where you might need to step outside of this tutorial and pull open handy-dandy google if your host does not apply to this method. I'm assuming you're using a standard web-host here so they will likely have CPanel or something very similar to it.

As Mentioned by the XXZ568 resource, you need to create certain databases. I highly neglected this in my resource, so luckily for you since this is a purchased pack, I'm going to be extremely detailed this time in explaining everything that needs to be done, and I even have pictures to help you.

First, sign in to your web server. If you're using CPanel, you will have a display similar to the one shown below. Select the MySQL Databases button




Once you do so, a new window will open up. First we need to create our new database. This is where the information regarding user accounts and the account keys will be placed. Simple enough, scroll down to the "Create new Database" section, and type in the text box: *authSystem* and then click the associated button. For the visual learner:

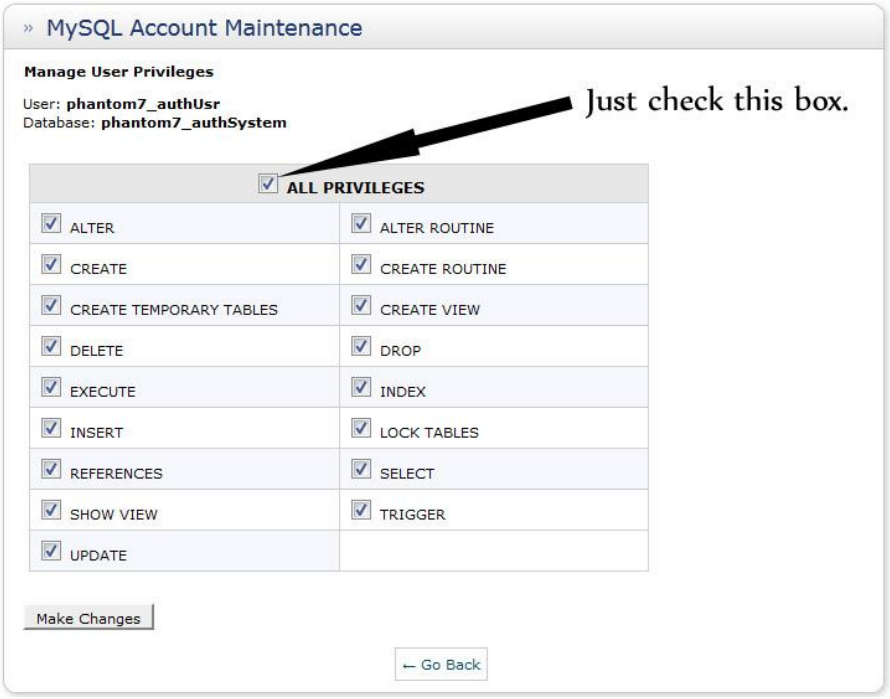



Now that the database is created, we need to add a user. A user is something MySQL uses to identify what sections of your website has access to what databases. For us, the only relevance is giving access to creating new accounts, and reading them, but since this is a database with limited tables, not much is needed. Scroll down to the create user section and input something along these lines:

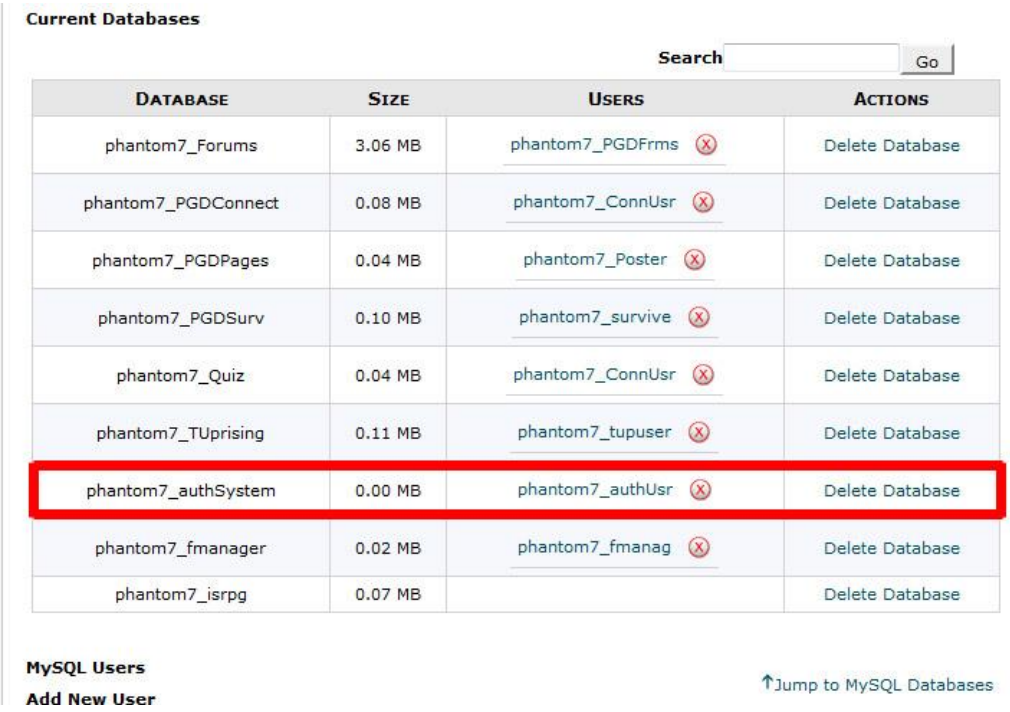

You can use the password generator if you feel necessary; just write it down as we will need it for our PHP scripts later. Once you click create user a confirmation page will appear just like last time.

Next step is to add the new user to our newly created database. This is very simply done in the labeled section add user to database.

Once you click add a new page regarding user permissions will appear. Since we are dealing with two tables that are only edited via one PHP script, giving all permissions will make the task much more simplistic:




Click the make changes button, and if all has been done correctly, you should now see something along these lines in your list of databases:



## V. Installation: PHP Work

Inside the MAP main folder, you will see a *php* folder, this contains the authentication system PHP scripts. Installation is very simplistic here. Create a new folder on your web server, and give it a simple name (*auth* works best). Then copy all of the scripts from the php folder into the *auth* folder.

Once placed on the web server, run the included install.php file and walk through the five step process to complete the final MySQL / PHP installation. Please delete the install.php file from your web server when completed.

## VI. Installation: Torque Script

The last thing we need to do is to set up the TorqueScript functioning. The final task is to tell the engine what file to point the authentication code to, or simply, where on the web our authentication server is.

Open the core/PGD/connectivity.cs file and the global variables at the top of the file with the web address, are to be changed to your web address and path to the authentication index.php file respectively.

The next thing we need to do is execute our new files, and make one quick edit. Open scripts/server/gameCore.cs and modify the top of `GameConnection::onConnect(%client, %name) {` to read the following:

```
   function GameConnection::onConnect(%client, %name) {
      //Phantom139: Begin authentication Process
      if(!%client.doneAuthing) {
         //echo("requesting account cert");
         commandToClient(%client, 'requestClientCert');
         return;
      }

      %authInfo = %client.getAuthenticationInfo();
      %accountName = getField(%authInfo, 0);
      %guid = getField(%authInfo, 1);
      %email = getField(%authInfo, 2);
      if(%name $= "") {
         %name = %accountName;
      }

      //Ask the client for their info now
      %client.namebase = %accountName;

      // Send down the connection error info, the client is responsible for
      // displaying this message if a connection error occurs.
      messageClient(%client, 'MsgConnectionError', "",
$Pref::Server::ConnectionError);

      // Send mission information to the client
```

```
	sendLoadInfoToClient(%client);

	%client.guid = %guid;
	addToServerGuidList(%client.guid);
```

Save the file, now open scripts/server/scriptExec.cs and add this line to the list of exec's:

```
exec("./clientChallenge.cs");
```

Now open  and add this block to the area where the exec's are:

```
exec("art/gui/AccountRegistrationWindow.gui");
exec("art/gui/LoginWindow.gui");

exec("./Auth/TCPConnections.cs");
exec("./Auth/GUIWindows.cs");
exec("./Auth/Login.cs");
exec("./Auth/Registration.cs");
exec("./Auth/ServerChallenge.cs");
```

Lastly, we will have the client load the login screen instead of the main menu so open scripts/client/init.cs and scroll down to function LoadMainMenu().

Replace the calls to MainMenuGui with LoginWindow.

Save all files, and you are done!

## VII. Finishing Up: Generating Account Keys

Obviously we're going to want to be able to create account keys as they drive the initiation of the authentication process. Doing this is quite simple, so I'll just put it all here. Somewhere on your website (related to where the user purchases your game) create a new php file named onBuy.php. Add these contents to it:

```
<?php
   //check this
   require("keyMake.php");
   //
   $email = strtolower($_POST["email"]);
   $key1 = KeyGen(5);
   $key2 = KeyGen(5);
   $key3 = KeyGen(5);
   $key4 = KeyGen(5);
   $key5 = KeyGen(5);
   // check for a valid email
   if(!isSet($email)) {
      die("Please enter a valid email address");
   }
   $key = $key1.$key2.$key3.$key4.$key5;
   ProcessKey($key, $email);
   echo "Account Entry Ready <p>";
   echo "Your Account key is: <font size=\"3\" face=\"Georgia, Arial\"
color=\"maroon\">".$key1."".$key2."".$key3."".$key4."".$key5."</b>";
   echo "<p>Common Errors: 0 - Zero, o - Letter, 1 - One, l -
Letter</font><p>";
   echo "<br/>You can copy and paste this into the game's register window!";
?>
```

This script will control the process after a user buys your game. In fact, when you bought this pack, you triggered a php script of similar fashion that added your email address with a related key to my server, which was then used in part 1 to register your copy of MAP. If you are going to use a paypal system similar to the one I did, I'd advise you take a peek at this article: http://stackoverflow.com/questions/12043448/running-a-script-when-i-receive-paypal-payment, as this helped me set up my system. You would paste the inner contents of this script inside the "on validated payment" section. Now, are you the script kiddie who is noticing a missing piece? Good. I have neglected to give you keyMake.php, so let's do this now.

Create a new php file named keyMake.php in the same directory as onBuy.php and paste the following into it:

```
<?php
   //keyMake.php
   //Functions needed to create account keys!
   function PGDHash($string) {
      $interior = sha1($string);
```

```
      $exterior = hash('whirlpool', $interior);
      return $exterior;
   }

   function KeyGen($len) {
      $key = "";
      $alphanumeric = array("a", "b", "c", "d", "e", "f", "g", "h", "i", "j",
"k", "l", "m",
         "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z",
"0", "1", "2", "3",
            "4", "5", "6", "7", "8", "9");
      for($i = 1; $i <= $len; $i++) {
         $key = $key.$alphanumeric[rand(0, 34)];
      }
      return $key;
   }

   function ProcessKey($key, $email) {
      $con = new mysqli("localhost", "host_mySQLUser", "mySQLPass",
"host_authSystem");

      /* check connection */
      if (mysqli_connect_errno()) {
         printf("Connect failed: %s\n", mysqli_connect_error());
         return -1;
      }
      $pCode = rand(10000000, 99999999);


      $Finalkey = $con->real_escape_string(PGDHash($key));
      $email = $con->real_escape_string($email);
      $stmt = $con->prepare('INSERT INTO AccountKeys (Email, AccKey,
PurchaseCode) VALUES (?, ?, ?)');

      $stmt->bind_param('sss', $email, $Finalkey, $pCode);

      $stmt->execute();

      $con->close();
   }

?>
```

Oh hey look, it's your favorite friends, the MySQL fields ;). Be sure to update those to match the correct information (part 4), save your work and put it on your server. And we are now all set!

In case you need help: `$con = new mysqli("localhost", "host_mySQLUser", "mySQLPass", "host_authSystem");` is what you need to change.

## VIII. Final Remarks

Everything is now in place, and you are ready for a test run. Fire up your new game client, visit the onBuy.php file on your web site to generate your key (make sure you set up an HTML POST form to send the email field, just google it if you need help), and then register your account. If everything was properly set up, you will now have a certs.xml file generated in your user data folder, and the MySQL table (Accounts) will now have one new entry for your account! If this is the case, you my friend, have successfully installed MAP and are now officially ready to begin developing your Multiplayer Game under a secured environment!

Some final remarks: Feel free to edit any of the GUI files, you purchased this pack, and so you have full control over how you want your user login screen and registration window to look.

The C++ source code contains some setup variables to adjust the level of security on your RSA keys, feel free to change these to adjust server key sizes, or client certificate sizes. I prefer keys between 512 and 1024 of length. 512 is my absolute minimum, because security can become a problem with key sizes any lower.

Once again, I want to thank you for purchasing my first pack. I'm looking forward to all of the great things that come out of this pack, and I hope to deliver even more exciting new features and systems to Torque in the future! And please, visit the forums, even if you don't have MAP related questions, comments or concerns, I like to converse with people, and the more people who visit my forums, the better. I might be even inclined to offer a few things to you guys ( hint hint ;) ). Thanks again, and enjoy MAP!