

## Welcome to the MQ2 development model

[Top](#) [Next](#)

MQ2 is developed with ease of use, customizability, and ease of development in mind. The major improvements in MQ2 over old MQ are in the customizability and development areas. Because development has so radically changed from the original path, new and old MQ developers alike are going to need a kickstart with the new system.

Adding commands, macro parameters, aliases, and detours has never been easier. The MQ2 API provides functions to add and remove any of these dynamically, at any time. Instead of maintaining cumbersome arrays, the API automatically maintains its own cumbersome linked lists! More info in... somewhere that will be made soon.

eqgame.ini has been shortened quite a bit by hardcoding many offsets into eqgame.h. More specifically, offsets of class instances and class member functions have all been moved there. At the same time, most of the EQADDR\_XXXXX variables have been removed completely. This may cause some confusion at first, but once you realize what's going on the lightbulb over your head will come on and all is well. You can find the list of replacements here [EQADDR\\_\\* Replacement Index](#)

All EQ classes, at least those from the leaked .map file, are declared in EQClasses.h. Now, every class member function from that time is available to us directly. We use a neat trick to take those function declarations, along with their offset, to tell the compiler how to call them directly. At this time, most classes do not have the data members, although they certainly could. To get the data from a class pointer, simply cast to the struct. More on this here [EQ Classes](#)

## Creating a plugin

[Previous](#) [Top](#) [Next](#)

Coming soon. until then look at [MQ2PluginDevelopment.html](#)

The MQ2 API includes namespaces in order to provide logical divisions for finding variables, structures and classes that you may have difficulty finding otherwise. All of the namespaces are "used" so you do not actually need to use the :: scope operator unless you happen to use that method for looking through the namespaces. The current namespaces are as follows:

- **EQClasses** - Contains all classes from EQ (e.g. CBreathWnd, CBookWnd)
- **EQData** - Contains all EQ class data structures (e.g. SPAWNINFO, CHARINFO) except for UI
- **EQUIStructs** - Contains all EQ UI class data structures (e.g. CXWND, CSIDLWND)
- **MQ2Globals** - Contains all MQ2 global variables (e.g. ppBreathWnd, ppBookWnd)
- **MQ2Internal** - Contains all MQ2 internally used structures (e.g. MQCOMMAND, PARMLIST)
- **MQ2Prototypes** - Contains all function prototypes (e.g. fEQCommand, fEQLoadSpells)

In Visual Studio .NET you can browse through these in the Class View window. Expand MQ2Main and the namespaces appear. Then you try to find what you're looking for ;) Another way is to use the scope operator by typing out EQClasses:: and waiting for the context menu to pop up to scroll through.

MQ2 differs from original MacroQuest partly because the MQ2 API covers stealing EQ's C++ classes for our use. This means no inline assembly required to call EQ functions. All you need to have is the function's offset, and voila, instant access to the function through C++ class member function calls. This difference confuses new developers at first, because they're used to having to do inline assembly and referring to the class as a struct. Now, we can place the struct inside of the class directly and use the class as if it were a class...which is just like development within the actual EQ client.

MQ2 is still in a period of transition into the new methods. Eventually, classes we use data from will have an instance of the data struct we use to access the class data. Until then, you can explicitly cast from a pointer to the class to a pointer to the struct, and vice versa. In other words, `EQ_Character*` is exactly the same as `PCHARINFO`. The compiler doesn't realize this, and if you want to ... `PCHARINFO pCharInfo=pCharData;` (note that `pCharData` is `EQ_Character*` type, and globally defined in the MQ2 API) .. then you MUST explicitly cast to `PCHARINFO` like this: `PCHARINFO pCharInfo=(PCHARINFO)pCharData;`

Every possible UI window and all other classes previously used have pointers for access from the MQ2 API.

### Debugging Crashes

The absolute first thing to do when debugging crashes is to attach a debugger. As silly as it sounds, most of you reading this probably didn't even consider it. In MS Visual Studio, pull down Debug and hit Processes.. Then double click on eqgame.exe assuming you have it running. Leave "Native" selected in the program types box and just hit OK. Close the process list box and you're done with that. If you open the output window (CTRL+ALT+O) in VS, you can see the DebugSpew. Go make it crash, and hopefully it will break into the debugger and show you the line that it broke on.

Often it's difficult for you to figure out what's going on, since it may crash inside EQ and not be able to show you the line that broke. If this is directly caused by your code and you cannot determine the line that's crashing, MQ2 provides a way for you to debug this. Place `DebugTry(whatever);` around each line of code you think may be crashing. For example, if a line of code says `"pTarget->Something(x);"` you can make it **`DebugTry(pTarget->Something(x));`**. `DebugTry` is something you must turn on for it to do anything at all. If it's not turned on, nothing extra at all is added to the compiled output. To turn it on, you need to **`#define DEBUG_TRY 1`** before `MQ2Main.h` is included. If you are working within a plugin, you need to do the define before `"../MQ2Plugin.h"` is included. What `DebugTry` will do is output some `DebugSpew` directly before and after the code that gets executed. By looking at the resulting debug spew, you can determine exactly which line is causing the problem. Fix that line and you have solved the current problem. Then you can disable `DEBUG_TRY` and have the choice of removing the `DebugTry()` around your lines of code or leaving it there. Again, if you leave it there and do not turn on `DEBUG_TRY`, it will do absolutely nothing to the compiled output.

## Variables

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## Functions

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## AddAlias

[Previous](#) [Top](#) [Next](#)

Enter topic text here.



## AddCommand

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## AddCustomEvent

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## AddDetour

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## AddFilter

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## AddMacroLine

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## AddParm

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## AppendCXXStr

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## CheckChatForEvent

[Previous](#) [Top](#) [Next](#)

Enter topic text here.



## CompareTimes

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## ConColor

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## ConColorToARGB

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## ConvertHotkeyNameToKeyName

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## ConvertItemTags

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

Enter topic text here.

# DebugSpewAlways

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## DefaultFilters

[Previous](#) [Top](#) [Next](#)

Enter topic text here.



## DistanceToSpawn

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## DoCommand

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## FindContainerForContents

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

Enter topic text here.

## FindSpeed

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

Enter topic text here.

## GetCharInfo

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

Enter topic text here.



## GetCXStr

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## GetDeityTeamByID

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## GetEnviroContainer

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## GetEQPath

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## GetFilenameFromFullPath

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## GetFuncParamName

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## GetGuildByID

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## GetGuildIDByName

[Previous](#) [Top](#) [Next](#)

Enter topic text here.



## GetItemLink

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## GetItemLinkHash

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## GetLightForSpawn

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## GetLoginName

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

Enter topic text here.

Enter topic text here.

Enter topic text here.

## GetSpellByID

[Previous](#) [Top](#) [Next](#)

Enter topic text here.



## GetSpellByName

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## GetSpellDuration

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## GetSubFromLine

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## LoadMQ2Plugin

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## MQToSTML

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## **RemoveAlias**

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## RemoveCommand

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## **RemoveDetour**

[Previous](#) [Top](#) [Next](#)

Enter topic text here.



## **RemoveParm**

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## SetCXStr

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## STMLToPlainText

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

Enter topic text here.

## UnloadMQ2Plugin

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## WriteChatColor

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## EQ Classes

[Previous](#) [Top](#) [Next](#)

Enter topic text here.

## Developing new API functionality

[Previous](#) [Top](#) [Next](#)

There is literally no reason to develop directly inside the API unless you are modifying existing features. Developing inside the API is only going to confuse you and make your life more difficult. For example, developing directly in the API you must unload MQ2 completely and reload before continuing a test. Instead, do your developing in a plugin. The code can then be copied directly and with very little modification into the API if necessary. Plugins can always be unloaded and reloaded on the fly without taking extra steps of shutting down MQ2 completely. In this way you can also take advantage of the MQ2 framerate limiter while developing, and not have to sit around or close EQ while your work compiles.



## EQADDR\_\* Replacement Index

[Previous](#) [Top](#) [Next](#)

All of these have been replaced using the actual classes. You can cast directly to the old structs using the replacements. Most of these are double pointers, denoted by ppWhatever. pWhatever is automatically defined as (\*ppWhatever). In other words, if you previously used (!EQADDR\_TARGET || !\*EQADDR\_TARGET), this is now (!ppTarget || !pTarget). Then, to cast to PPSAWNINFO from pTarget do this: ((PPSAWNINFO)pTarget)->whatever.

**Here is a direct comparison list. Note that some old offsets were duplicates of others and/or incorrectly named**

- **EQADDR\_ACTIVECORPSE** - unused
- **EQADDR\_ACTIVEMERCHANT** - EQPlayer \*\*ppActiveMerchant - EQPlayer \*pActiveMerchant
- **EQADDR\_CASTINGWND** - CCastingWnd \*\*ppCastingWnd - CCastingWnd \*pCastingWnd
- **EQADDR\_CHAR** - EQPlayer \*\*ppCharSpawn - EQPlayer \*pCharSpawn
- **EQADDR\_CHAR\_INFO** - EQ\_Character \*\*ppCharData - EQ\_Character \*pCharData
- **EQADDR\_CLASSBANKWND** - CBankWnd \*\*ppBankWnd - CBankWnd \*pBankWnd
- **EQADDR\_CLASSCASTSPELLWND** - CCastSpellWnd \*\*ppCastSpellWnd - CCastSpellWnd \*pCastSpellWnd
- **EQADDR\_CLASSCONTAINERMGR** - CContainerMgr \*\*ppContainerMgr - CContainerMgr \*pContainerMgr
- **EQADDR\_CLASSDISPLAYOBJECT** - CDisplay \*\*ppDisplay - CDisplay \*pDisplay
- **EQADDR\_CLASSGIVEWND** - CGiveWnd \*\*ppGiveWnd - CGiveWnd \*pGiveWnd
- **EQADDR\_CLASSHOTBUTTONWND** - CHotButtonWnd \*\*ppHotButtonWnd - CHotButtonWnd \*pHotButtonWnd
- **EQADDR\_CLASSMAPWND** - CMapViewWnd \*\*ppMapViewWnd - CMapViewWnd \*pMapViewWnd

- **EQADDR\_CLASSMERCHANTWND** - CMerchantWnd \*\*ppMerchantWnd - CMerchantWnd \*pMerchantWnd
- **EQADDR\_CLASSNOTESWND** - CNoteWnd \*\*ppNoteWnd - CNoteWnd \*pNoteWnd
- **EQADDR\_CLASSTEXTUREANIMATION** - CInvSlot \*\*ppSelectedItem - CInvSlot \*pSelectedItem
- **EQADDR\_CLSITEMS** - CDisplay \*\*ppDisplay - CDisplay \*pDisplay
- **EQADDR\_CLSMAINNEWUI** - CEverQuest \*\*ppEverQuest - CEverQuest \*pEverQuest
- **EQADDR\_CLSSPAWNS** - **unused**
- **EQADDR\_DOORS** - EqSwitchManager \*\*ppSwitchMgr - EqSwitchManager \*pSwitchMgr
- **EQADDR\_GROUP** - EQPlayer \*\*ppGroup - EQPlayer \*pGroup
- **EQADDR\_INVENTORYWND** - CInventoryWnd \*\*ppInventoryWnd - CInventoryWnd \*pInventoryWnd
- **EQADDR\_ITEMS** - EQItemList \*\*ppItemList - EQItemList \*pItemList
- **EQADDR\_LOOTWND** - CLootWnd \*\*ppLootWnd - CLootWnd \*pLootWnd
- **EQADDR\_PACKLOCS** - **unused**
- **EQADDR\_SPAWNLIST** - EQPlayer \*\*ppSpawnList - EQPlayer \*pSpawnList
- **EQADDR\_SPAWNTAIL** - EQPlayer \*\*ppSpawnListTail - EQPlayer \*pSpawnListTail
- **EQADDR\_SPELLBOOKWND** - CSpellBookWnd \*\*ppSpellBookWnd - CSpellBookWnd \*pSpellBookWnd
- **EQADDR\_SPELLFAVORITES** - SPELLFAVORITE \*pSpellSets
- **EQADDR\_SPELLS** - SpellManager \*\*ppSpellMgr - SpellManager \*pSpellMgr
- **EQADDR\_TARGET** - EQPlayer \*\*ppTarget - EQPlayer \*pTarget
- **EQADDR\_ZONEINFO** - EQZoneInfo \*pZoneInfo
- **EQADDR\_ZONELIST** - EQWorldData \*\*ppWorldData - EQWorldData \*pWorldData

## Frequently Asked Questions

[Previous](#) [Top](#)

**Q:** How do I access EQADDR\_(something) in MQ2?

**A:** If you cant access what you used to use, it was replaced with something more suitable. For example, **EQADDR\_CLASSDISPLAYOBJECT** was replaced with the variable **ppDisplay**. They are essentially the same thing. However, now we have defined **pDisplay** as **(\*ppDisplay)**. So instead of saying **if (!EQADDR\_CLASSDISPLAYOBJECT || !\*EQADDR\_CLASSDISPLAYOBJECT) return;** You would now use **if (!ppDisplay || !pDisplay) return;**. You can also access the CDisplay class (the one from EQ) member functions directly by **pDisplay->Function(blah);**. However, unless the offset of the function is defined, attempting to compile will fail on the LINK step.

**Q:** I don't see class member functions with Visual Studio 6.0 when I type something like **pDisplay-> ...** what gives?

**A:** VS6 isn't calculating the #define on the fly. **pDisplay** is a #define of **(\*ppDisplay)**, so if you want to see the member functions start by using **CDisplay::** or **(\*ppDisplay)->** to see the functions. Make sure to go to the Class view in the workspace window and click on the MQ2Main project. From there you can see all of the classes fill in.