Admin Mod Version 2.50e

Documentation and Online Help

June 15, 2001
v2.50e.150601

Please visit our web site at http://www.adminmod.org/

Admin Mod works alongside Metamod
Check out Metamod at http://metamod.org/

The latest version of the Admin Mod documentation can always be found online, at the Admin Mod web site: http://www.adminmod.org/help

Visit the Admin Mod web forums to discuss Admin Mod with others and to get help when you can't find answers to your questions in the manual.

Documentation Revision Information:

v2.50e.150601
v2.50d.060601

Revision information for recent release is available in the Reference Section

Table of Contents

What's New?

Admin Mod 2.50 has a load of new commands and functions, making it more accessible and powerful. Some are obvious, such as the implementation of plugin scripts, and others are less so. Below is a high-level list of the changes in the last few releases of Admin Mod.

New in version 2.50e release (June 15, 2001):

- Version 2.50e contains several bug-fixes and performance enhancements. See the change logs in the Admin Mod Reference section for details.
- The procedure for using Bots with Admin Mod has been changed, and is now simpler. Please see How to Use Bots with Admin Mod for more information.
- Updated the credits for the A-Team in the readme and docs.

New in version 2.50d maintenance/bug-fix release (June 2001):

Among the most notable of the changes made for 2.50d are:

Client Authentication instructions and procedures modified
For security reasons, the installer will no longer create an autoexec.cfg file. Instead, it creates a file called adminpass.cfg , which you can either exec from the console, or you can add it to a desktop shortcut just for connecting to your server. More information is available in the section Required Client Configuration Instructions.

New CVARs: admin_repeat_freq, admin_vote_echo, amv_autoban
You now have the ability to specify in your server.cfg file how often the center-say repeating message should appear. You also can control voting display a little better, and an auto-kick/ban option has been added for dealing with players that use non-printable characters in the game to crash people.

Access levels chaged for admin_exec functions
All admin_exec functions have now been changed to a higher access level. See that section of the documentation for more information.

You can now use * without it being recognized as a comment
This was causing some confusion and unnecessary tech support requests. It has been fixed in this version.

Improved bot protection
If you use bots with Admin Mod, you probably know about the admin_bot_protection cvar setting already (See How to Use Bots with Admin Mod for more info). Bot protection has been enhanced, so it's even safer now to

run bots on your server.

Changes made to the sample wordlist.txt file
To assure people are happy and content, and because it seemed like a worn out joke, one word has been removed from the sample wordlist.txt file that is used for filtering "bad" words on the server.

A partial list of changes introduced in v2.50:

Plug-in Scripts!
You can now use multiple scripts. Yes, that is correct, you can use multiple scripts - we call them plugins now. Thanks to the plugin.ini file, simply list the path to the plugin script you wish to use. This feature can easily be disabled if you prefer to stick to the old way of doing everything in one big script, but we think if you give the plugin setup a try, you'll love it.

Metamod Integration
Metamod allows you to add features from multiple, separate DLLs. Under that framework, Admin Mod is a sort of plugin itself, and can be run alongside other applications, like bots. Check out the Metamod web site for more information.

"admin_command" allows commands to be entered in server console or RCON
You are now able to enter commands directly into the HLDS console window, or via rcon . Enter admin_command followed by the normal command you would enter if you were in game. For example, to do a admin_csay while you are working at the HLDS console, just enter:

```
admin_command admin_csay All your base are belong to us.
```

Partial name matching is now available.
Instead of having to enter the whole name of player to execute a command on him, you can enter only a portion of his name. However, that portion of his name must be unique to his name. For example, to slay a player called Superkalifragilisticespialidocous (i can't spell it, its atrocious) enter:

```
admin_slay kalifrag
```

and if no one else on the server has the string "kalifrag" in their name, the above-mentioned player will go dead. This works with all commands where you would specify a player name. As long as you specify a unique name or partial name match, the user will be found.

The vote option chosen by each user is echoed to all clients

This is optional and can be turned off via a cvar in the (listen)server.cfg file.

No more need for "old password_timeout" cvar
There is no longer a password_timeout problem, thanks to our next new friend:

admin_reconnect_timeout
The admin_reconnect_timeout cvar allows users to reconnect for a few minutes after dropping from the server, without having to re-enter a password. You just set how long you want people t be able to be gone for before they have to re-enter or re-exec their authentication info.

admin_connect_msg
Every client receives a customized text message 30 seconds after connecting, welcoming them to the server.

admin_highlander
There can only be one. One admin that is. When multiple administrators are connected to a server, only the one with the highest access level will have access to the admin commands.

admin_fun_mode
If you get bored, enable the fun mode to make things a little more spicy. This cvar gives you access to a multitude of other commands that will keep you laughing ... til the next release.

admin_reject_msg
All rejection messages are easily configurable by setting this cvar. If someone tries to use an un-authorized commands, he will be politely informed.

admin_repeat_msg
How do I change that green thing that appears every ten minutes and says, "This server uses Admin Mod?" Simple really: Change this cvar in the server.cfg or listenserver.cfg and it will be whatever you want it to be.

admin_quiet
A classic. The old verbosity setting has been changed to this handy little option, setting which admin commands are echoed to the clients, and which are not.

admin_vault_file
To keep your admin settings over map changes, enable admin_vault_file in your server configuration file. Can be used by plugins to store and retain information over map changes and server restarts.

admin_vote_maxextend
No need to go compile scripts to change the number of times a map can be extended by a vote.

admin_disco
This cannot be explained, and neither can its counterparts. Go see it for yourself, but enable fun mode first. **:)**

admin_restartround
Self explanatory.

admin_denymap and say denymap
If you are the admin and you are sick of playing a certain map that keeps on getting voted for, deny it and its votes will be disregarded.

admin_prematch
TFC only. To get warmed up and ready for clan matches.

About Admin Mod

Introducing Admin Mod

Admin Mod is a modification add-on for the Half-Life Dedicated server , commonly used for Counter-Strike and Team Fortress Internet game play, among a significant list of other modifications (commonly called “mods”) of the original Half Life game. A list of supported game mods appears below.

What is Admin Mod?

This server-side modification application will let you give people access to manage and administer your HLDS (Half-Life Dedicated Server) machines, without giving direct access to rcon (server-speak terminology for “Remote Console”). It also lets you define users who can access the server to play, each with a different password, and allows you to assign players the ability to execute Admin Mod's commands during the game. The security is configurable per-player, meaning you can assign each player differing access levels, and therefore allow or restrict any given set of Admin Mod commands.

It is a plug-in modification designed to work alongside the following mods.

(Note that some of the mods listed may have changed versions or been updated since this list was last updated, and therefore may not be completely compatible. Also note that there may be mods that are not list which Admin Mod will work with. Always use the latest version of the MetaMod DLL file, available from the MetaMod web site, to assure you are current and able to use Admin Mod with the latest supported games.)

- Judgement
- FireArms rc2.5
- DoD 1.1
- DeathMatch Classic mod (Valve) v1
- Phineas (v0.21): The normal Half-Life mod with bots !
- Golden Eye (1.7): You loved it on the N64, now its on the PC!
- Counter-Strike (1.1): The cool counter terrorist mod
- Action Half-Life
- Freeze: (untested)
- Botman
- TFC (v1.1.0.6):
- Valve (v1.1.0.6): The standard Half-Life death match
- Oz (v1.7): Extended death match
- SvenCoop (v1.3): You have to try this
- Science and Industry (v0.96): A unique idea with a great execution.
- Front Line Force (v1.1): Join the battle on the front lines.
- WizardWars (beta 2): Zap, kapowi, spin up a spell.

- GangstaWars (beta 2.5): Become the don!
- Arg! (one): Become a pirate!
- Swarm (2.1.0.4):   Buzz?
- Opposing Force: Bang?
- Global Warfare
- Jailbreak
- Paintball
- Vampire Slayer
- Wasteland
- Others: (Tell us if it works - or if it doesn't)

(NOTE - If you are a mod author contact Will Day at http://www.meta-mod.org so he can add support for your mod, its easy to do. He will need a complete list of the game's entities)

With the advent of version 2.50, all Admin Mod command functionality has been made accessible via the dedicated server console, meaning that server owners and admins can now execute Admin Mod functions without being in the game itself. To take advantage of this, the server admin would either enter the commands directly in the dedicated server window, or would use a remote console tool to “rcon” into the dedicated server and execute commands. Any Admin Mod command can be executed in this fashion by preceding the desired command with "admin_command" - So to do an "admin_say Hello there..." command in the HLDS window, one would simply type: "admin_command admin_say Hello there..."

As Alfred and the team have written this mod they have taken security into mind, so there should be no problems with the code (but there is always a possibility... Remember that security is an ongoing process both for software authors and the people who use the software). Source code is made available when it is stable via the Admin Mod web site.

If you are interested in contacting Alfred, the original author of the Admin Mod, or if you are interested in participating in the community of server administrators who use the Admin Mod on their servers, please see “Where to go for more help” at the end of this document.

What this document covers (and what it doesn’t)

This documentation specifically covers the version 2.50e release of Admin Mod. This was a bug and maintenance update release designed to replace the last major version of Admin Mod, version 2.50.

Who should use Admin Mod - and who should not

***Simply put:** If you don't know what a file extension is or how to make your operating system display them to the user, or if you don't understand how to set up a Half-Life server without Admin Mod, or if you have not even set up your server and tested it to make sure it is running smoothly, **PLEASE, STOP NOW**. Admin Mod is probably not for you (at least not yet, anyhow). We're being serious here - Admin Mod is a server application that requires some practical knowledge of how computers work and the operating relationship between servers and their clients, etc. Just keep in mind that if you dive in too early, you may be frustrated and disappointed with the results.*

There. You have been forewarned. **:)**

This documentation assumes that you already have a working Half-Life Server up and running. Your server may be a dedicated (stand-alone) server, or it maybe one where you start it in-game and host a temporary server (called a listen server). If you do not already have a stable Half-Life server up, running, tested and confirmed to be bug-free, *do not install Admin Mod*. Instead, fix whatever HLDS problems you have and install Admin Mod only after you have a working server.

> ALWAYS MAKE BACK-UPS OF YOUR CONFIGURATION BEFORE INSTALLING ANY SOFTWARE THAT MODIFIES ANY PROGRAM ON A COMPUTER, INCLUDING HALF-LIFE AND HLDS. ADMIN MOD'S INSTALLATION PROCEDURE REQUIRES THE ALTERATION OF SEVERAL FILES THAT ALREADY EXIST IN YOUR HALF-LIFE SETUP.

We have not attempted to document how to install a HLDS system in this manual – This documentation explains how to install, configure and use the Admin Mod. If you don't know what HLDS means, you need to stop now - You have plenty to do and learn before Admin Mod will do you any good.

Admin Mod Version 2.50 and later represents a significant number of changes over previous versions, so be sure you are using the same version of the mod as covered by this document. The software version covered can always be found on the front page of this documentation.

Important Information for Before You Start

Introductory Information (or, Read Me First)

Setting up Half-Life Admin Mod isn't all that difficult nor tedious; it just requires some understanding of what's going on. It actually could (in a perfect world) take you less than ten minutes to get it up and running after you read this. We will address each file you need to configure separately. Hopefully this will ease up on the confusion.

READ THIS DOCUMENTATION <u>BEFORE</u> YOU START

We recommend highly that you **read this entire document** before attempting your first installation of the Admin Mod, including the key information presented below. The basics presented here will carry you a long way in understanding how and why it works the way it does. We're not kidding. If you don't have the time or inclination to read the documentation in detail, you probably should not be running a server with Admin Mod.

IMPORTANT NOTE ABOUT EDITING CONFIGURATION FILES

In order to properly edit the configuration files supplied with and created by Admin Mod, you MUST use a PLAIN TEXT editor. You *CANNOT* use editors such as Microsoft Word or WordPerfect or anything along those lines. Most Windows server administrators use Notepad, which is part of every Windows setup. Another good text editor you might want to try that is popular with Admin Mod server administrators is The Programmer's File Editor. Any plain text editor will work just fine.

HOW ADMIN MOD WORKS

Admin Mod is an application that runs on top of the Half Life Dedicated server (or HLDS for short). It works in conjunction with another application called MetaMod, which enables the HLDS to have multiple other helper applications running at any given point in time. MetaMod allows Admin Mod to support any number of modifications of the Half-Life game, and since MetaMod takes on the task of integrating and assuring compatibility with game mods, the developers of helper applications like Admin Mod can focus on growing and expanding their own programs' capabilities.

MetaMod essentially sits between the engine and game DLL , catching routines, and passing them on. thus modifying functionality. However, rather than providing just the additional server features built in to the Admin Mod DLL, MetaMod allows you to add features from multiple, separate DLLs. Under that framework, Admin Mod can be a plugin itself, and be run alongside numerous other plugins. And, as of version 2.50 of Admin Mod, AM has now transitioned to running as a MetaMod plugin.

Admin Mod intercept messages between the server and its clients. A " client " machine in this case is the computer that a person uses to connect to the game server - So when we say "clients," we mean the computers on which people are playing the game.

Admin Mod is able to provide a whole new level of alongside the Half-Life server application. With it's integrated security model, a server owner can delegate administrative authority to individuals. Admin Mod also incorporates some functionality that would probably better be described as "fun" rather than "administrative." But it's all part of the same application, and it's all completely configurable by the server owner, just as long as they have some idea about how to set it up properly.

HOW ADMIN MOD IS ORGANIZED

Admin Mod uses a set of what are called “.ini” and " .cfg " files to let the mod know about certain information it needs to be able to function. The information that you put in these files includes things like the names of the players that you want to allow to use the Admin Mod commands, their passwords, the maps that you want to allow people to vote for on the server, what Admin Mod options you want to have enabled on your game server, and a variety of other important and useful things. For that reason, please take the time to read each of the sample .ini and .cfg files that are provided with Admin Mod, as discussed later.

In addition to the configuration files discussed above, Admin Mod allows the use of a scripting language to define features and functionality of the application itself. On other words, the plugin "scripts" that come with Admin Mod are what provide its core functionality and features. However, the beauty of the system is that users can develop their own plugins for Admin Mod, thus extending the features and functionality of the mod on their servers. Scripting and compiling is discussed later in this document in part, and in more detail on the AScript web site. In addition, the AScript site has a large library of plugins developed by members of the Admin Mod community, freely available to download and use.

Next: Default Windows Installation of Admin Mod

Installing and Configuring Admin Mod 2.50

Default Windows Installation of Admin Mod

First of all, if you have not yet read the Introductory Information page in the "Before you start" section of this document, you should do that now. In that section we discuss some key concepts as well as the differences between Admin Mod version 2.50e and prior versions. There are some key differences, so whether you are new to this or if you have been working with Admin Mod since it was first born, that section is important to become familiar with.

Admin Mod can be installed on any Windows platform that supports the HLDS server application, meaning any of the following (yes, it can also be installed on Linux, a little of which is covered later in this documentation):

- Windows 95 **
- Windows 98 **
- Windows NT4
- Windows ME **
- Windows 2000
- Windows XP
- etc.

> ** Note: The consumer versions of Windows (95/98/ME) are not recommended for running HLDS under any circumstances. It will work, but server performance and reliability will not be acceptable in many cases. Recommended versions of Windows as of the time of this publication are Windows NT4 and Windows 2000.

So, if you have a server running on a 32-bit Windows-based machine, you can successfully use the Admin Mod. We can’t guarantee the HLDS server will run well in all setups, but from extensive experience we can tell you that Admin Mod will work in these configurations.

For those of you performing Windows installations, a Visual Basic script is supplied that automates and simplifies installation of the mod, as long as you have a fairly typical installation of Half Life and any game mods. If you have specified any ultra-custom paths for your mods or half-life, the installation script may get confused, and you will need to do a manual installation which is covered later in this documentation.

There are a few important things to consider from the onset – if you are skimming this file, stop and read this next part carefully:

THE FIRST THING YOU SHOULD DO IS BACK UP YOUR CONFIGURATION AND OTHER CUSTOM FILES!!!

If you think you'll figure it out later, you might just be in for a surprise. Now, Admin Mod is easy to install, but our experience has shown us that people tend to skip this step. So, it bears repeating:

THE FIRST THING YOU SHOULD DO IS BACK UP YOUR CONFIGURATION AND OTHER CUSTOM FILES!!!

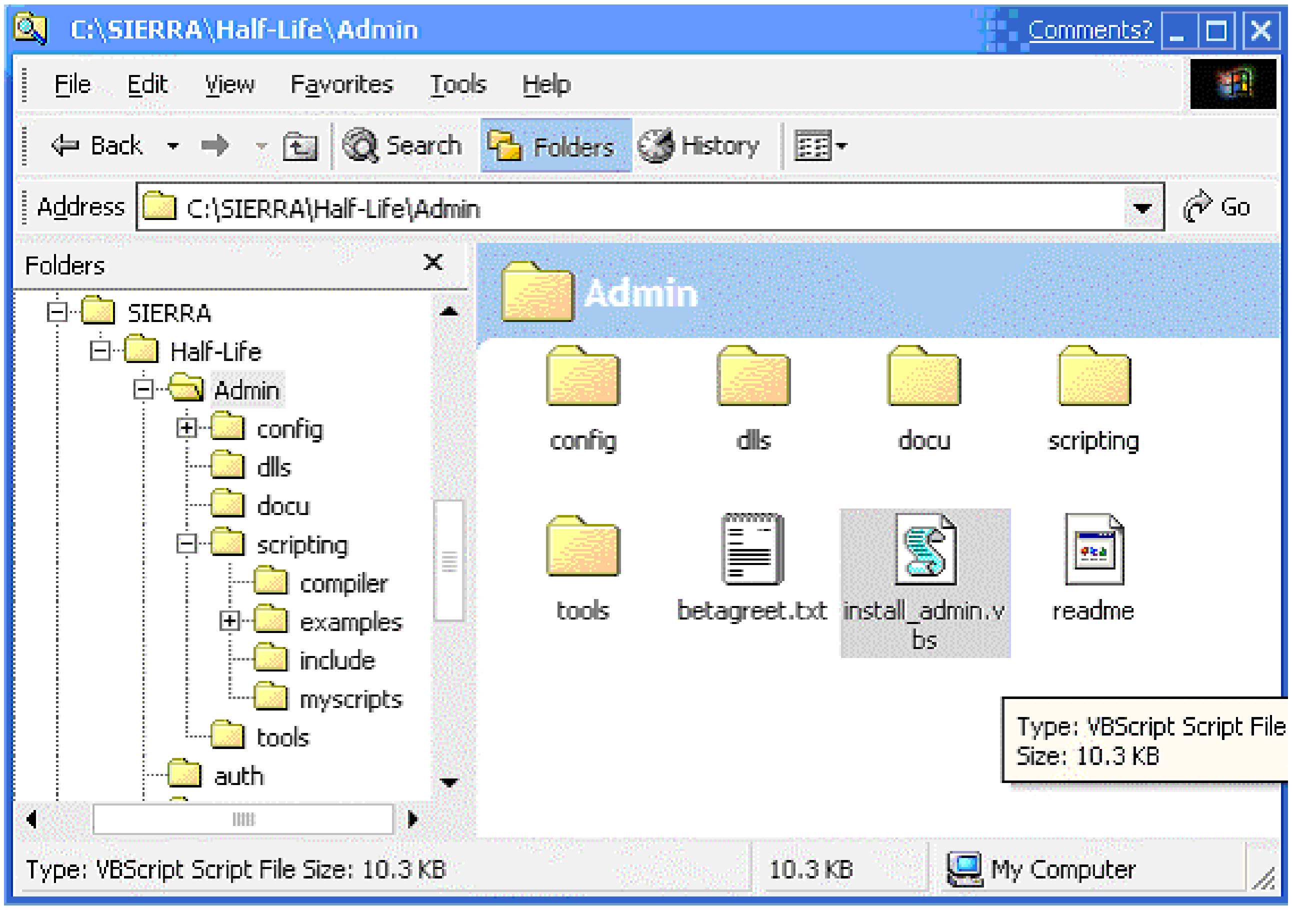
Okay. Hopefully you get the picture. Come back here after you have completed your backup.

Next: Unzipping Admin Mod the Right Way

Unzipping Admin Mod the Right Way

When you download the Admin Mod ZIP file, you need to **be sure to unzip it and use the folder information in the ZIP file** – This means that when you unzip the file, it will create a folder called Admin and several subfolders inside the Admin folder, as shown in the figure below.

You unzip the Admin Mod distribution to the SAME FOLDER that contains HLDS.EXE or HL.EXE. This directory is also the directory in which your game mods are installed, so you will see a subdirectory for each of your mods here (in the case of a Linux server, we are talking about the same directory as the hlds_l .so runtime). Typically this means that you unzip it into the c:\SIERRA\Half-Life\ directory or similar. *DO NOT unzip the file into your game mod directory! If you do, you will have problems.*



Address: C:\SIERRA\Half-Life\Admin

Type: VBScript Script File
Size: 10.3 KB

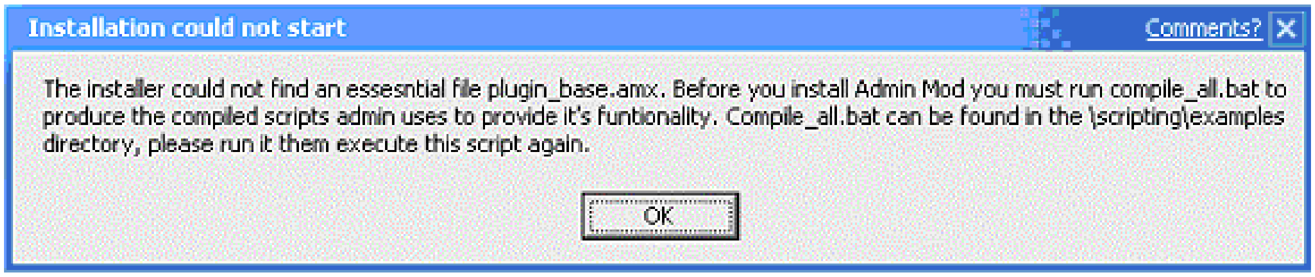
Type: VBScript Script File Size: 10.3 KB | 10.3 KB | My Computer

If you don't see these folders/directories after unzipping the file, you have done something wrong, so **STOP NOW** and save yourself the headache. You most likely did not use the folder names included in the ZIP file.

Running the Windows Installation Script

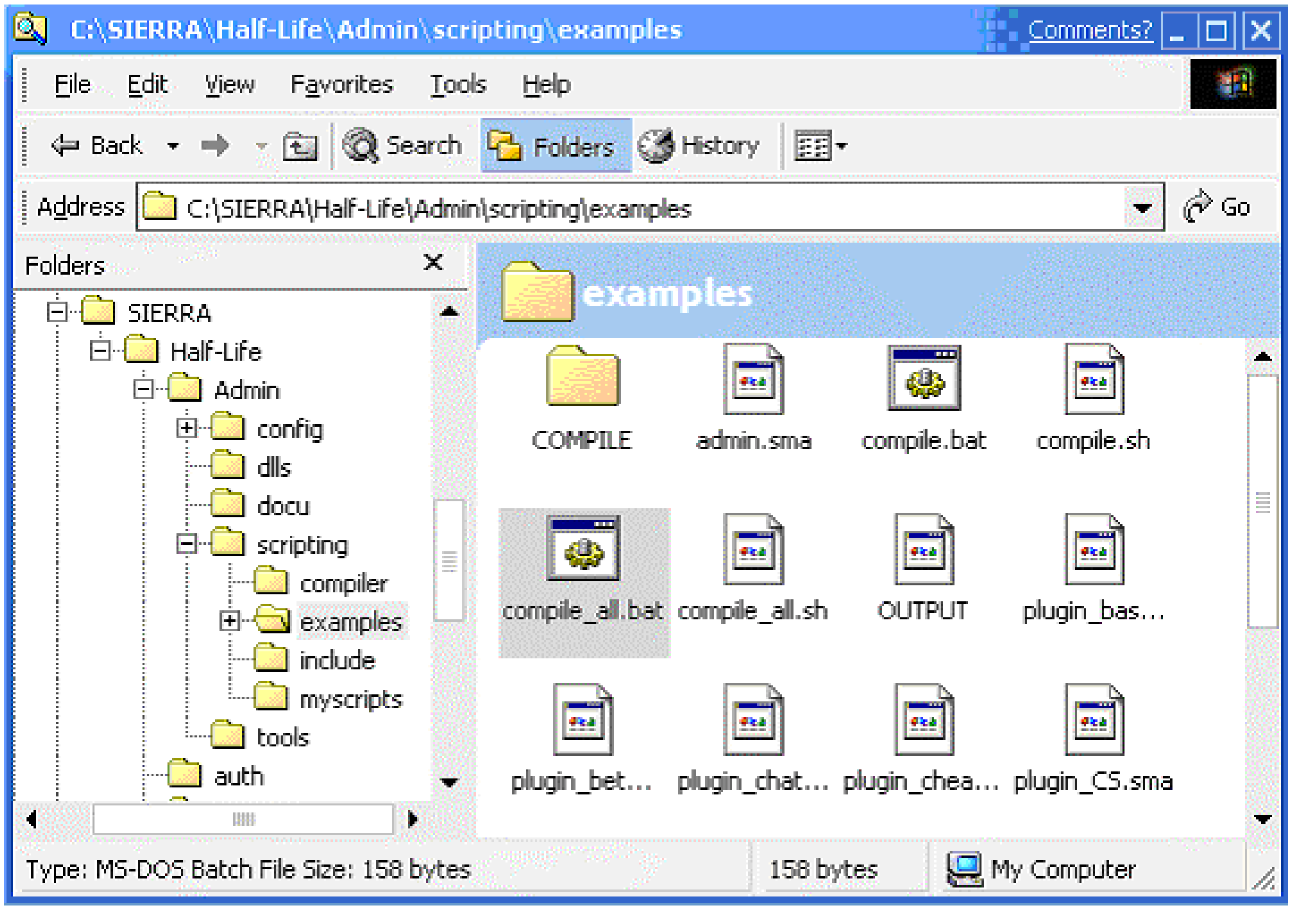
> ***NOTE:*** *If you experience difficulties running the installation script that is supplied with Admin Mod , it is possible you either have an outdated version of the Windows Scripting Host or you do not have it installed on your computer. In either case, the latest version of Windows Scripting can be downloaded from:*
>
> *http://msdn.microsoft.com/scripting/*
>
> *It is also possible that your anti-virus program might warn you or prohibit you from running VBScript files such as this installer. If that is the case, you will need to disable your anti-virus software while installing Admin Mod, but be sure to re-enable it after you are done.*

To install Admin Mod, you will run a friendly little VBScript that will greatly simplify the process of installing the Admin Mod. If you are used to the script that was included with version of Admin Mod 2.50, we thing you will be pleasantly surprised. The scripts come pre-compiled for you and in the correct location, just so long as the files and folders were unzipped properly.

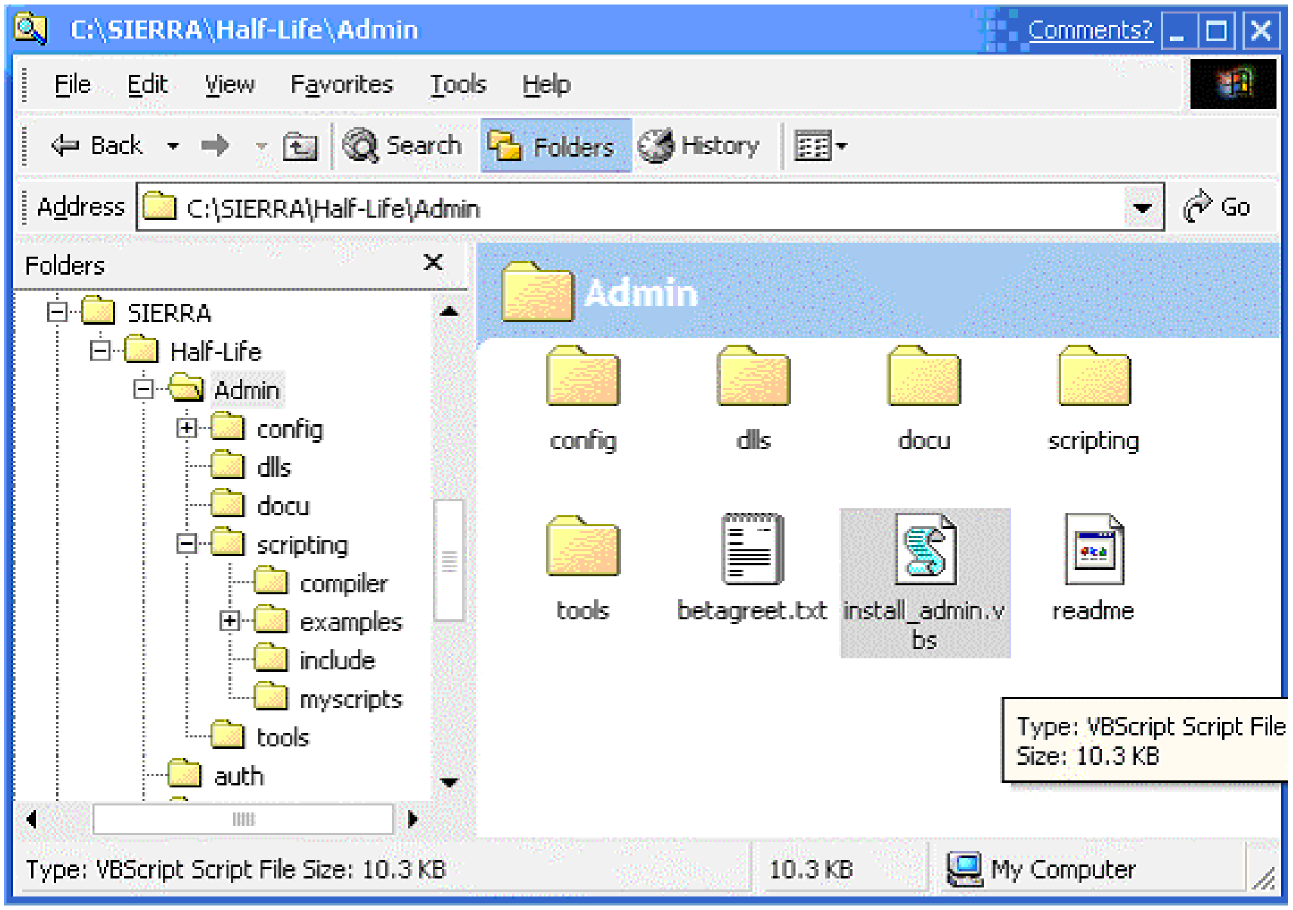


The installer could not find an essesntial file plugin_base.amx. Before you install Admin Mod you must run compile_all.bat to produce the compiled scripts admin uses to provide it's funtionality. Compile_all.bat can be found in the \scripting\examples directory, please run it them execute this script again.

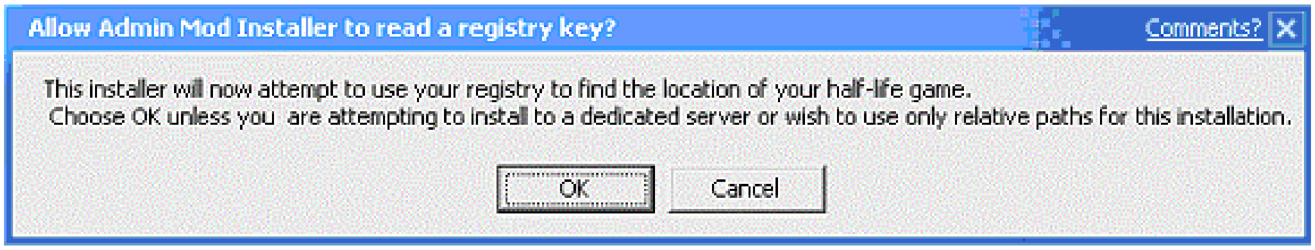
If you do not have the script files compiled ahead of time, or if they are not in the appropriate folder (\Admin\scripting\examples), the above window will pop-up. As explained in the window, you simply need to run the file named compile_all.bat in the \Admin\scripting\examples folder. Doing this will - of course - compile all of the scripts so they are ready to use. The image below shows the location of that folder in the Windows Explorer view.



Once you have the scripts compiled, run the install_admin.vbs file located in the \Admin folder. The image below shows the location of install_admin.vbs:



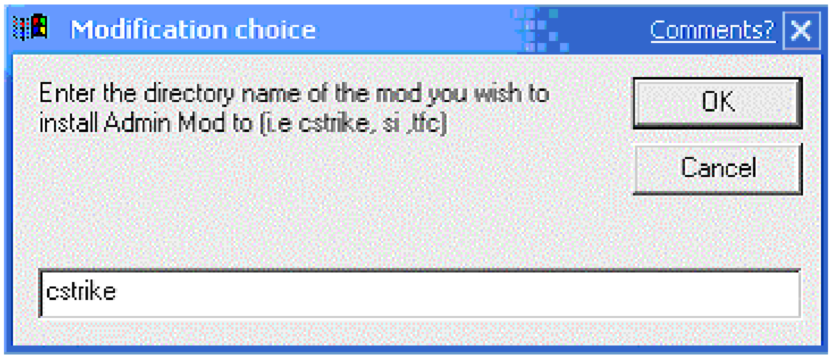
If you have a server that is set up without entering a cd key (the stand-alone dedicated server download), select cancel for the following pop-up. If you have entered a cd key (you are using the server that came on your Half-Life or retail version of Counter-Strike CD, even if it has been upgraded), you have the choice between either option. If you are new at this and the folder structure on your computer looks pretty much like the ones in the above images, just press OK.



Allow Admin Mod Installer to read a registry key?

This installer will now attempt to use your registry to find the location of your half-life game.
Choose OK unless you are attempting to install to a dedicated server or wish to use only relative paths for this installation.

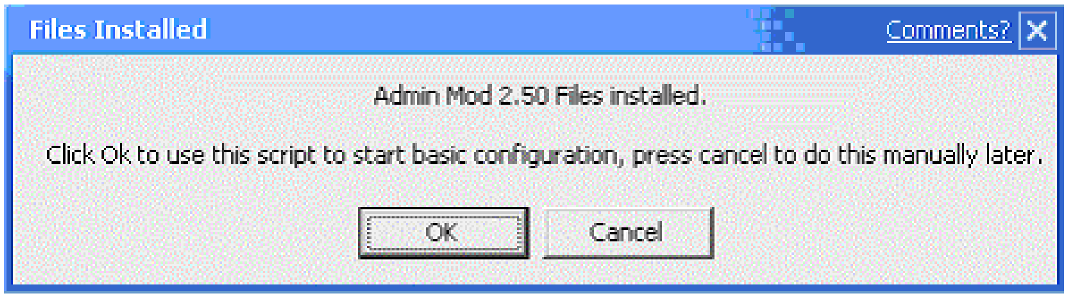
This window will pop-up if you selected cancel in the preceding window. If you selected OK, skip this step. This dialog asks for the *directory name* of the game mod you want to install to. The mod directory name depends on the mod you are hosting on your server. If your serve is a Counter-Strike server, enter "cstrike" and if it a TFC server, enter "tfc" in this box. Other game mods will require you to enter the name of the mod-directory for your game.



Enter the directory name of the mod you wish to install Admin Mod to (i.e cstrike, si ,tfc)

cstrike

The next one is pretty self-explanatory. Selecting OK will allow the installer to create your users.ini file and sample adminpass.cfg file, which is nice, since that way you will have a sure-fire matching set of files. These files are described in detail later in this manual, in the Configuration section.



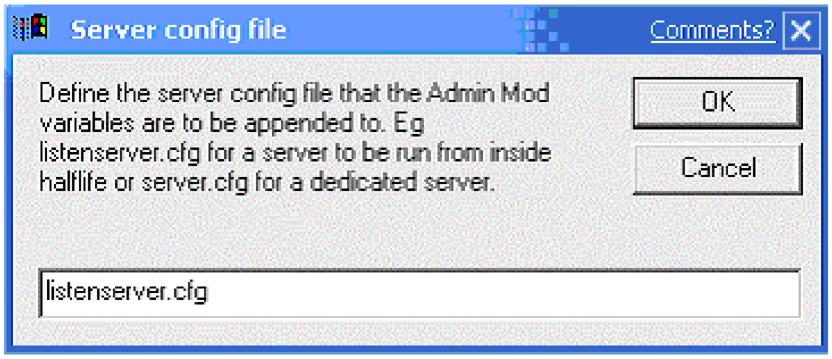
Admin Mod 2.50 Files installed.

Click Ok to use this script to start basic configuration, press cancel to do this manually later.

The next step, pictured below, will determine which server configuration you will use: Dedicated Server, or non-dedicated (also called a listen-server). Both options are described and pictured here.

Choose the one that fits your desired server configuration from the two options below.

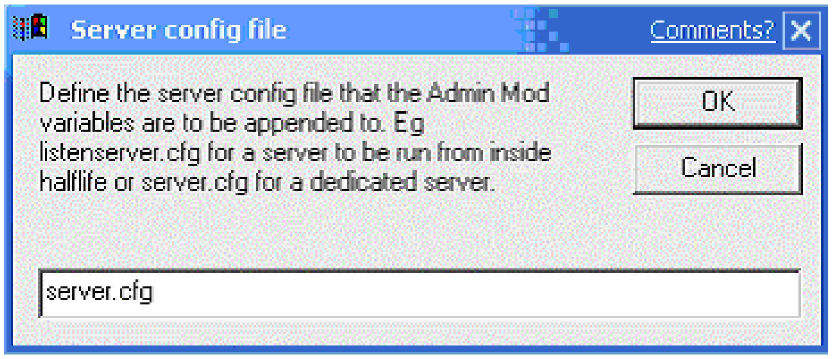
If you create a *non-dedicated* server by going through the Half-Life GUI and selecting “Create” *and* do not check the dedicated server box, you need to enter "listenserver.cfg" in the box like this:


Server config file

Define the server config file that the Admin Mod variables are to be appended to. Eg listenserver.cfg for a server to be run from inside halflife or server.cfg for a dedicated server.

listenserver.cfg

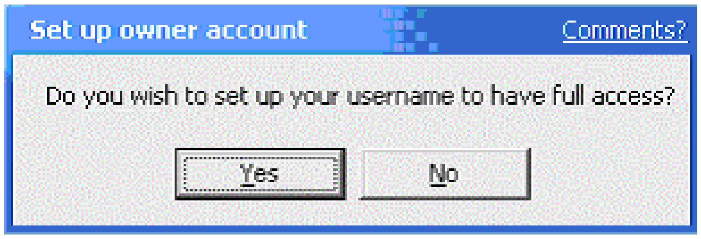
However, if you use a *dedicated server* (a DOS-like window opens and lots of data scrolls down when you start it), enter " server.cfg in the box pictured below:


Server config file

Define the server config file that the Admin Mod variables are to be appended to. Eg listenserver.cfg for a server to be run from inside halflife or server.cfg for a dedicated server.

server.cfg

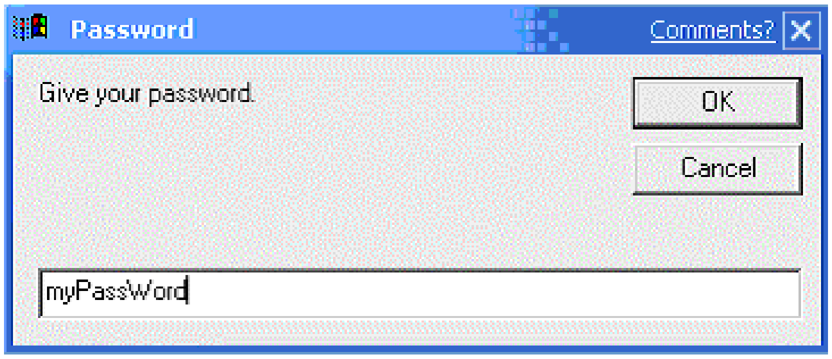
Depending on your setup, you may or may not be asked for the following information:


Set up owner account

Do you wish to set up your username to have full access?

If you click "Yes," you will see the following - It creates an **adminpass.cfg** file in order to set up your password automatically if you play on this computer:

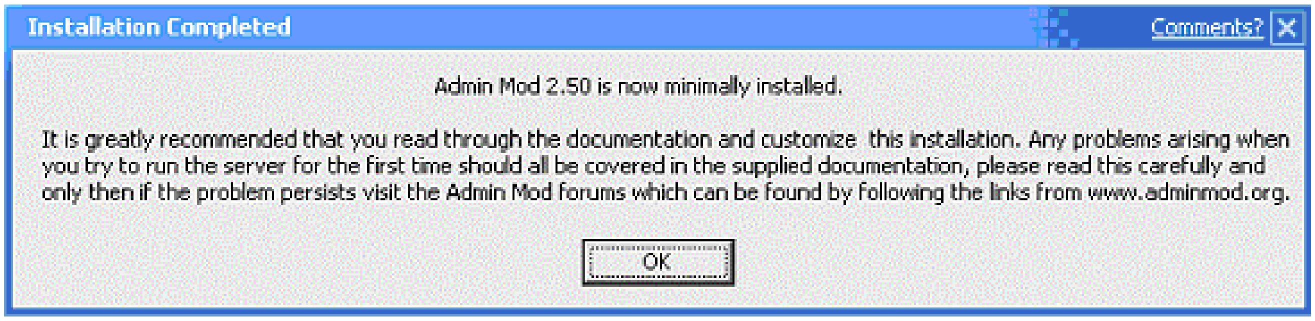
NOTE: If you are setting up a dedicated server, this file will need to either be moved or re-created on the computer where you play the game. This file is created on your server during installation, but it is not meant to stay there, unless you are running a listen-server. Be sure to see the section about Client configuration via setinfo commands in this manual for more information.



Give your password.

myPassWord

When the installation script is done, you will see the following. Be sure to read this, as the information displayed is very important:



Admin Mod 2.50 is now minimally installed.

It is greatly recommended that you read through the documentation and customize this installation. Any problems arising when you try to run the server for the first time should all be covered in the supplied documentation, please read this carefully and only then if the problem persists visit the Admin Mod forums which can be found by following the links from www.adminmod.org.

So what's Next???

You're not done yet! You will need to set up and configure a number of server files. You're a good made a good start, but the remaining tasks are critical. So, what are you waiting for - Let's get started!

Note that not every problem can be resolved in this documentation, but we suggest *strongly* that you look here first. If you have problems you can't solve after reading this manual - and that does happen - please ask your question in the forums at http://www.adminmod.org. Please do not email your questions to the team directly, as we cannot guarantee we will be able to respond to individual emails in a timely fashion.

Next: Setting up your .cfg and the *.ini files

Manual Installation - Walk-Through

or, Snaller's Heavily Adapted Step-by-step Guide To Installing Admin Mod manually

> **Note:** This portion of the guide discusses a manual installation of the Admin Mod, and is not recommended for people who do not have a clear understanding of directory structures, file editing, etc. – For a script-based, more automated installation, please read the Running the Windows Installation Script section. However, no matter what type of installation you are undertaking, there is a lot of good information to learn from here, so be sure to read it at some point in time, even if you do script-based installation.

What was done in the process of describing this installation was for CS under WinNT, but there isn't any great difference for TFC and Linux (if that's what you are using), for example.

The Counter-Strike mod directory is called *cstrike*, and everything takes place in there (Your mod directory might be called something else, like *tfc* if you are running a Team Fortress Classic server, etc.).

We are going to assume you have the latest version of the Admin Mod (always get the latest).

First, stop your server. Unpack/unzip the file you downloaded, being sure to allow the directory structure in the ZIP file to be recreated when the files are unzipped (If you are using WinZip, you should make sure the box labeled "Use folder names" is checked before you extract the files from the zip archive).

Inside the files and directory structure you just unpacked (by default the top level directory will be called “Admin”), there is a dlls directory. Copy everything from that directory to the dlls directory inside your MOD directory:

- If you are running Counter-Strike, you should copy everything in the *half-life\Admin\dlls* directory to the *half-life\cstrike\dlls* directory.
- If you are running TFC, you should copy everything in the *half-life\Admin\dlls* directory to the *half-life\TFC\dlls* directory.
- Other mods, same procedure, just different directory names …

This shouldn't overwrite anything (unless you have had the mod installed before, in which case if you followed the first instruction in this manual, you are okay, because you've backed up your files). And the fact that these are new dll 's doesn't upset the server at all (it doesn't even know they are there yet).

> *If for some reason the directory structure that was created when you unzipped the Admin Mod distribution archive does not have subdirectories in it, or if things just don't seem to match up, delete the Admin directory and*

all its contents, and unzip again, being sure to preserve the directory information as described above while unzipping.

Now, into in your mod directory you *copy the content only* of each of the sample configuration files you find in the directories called "*Admin/config*" and *Admin/config/samples*" into the files of the same name in your mod directory:

Note that we are not saying to copy the files directly to the mod directory, because you might overwrite a file of the same name when what you need to do is take the content from the sample file and add it to your existing file.

In other cases, you will find that some of the files do not exist yet in your mod directory. In that case, just copy the who file over and save some effort.

Now you need to edit some of the configuration files. Start by editing your main server.cfg file (this should be in your mod directory). Note that if you are running a listen server, you need to work with the listenserver.cfg file, so anywhere we mention server.cfg here, just do the same thing to the listenserver.cfg on your computer.

Suggestion: Scroll to the bottom of the server.cfg file and make a separating comment line, so you know where the Admin Mod stuff begins. Then copy all the text from the server.cfg file in the samples directory in to the one in your mod directory (and then we won't use the sample server.cfg in the examples directory anymore, only the one in your mod directory).

Now - check the file size of your server.cfg file and make sure it is under 16 bytes. It takes a lot of junk in the server.cfg to make it 16 bytes or larger, but rest assured that if it is too big, HLDS won't use it when it runs, and that would be bad. On the off chance your file is over the limit, start deleting comments from the file to free up some space.

Now you need to edit the file paths listed in the server.cfg as shown below so they point correctly to the other config files (assuming you want to use those files - if you are not going to use any of these files, you can set the cvar in server.cfg to 0 [zero]):

```
users_file " users.ini "
models_file " models.ini "
maps_file " maps.ini "
words_file " wordlist.txt "
```

Or, alternatively, to show you do not want to use a particular file (in this case you are not going to use a word filter file):

```
words_file 0
```

It seems that under Windows NT at least, you need to specify them all or the mod may crash. If all else fails, specify them all. Can't hurt.

Now you need to know what kind of scripting you are going to use. Probably you are going to use the modular plugin-style scripts that are provided as the default scripting style with Admin Mod 2.50 -- In that case, you need to copy all of the files than end with an .amx extension from *admin\scripting\binaries* to the *[mod_name]\dlls* directory.

However, if you have a single old-skool script file that you want to use, you'll get a chance to deal with that now... Put a copy of it in the *mod/dlls* directory now. Just keep in mind that the officially-support scripting model going forward will be the modular-style scripts mentioned above, so it's a good idea to make that change either now or in the near future. Most all the functions from the older style scripts developed and distributed via the Admin Mod forums (and some new features and functionality as well) are available in the default plugin scripts. Plus, as people develop new plugins, all you have to do is drop the plugin script into place and you are done - no more recompiling everything to add new features!

Okay. Here comes the dinner menu part. Please choose ONE of the selections below and follow the instructions given. You options are "Old Skool" and "Plug-In"

> **Plug-In (supported model)** -- IF YOU ARE USING THE PLUGIN STYLE SCRIPTS (recommended), you need to be sure the admin_plugin_file cvar in your server.cfg file is pointing to "plugin.ini"
>
> **Old Skool (unsupported for development in the future)** -- ONLY IF YOU ARE USING A SINGLE OLD_STYLE SCRIPT, you need to locate the cvar field called "script_file" (still in server.cfg) and make sure it's pointing to the right file, usually called admin.amx

Now save the server.cfg file. There are a bunch of other settings that you can (and probably should) configure for your server. For more information about doing that, please see the section called "Setting up your server.cfg file" in this documentation. Nothing bad has happened to your server yet: If you restart your server it will still be the old faithful, it won't know or care about the extra things you have written to the server.cfg file.

Now you need to edit the other configuration files that you have copied into your mod directory.

The wordlist.txt file is a list of terrible words, plain text with one bad word per line. This is your censor list, if you want one. If you add a word here, people can't use that word when they chat. If you don't want to censor anything you can set the "words_file" cvar in the server.cfg to a value of "0" (zero). To enable it, specify the filename in the words_file cvar in your server.cfg.

The ips.ini file is used if you want to give players reserved spots based on their IP address, and examples are provided in the file. If you don't want anything to be reserved, just delete all the lines, or set the cvar in server.cfg to "0"

The maps.ini file contains a list of maps that the players can vote for, one map per line, in the exact same format as the mapcycle.txt file that your mod uses (of course Counter-Strike has built-in voting, but this enhances voting greatly).

The users.ini file is where you put the nicknames, passwords and security level of those people to whom you want to give some kind of admin rights. You should at least put yourself in here and give your username maximum rights. Note that permission levels are slightly different in version 2.50 and later than they were in previous versions, and that there is no longer a file used to reserve player nicknames, because this is now done using a security level.. See "Setting up your users.ini file" in this manual for more information on setting up this file, its format and access levels.

Your plugin scripts are listed in the plugin.ini file -- Be sure to uncomment (by removing the leading semicolon ";") the TFC or CS lines if you are running one of those mods, otherwise the commands and features from those scripts will not be available to you! You can also add new compiled plugins, many of which are available at the AScript web site, to your mod's dll directory, and then specify them in the plugin.ini in order to activate them on your server.

> *Hint: if you ever want to know what scripts are running on your server at any given point in time, run an admin_version command. The resulting output will show all the scripts currently active on your server. If the information is so long it scrolls in your game console, try using "admin_command admin_version" in the dedicated server console.*

The metamod.ini file is a simple file that specifies the DLL (dynamic link library) file to run when using meta mod in conjunction with Admin Mod. By default, it already has the settings you need to run Admin Mod preconfigured in it.

How this all works: Starting with Admin Mod version 2.50, here is what happens:

- HLDS executes metamod.dll
- Metamod in turn executes admin_mm.dll.

By default, this file's contents are preconfigured to point to the correct Admin Mod DLL files necessary for running Admin Mod. You may wish to see an example of how other mods, like bots, can be used in this way. See "How to Use Bots with

MetaMod (AM 2.50)" for more information about Metamod and running multiple HLDS modifications at the same time.

> *In the next step we will edit another file (liblist.gam), the one which will call the Metamod DLL file. Once Metamod is running, it will look for the last file we discussed (metamod.ini) to determine which other DLLs should be loaded. This is where Admin Mods DLL files are specified.*

Admin Mod is almost ready, but not quite. One last thing is still missing:

In your mod directory there is a file called liblist.gam - make a backup of this now (in other words put a copy of it in a safe place you won't forget), so that if you ever want to remove the Admin Mod, all you have to do is put your backup file back in place of the one you are about to edit.

If you have not stopped your dedicated server yet, do so now.

Open the liblist.gam file and find the line starting with *gamedll_linux* or *gamedll* (depending on your OS) and replace the text with either "*dlls/metamod_i386.so*" or "*dlls\metamod.dll*" respectively. (i.e. for Windows it's *gamedll "dlls\metamod.dll"*)

Now when you restart your server, it will start with the Admin Mod running. (When we say restart the server, we mean *completely* stopping it - quit in the console and then restart it from cold state).

Prior to connecting, you will need to type setinfo *pw-field your_password* in the console – where "*pw-field*" is the field defined in your server.cfg as the password field and "*your_password*" should be replaced with your real password. This has to be entered prior to each server login attempt, as it is erased from your profile upon successful login.

> *Note: You can get more information about setting up and automating entry of the "setinfo" commands for the client (game) setup in the section entitled "Required Client Configuration Instructions" elsewhere in this documentation.*

You can then join the server and type admin_help in the console and see a list of the commands available to you through the Admin Mod!

Thanks to Snaller for providing the huge majority of this tutorial, which was adapted heavily for this version of the Admin Mod.

Linux Installation Information

Linux people will run the install_admin shell script to perform an installation of Admin Mod . Alternatively you can install manually, but the shell script does a good job of getting the basic in place.

Linked below is the output of the install script sessions for your reference.

- 1st Time Linux Install
- Linux Upgrade Install

Information is also available about creating Linux Passwords

Following installation of the Admin Mod, proceed to The Admin Mod Configuration Files section to begin setting your configuration variables. Once you have your files in place, configuration of Admin Mod is essentially the same for Linux and Windows.

1st Time Linux Install output

Below is the output of the Linux install_admin shell script when running it on a machine that does not have Admin Mod installed on it yet: This content is provided for reference. We will assume that if you have successfully installed hlds_l on a Linux machine, you don't need to have your hand held.

```
    _______________________________________________________________
               Admin Mod Installation Script
                       Version 2.50
   This script will install Admin Mod version 2.50 on your
   system. It will ask you for the directory that you installed
   the Half-Life server in. To accept defaults just press Return.
   After the binaries are installed you should read the docu-
mentation and configure Admin Mod before you use it for the
   first time.
Shall I proceed with the installation? (y/n) [y]

Please enter the directory where your HL server is installed.
[/usr/local/hlds_l]:


Installing binaries and config files ...

Editing your liblist.gam file ...

   The Admin Mod files have now been installed. It looks like you
   install this version of Admin Mod for the first time. You will
   have to edit your server.cfg

 file in the directory /usr/local/hlds_l/tfc

.
   I can now add the necessary lines to your server.cfg file.
   Edit the file afterwards to configure it to your needs.

Shall I add the Admin Mod lines to your server.cfg file? (y/n) [y]

   Congratulations, Admin Mod is now installed.
   To configure it you should now edit your server.cfg file and
   create users.ini, maps.ini, ips.ini, models.ini and wordlist files as
needed.
   Follow the instructions in the documentation and check for more
infor-   mation at the Admin Mod web site http://www.adminmod.org.

   Enjoy!
```

Linux Upgrade Install

Below is the output of the Linux install_admin shell script when running it on a machine that has a previous version of Admin Mod installed on it: This content is provided for reference. We will assume that if you have successfully installed hlds_l on a Linux machine, you don't need to have your hand held.

```
Adminmod Installation Script
                       Version 2.50

   This script will install Adminmod version 2.50 on your
   system. It will ask you for the directory that you installed
   the Half-Life server in. To accept defaults just press Return.
   After the binaries are installed you should read the docu-
   mentation and configure Adminmod before you use it for the
   first time.

Shall I proceed with the installation? (y/n) [y]

Please enter the directory where your HL server is installed.
[/usr/local/hlds_l]:


Installing binaries and config files ...

Editing your liblist.gam file ...

   The Adminmod files have now been installed. It looks like you
   had installed a version of Adminmod before. You will still have
   to edit your server.cfg

 file in the directory /usr/local/hlds_l/tfc

.
   If you want to, I can now append the necessary lines to your
   server.cfg file. But since you have a previous installation you
   may want to do that by hand. Edit the file in any case because some
   options may have changed since the last version.

Shall I add the Adminmod lines to your server.cfg file? (y/n) [n]

   Conratulations, Adminmod is now installed.
   To configure it you should now edit your server.cfg file and
   create users.ini, maps.ini, ips.ini, models.ini and wordlist files as
needed.
   Follow the instructions in the documentation and check for more
infor-
   mation at the Adminmod website http://www.adminmod.org.

   Enjoy!
```

Linux Encrypted Passwords

LINUX USERS ONLY:

Password encryption

For security reasons, administrators are encouraged to encrypt their passwords. This is done by setting the cvar encrypt_password to 1 in the server configuration files (listen/ server.cfg) To encrypt a password, go to the Admin/tools directory, and type:

```
./make_pass password
```

This will output a cut'n'paste-able encrypted version of the password you entered. Now, copy'n'paste the ENCRYPTED version of your password into the users.ini file in the password area (between the two colons ":"). The un-encrypted version is to be placed in the second password area of the setinfo line. Remember that throughout this process, all the passwords, encrypted or not, are case sensitive.

Uninstalling Admin Mod

How do I uninstall Admin Mod ?

You don’t.   **:D**

Just kidding...

Hopefully you were a good scout and made a backup of your original files just in case. If so, just restore your old liblist.gam file and all is done.

But if you didn't (shame on you), It's really very simple. To keep things clean and tidy, you should remove the Admin Mod specific CVARs from your server.cfg or listenserver.cfg (whichever one applies to your situation).

Then edit your liblist.gam file in your server's mod directory to return the gamedll file setting back to "mp.dll " (Windows) or "dlls/cs_i386.dll" (in the case of Linux running under CS)

That's it.

Setting up Admin Mod with MySQL

To run Admin Mod in combination with a MySQL database you have to setup the database and then configure Admin Mod to use it. The information that Admin Mod can retrieve from a database is the information normally kept in the files users.ini, models.ini, ips.ini, wordlist.txt and plugins.ini. You can still use the files with the MySQL version of Admin Mod. You can mix file and database operation but the database tables, if specified, will take precedence. Data is read only from tables or files, not both. If Admin Mod fails to read data from the specified MySQL table it will fall back to using files. Please read the manual first to understand how to set up Admin Mod in general and what the configuration files are about and their contents.

MySQL database setup
===================

On your MySQL server you will have to set up a database to store the data for Admin Mod to use. What name you give to the database and what tables it contains is up to you. You can also have more fields in tables than those described below. This document only describes the maximum set of tables and columns necessary for a complete Admin Mod setup. You can leave out tables which you don't need or add other tables unrelated to Admin Mod. The names for the tables are also free. The only thing which you cannot choose freely are the column names. But we get to that later.

The database

Let's start with the database. For this example we choose to call it adminmod. You can choose another name. If you haven't created a database, yet, do so now with the CREATE DATABASE command. Refer to the MySQL manual for any details concerning MySQL commands and syntax.

```
mysql> CREATE DATABASE adminmod;
```

The users table

Next we create the table holding the admin users. We choose to call it users. You can give it another name. The users table has three columns: nick, pass and access. These columns correspond to the three fields for entries in the file specifies by the users_file cvar. nick is the nickname of the user, pass is his password and access his access level.

```
mysql> CREATE TABLE users( nick VARCHAR(30) PRIMARY KEY NOT NULL,
   --> pass VARCHAR(20), access INTEGER UNSIGED );
mysql> DESCRIBE users;
+--------+------------------+------+-----+---------+-------+
| Field  | Type             | Null | Key | Default | Extra |
+--------+------------------+------+-----+---------+-------+
| nick   | varchar(30)      |      | PRI |         |       |
| pass   | varchar(20)      | YES  |     | NULL    |       |
| access | int(10) unsigned | YES  |     | NULL    |       |
+--------+------------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

The models table

The models table has two columns: nick and pass. These two columns correspond to the fields in an entry of the file specified by the models_file cvar. nick is the name of the model and pass is the assigned password. We choose to call it models. You can choose a different name.

```
mysql> CREATE TABLE models( nick VARCHAR(20) PRIMARY KEY NOT NULL,
   --> pass VARCHAR(20) );
mysql> DESCRIBE models;
+--------+------------------+------+-----+---------+-------+
| Field  | Type             | Null | Key | Default | Extra |
+--------+------------------+------+-----+--------+-------+
| nick   | varchar(20)      |      | PRI |        |       |
| pass   | varchar(20)      | YES  |     | NULL   |       |
+--------+------------------+------+-----+--------+-------+
2 rows in set (0.00 sec)
```

The ips table

The table to hold the IPs corresponds to the file specified by the ips_file cvar and has only one column called ip. We choose to name the table ips. You can give it a different name.

```
mysql> CREATE TABLE ips( ip VARCHAR(15) );
mysql> DESCRIBE ips;
+-------+------------+------+-----+---------+-------+
| Field | Type       | Null | Key | Default | Extra |
+-------+------------+------+-----+---------+-------+
| ip    | varchar(15) |      |     |         |       |
+-------+------------+------+-----+---------+-------+
1 row in set (0.01 sec)
```

The words table

The words table holds the words to be filtered from player chat. It corresponds to the file specified by the words_file cvar and has only one column called word. We choose to name the table words. You can give it another name.

```
mysql> CREATE TABLE words ( word VARCHAR(30) );
mysql> DESCRIBE words;
+---------+------------+------+-----+---------+-------+
| Field   | Type       | Null | Key | Default | Extra |
+---------+------------+------+-----+---------+-------+
| word    | varchar(30) |      |     |         |       |
+---------+------------+------+-----+---------+-------+
1 row in set (0.01 sec)
```

The plugins table

This table lists the plugins to be used on the server. It has only one column called plugin. It corresponds to the file specified by the admin_plugins_file cvar. We choose to call it plugins. You can give it another name.

```
mysql> CREATE TABLE plugins( plugin VARCHAR(30) );
mysql> DESCRIBE plugins;
+----------+------------+------+-----+---------+-------+
| Field    | Type       | Null | Key | Default | Extra |
+----------+------------+------+-----+---------+-------+
| plugin   | varchar(30) |      |     |         |       |
+----------+------------+------+-----+---------+-------+
1 row in set (0.01 sec)
```

Admin Mod variables setup
=========================
To use Admin Mod with the database that you just set up you have to tell it how to connect to the MySQL server and what database and tables to use. Admin Mod provides you with a set of cvars to do so. You set these cvars in the server.cfg file just like all the other cvars.

Connecting to the MySQL server

To connect to the server you need to specify the address of the server, the user to access it as and the password to use.

mysql_host This variable specifies the address of the machine that the MySQL
server is running on. Examples:

```
mysql_host "192.168.2.1"
mysql_host "localhost"
mysql_host "some.machine.net"
```

mysql_user This variable specifies the username under which Admin Mod

```
should
connect to the MySQL server. Examples:
mysql_user "adminmod"
mysql_user "someclient"
mysql_pass This variable specifies the password of the user to connect
to the
MySQL server. Examples:
mysql_pass "somepass"


Retrieving data from tables
---------------------------
The following variables are used to specify which database and which
table to use. The format is "database.table". The list should be
self-explanatory. The examples use the database and table names which
were used in the examples above.
The table for admin users:
mysql_dbtable_users E.g.:
mysql_dbtable_users "adminmod.users"
The table for models:
mysql_dbtable_models E.g.:
mysql_dbtable_models "adminmod.models"
The table for IPs:
mysql_dbtable_ips E.g.:
mysql_dbtable_ips "adminmod.ips"
The table for censored words:
mysql_dbtable_words E.g.:
mysql_dbtable_words "adminmod.words"
The table for plugins:
mysql_dbtable_plugins E.g.:
mysql_dbtable_plugins "adminmod.plugins"
You don't have to use all available tables. As an example you can
choose to only use the users and plugins tables from the database. The
reserved models will be read from a local file and you don't want to
use the IPs. That would result in the following setup:
modles_file "models.ini"
ips_file 0
mysql_dbtable_users "adminmod.users"
mysql_dbtable_plugins "adminmod.plugins"
It is a good idea to have users.ini and plugins.ini files, too. That
way Admin Mod can fall back to those files if it is unable to read
from the MySQL tables.
```

Configuration

The Admin Mod Configuration Files

If you have run a Half-Life dedicated server for more than a minute or two already, then you know that the server uses configuration files (usually with a file extension of .cfg) to set many optional configuration variables that the server reads when it is started. Admin Mod has a similar set of files, which determine how the Admin Mod is configured and behaves. Most of Admin Mod's configuration files have an extension of ".ini" - although some of Admin Mod's configuration options are set in the Half-Life server config files.

See also: Setting up server.cfg and the *.ini files

Here’s a quick description of what each of the specific Admin Mod .ini files are used for:

users.ini
The users.ini file is used to specify who will have special rights on your server (by name or WON ID) and what commands they have permissions to execute. **See: Setting up your users.ini file**

ips.ini
The ips.ini file is used for slot reservation on your server based on IP address. All you have to do is specify the IP addresses of the computers that can use one of your reserved slots. **See: Setting up your ips.ini file**

models.ini
The models.ini file is used for model reservation. It prevents players from using the specified models unless they set the correct password. It lets you protect any given model from use by unauthorized players. **See: Setting up your models.ini file**

maps.ini
The maps.ini file is used to control which maps users can vote for. It has the same format at the mapcycle.txt file. Note that it does not replace the mapcycle.txt file (which controls the maps that run on your server and the order they are played in). Instead, it is just a list of maps available to be started via an Admin Mod vote. **See: Setting up your maps.ini file**

plugin .ini
New to version 2.50, this file is used to denote the compiled script files that are to be used on your server. The scripts that come with Admin Mod 2.50 are organized and split into chunks by functionality and by game type. So, to add a custom script or disable certain parts of the functionality, one has the option with this file to do so without having to recompile every piece of functionality, every time you make a change. Want to test a new script? Just compile it, dump it into the mod's dlls directory, add the script name to the plugin.ini file and reload – presto, you can test away, and you didn’t have to modify (and possibly break) your already existing

scripts.

See: Setting up your plugin.ini file

metamod .ini

This is a simple file that specifies the DLL (dynamic link library) file to run when using meta mod in conjunction with Admin Mod. Starting with Admin Mod version 2.50, here is what happens:- HLDS executes metamod.dll - Metamod in turn executes admin_mm.dll. If that’s confusing to you or makes absolutely no send, don’t worry – it’s more useful for people upgrading from an older version of Admin Mod, and hopefully they will know what we mean. If they don’t, they should read the fast-track/manual upgrade documentation.

The metamod.ini file comes pre-configured and is very simple. You should not need to change it, but if you do, we will assume you know how it works and what you are doing to an extent where you will not need such an explanation.

Note that Meta Mod is frequently updated. Be sure you are always running the latest version by checking regularly at http://www.metamod.org for updates.

See: How to Use Bots with MetaMod (AM 2.50)

There is also a “**metagame.ini**” file that can be used, but it is usually not needed. It would be used, for instance, to start another mod program which otherwise would have been loaded in the liblist.gam file if Admin Mod was not installed on the computer (Bots are a good example of this). If you are using a non-supported mod or auto detection of your game mod fails, feel free to go to the Admin Mod Forums and ask for help.

Another file, named “**liblist.gam**,” is a plain text file on your server that the dedicated server program reads when started to determine what it should do and some of the files it should load and run. This file is not unique to Admin Mod -- all servers utilize this file to specify what base runtime options and resources the server should use. Admin Mod requires that this file be changed in order to allow the server to start Admin Mod when the server itself is fired up. The specifics of this file and it's setup for Admin Mod are discussed later in a section covering manual installation. If you are doing the standard, automated installation, this file is configured for you.

Setting up server.cfg and the .ini files

Now is the time to take a quick break if you need one. Clear your head. The next steps are not complicated, but they will make you think.

These file determine how Admin Mod interacts with your server and the users of the Admin Mod. You must set up each of these files as described. A section entitled The Admin Mod Configuration Files contains descriptive information about each file. Please choose a configuration file below to get more information about configuration of that particular file.

- **Setting up your server.cfg file** (required)
- **Setting up your users.ini file** (required if you want to set up admin user accounts on the server)
- **Setting up your plugin.ini file** (required unless you are using the old-skool scripting style)
- **Setting up your ips.ini file**
- **Setting up your maps.ini file**
- **Setting up your models.ini file**

It's time to look back to the Admin\config and Admin\config\samples directories. In those folders you will find all of the .ini files that we need to work with, as well as a sample server.cfg file and the wordslist.txt file.

Setting up your server.cfg file

> **NOTE:** The server.cfg / listenserver.cfg file is the configuration link between Half-life and Admin Mod . Without a proper setup, Admin Mod will cause more problems than it will solve. Each portion of the setup is critical, so read through this section carefully. Many configuration variables (or cvars for short - we uses the term cvars in this document extensively, so now you know what we mean) have been removed or changed with Admin Mod 2.50, so everyone should read this section, confirmed user or not.

All of the Admin Mod configuration variables (cvars) are listed here. You must place all of these inside your server.cfg or listenserver.cfg file AMONG all the other Half-life cvars. A complete list of Admin Mod cvars is available in the reference chapter, under Admin Mod Configuration Variables (CVARs).

Please note that a 'dedicated' server will, unless told otherwise, read it's initial settings from 'server.cfg'. A 'listen' server, on the other hand, reads them from 'listenserver.cfg'. *You need to be sure you are editing the correct file before your start.* You will only use one or the other of these files, depending on your setup. Since it's so important, here is the simple explanation for all the visual people out there:

Type of Server	**File to Configure**
Dedicated Server (runs in a text window on the computer - you don't play the game on this machine, you just connect to it)	**server.cfg**
Listen Server (you start the server in your Half-Life game, and you can play on the same machine while it is acting as a server for other players)	**listenserver.cfg**

Setting variables in server.cfg when you're running a listen server, and vice versa, does nothing. A listen server is one that you start from within Half-Life itself: it starts when you join, it ends when you quit, and you have 0 ping. A dedicated server is started from the HLDS program, and appears on the server machine as just lots of text information. If you have no idea what type of server you are running, even after this explanation, simply configure your server.cfg file to you liking with all the necessary cvars and copy the contents of the server.cfg file to the listenserver.cfg file. This way, no matter what you do, you are covered.

Creating/Configuring Your server.cfg file:

Below is the contents of the default server.cfg file for Admin Mod 2.50 (Again, if you run a listen server, this applies equally to you, only your config file is called listenserver.cfg). Each entry is described in more detail here than you will find in the config file itself. Each cvar has been made to appear here in bold print, so you will be able to more easily find each one, and so you can differentiate the cvars from the comments.

// This is a comment

In the server.cfg file, any line beginning with " // " (comment characters) is ignored by the server when it loads this file. In addition, if the " // " characters appear anywhere on a line, anything after those characters is ignored by the server Any line *not* beginning with the comment characters is assumed to contain server instructions, and is parsed by the server for execution.

Here are the contents of the default server.cfg that comes with Admin Mod:

```
// Used by the TFC plugin only. If enabled, an attempt to switch
// teams will be rejected if it would unbalance the teams
// (0=disabled, 1=enabled)
admin_balance_teams 0
```

```
// If you are using bots with admin mod you can set this to 1
// It will protect bots from receiving client commands which
// would crash your server.
// (0=disabled, 1=enabled)
admin_bot_protection 0
```

```
// This is the message displayed to everyone after connecting.
admin_connect_msg "Welcome to the Real World..."
```

```
// This controls the availability of weapon restriction in CS.
// (0=disabled, 1=enabled)
admin_cs_restrict 0
```

```
// This will produce debugging messages in your logs which can
// be used to troubleshoot problems. Not recommended for general use.
// (0=disabled, 1=enabled)
admin_debug 0
```

```
// Determines whether or not the fun commands are allowed by default.
// If not on by default, the admin can still turn it on when wanted.
// (0=disabled, 1=enabled)
admin_fun_mode 0
```

```
// Enable to get special effects with certain commands
// like teleport or slap.
// (0=disabled, 1=enabled)
admin_fx 0
```

```
// If enabled, people who are gagged (not allowed to "say") will be
// unable to change their name while gagged
// (0=disabled, 1=enabled)
admin_gag_name 0
```

```
// If enabled, people who are gagged will not be able to use
// the say_team command.
// (0=disabled, 1=enabled)
admin_gag_sayteam 0
```

```
// Makes the admin with the highest access level the only admin in power.
// Example: if multiple admins are present, only the one with the highest
// access level will have admin access.
// (0=disabled, 1=enabled)
admin_highlander 0
```

```
// If admin_ignore_immunity is enabled, ACCESS_IMMUNITY is
// ignored and does nothing.
// (0=disabled, 1=enabled)
admin_ignore_immunity 0
```

```
// This file specifies which script plugins get loaded.
// It should be relative from the <mod> directory
admin_plugin_file "plugin.ini"
```

```
// The time (in seconds) during which an admin can reconnect after
// disconnecting without resetting his password in the setinfo line.
admin_reconnect_timeout 300
```

```
// Message displayed to users who try to execute commands that
// they don't have the appropriate access rights for.
admin_reject_msg  "You do not have access to this command."
```

```
// How often, in seconds, the repeat_message should be shown on
// the screen suring the game. Minimum value is 15 seconds.
// So, a value of 600 = 10 minutes between messages.
admin_repeat_freq 600
```

```
// Message that is show to everyone on the server every
// ten minutes by the message plugin.
admin_repeat_msg  "This server is using Admin Mod"
```

```
// The old verbosity.  Defines what commands are repeated to
// clients in the form: "[ADMIN] <user> used command <command>"
// 0=all commands labeled with the admin name
// 1=all commands labeled but without the admin name
// 2=most commands not labeled (except "cheat" commands)
admin_quiet 0
```

```
// This file is used to store configuration data across maps
// and even across server restarts.
admin_vault_file  "vault.ini"
```

```
// If enabled, a hlds_ld-style map vote will automatically
// start five minutes before the end of a map.
// (0=disabled, 1=enabled)
admin_vote_autostart 0
```

```
// If set to on status (1), when a vote is in progress all
// players will see the votes of other players as they vote.
// (0=disable, 1=enabled)
admin_vote_echo 0
```

```
// Number of seconds that must elapse after start of the map,
// or the end of another vote, before another hlds_ld-style
// map vote can be called.
admin_vote_freq 600
```

```
// Controls how many times the current map can be
// extended for thirty minutes
admin_vote_maxextend 0
```

```
// Percent of players who have to vote for a map to get it
// to win a hlds_ld-style map vote.
admin_vote_ratio 60
```

```
// Ability to make clients execute commands
// (0=disabled, 1=enabled)
allow_client_exec 0
```

```
// If set to 1, players who try to crash the server by using
// non-printable characters in mid-game will be banned from
// the server for 24 hours. Regardless of this variable's
//setting, offenders will be kicked if they do this.
// (0=disabled, 1=enabled)
```

```
amv_autoban 0
```

```
// Default access rights for players not in the users.ini file.
// (See documentation for access levels and information)
default_access 1
```

```
// encrypt_password, for LINUX ONLY, whether to use
// encrypted passwords or not
// (0=disabled, 1=enabled)
encrypt_password 0
```

```
// If enabled, the scripting file functions have read
// access to files
// (0=disabled, 1=enabled)
file_access_read 0
```

```
// If enabled, the scripting file functions have write
// access to files
// (0=disabled, 1=enabled)
file_access_write 0
```

```
// This is only used with the old script system and is not
// needed with the new plugin system
help_file "admin_help.cfg"
```

```
// Declares priority IPs that are allowed to take a
// reserved spot (if any are set up) without a password.
ips_file "ips.ini"
```

```
// Ratio of players who must vote 'yes' to a kick for
// it to be successful.
kick_ratio 60
```

```
// Ratio of players who must vote 'yes' to a map change
// for it to be successful.
map_ratio 80
```

```
// List of maps people are allowed to vote for. 0 to disable.
// Disable to enable all maps and use list from mapcycle.txt.
maps_file maps.ini
```

```
// The file (relative to the <mod> dir) that
// reserved models are loaded from. 0 to disable.
models_file  models.ini
```

```
// The message shown to someone who gets kicked for
// trying to use a reserved model.
models_kick_msg "[ADMIN] That model is reserved on this server."
```

```
// The message shown to someone who gets kicked for
// trying to use a reserved nickname.
nicks_kick_msg "[ADMIN] That name is reserved on this server.
```

```
// Password_field...first password of the setinfo line
// If password_field is "pw-AdminMod", the setinfo will be
// setinfo "pw-AdminMod" "password-in-users.ini"
password_field pw-home
```

```
// If pretty_say is enabled, centersay() fades in
// and out and does some other tricks.
// (0=disabled, 1=enabled)
pretty_say 1
```

```
// Controls how many (number) of the server's slots are reserved.
// This is useful only if reserve_type, below, is either 0 or 2.
reserve_slots 0
```

```
// Custom message given to clients trying to connect who do not have
// a reserved slots, when no public slots are free
reserve_slots_msg "There are no reserved slots available on the server."
```

```
// This controls how reserve slots work on the server. See documentation.
reserve_type 0
```

```
// NOTE: If you have "admin_plugin_file" set above, this CVAR is ignored.
// Define the path to the script file here, if you don't want
// to use the plugin system.
script_file "cstrike/dlls/admin.amx"
```

```
// If enabled, names are compared to those who have privileges
// with regular expressions.
// (0=disabled, 1=enabled)
use_regex 0
```

```
// The file in which you define your admins, their passwords
// and the access levels that they are assigned to.
users_file "users.ini"
```

```
// The minimum number of seconds allowed between votes
// called with the vote() (admin_vote functions) scripting function.
// If 0 or disabled, the vote() scripting function is disabled.
vote_freq 180
```

```
// Location of word filter file. 0 to disable, or something
// like "wordlist.txt" if enabled
words_file wordlist.txt
```

Setting up your users.ini file

Using REGEX for your user names

REGEX is a text formatting tool supported by Admin Mod , which allows you to so a number of things, including using special characters in user names, using partial names for protection or blocking, and protecting clan tags

Getting started with REGEX and AM:

It's important to realize up from that once you switch on REGEX you have to adapt your users.ini to match the REGEX rules.

In order to utilize REGEX in Admin Mod, you first must set the value of the use_regex cvar in your server.cfg file to "1" to enable it.

Once enabled, you can use a partial name in your users.ini file, for example, to protect your clan tag. Let's say your clan uses the tag [AMbT]. You could make users.ini entries like this:

```
\[AMbT\]Jaguar:PaSsWoRd:65535
\[AMbT\]Rope:paSSwOrd:65535
\[AMbT\]:PAssWOrd:16384
```

The above entries would allow Jaguar and Rope to connect using the clan tag, if they successfully authenticate, but will not allow anyone who is not listed in users.ini to connect with the [AMbT] tag. Note that with this setup, anyone who is not listed by username, but who uses the tag and supplies the correct password, would be given only reserved nickname access (16384) and nothing else in the above example.

Note that the brackets in the above example are both preceded by back-slash characters ("\"). The back-slash character denotes that the character to the right of it is to be treated as a text character, rather than as an operator. In other words, it allows non-alphanumeric characters to be used as part of a username. This is especially helpful when you are wanting to assign a username that includes any of the "forbidden characters." As a general rule, any character **not** listed below should be escaped (preceded with a back-slash) when using REGEX.

A-Z
a-z
0-9
- _ / .

A complete discussion of REGEX, it's uses and detailed explanation is beyond the scope of this manual. If you need REGEX support with Admin Mod, feel free to post your questions in the Admin Mod Forums.

More information about REGEX can be found on the web. Here's one of the many resources available:

http://www.delorie.com/gnu/docs/regex/regex_toc.html

The users.ini file is where you will store all of the player names, passwords, and access levels of the players that you want to have privileges on your server. In the interest of being complete, but not confusing some, we will start simple here and get more technical as we go. The sample users.ini file that comes in the Admin Mod distribution is documented with internal comments, as well.

The format of the users.ini like this:

```
username:password:access_level
```

This file controls the access levels of administrators. To be an administrator on the server, your name or WONID must be in this folder with an adequate password and access level. The users.ini file also serves as a nick and WONID protector (the second is obviously useless).

Without the appropriate password, a user with a playername identified in the users.ini file and the wrong password will not be able to play on the server. The users.ini file uses permissions for allowing access to command levels, which are described below.

There are two ways to identify users in this file and give them access:

```
playername:password:accesslevel
```

or

```
wonid:password:accesslevel
```

More information about access levels, how they are determined and what mean is contained at the end of this documentation section.

Each method of specifying the user's identity (player name or WONID) offers its own different advantages and disadvantages:

Player name:
Using someone's player name to give them access to ADMIN commands means that, to use the commands, the person has to use the name exactly as it is in the users.ini file. (Names AND passwords are case sensitive.) However, they can use the name and password provided on any computer, at any time.

WONID:
Using someone's WONID to give them access to ADMIN commands has the advantage of letting a player use any name he wishes while playing, because, as long as he stays on the same computer, his WONID will not change. On the other hand, because of the way the WONID is calculated, if the player who has access to ADMIN commands attempts to play on another computer, he will be unable to access those commands. And if the user is on his computer, but does not have the appropriate password, he will be kicked from the server to which he should normally have access. Had the permission been given through the use of the player name, he would have been able to access the server with a different name.

If permissions were to be given to 4 players, the users.ini file would look like this:

```
PlayerX:password1:65535
Gerg:password1:65535
Bud-froggy:password2:65535
12345678:password3:199
```

This would give PlayerX an access level of 65535 if, in his setinfo line, he had the appropriate password listed. For more information, see Client configuration via setinfo commands elsewhere in this documentation.

Players Gerg and Bud-froggy would have an access level of 65535 with two different passwords. The player with a WONID of 12345678 would have an access level of 199 with a password of "password3" and would be able to change his or her name and still retain admin access, since the WONID is what is being used to identify him/her, and that value always stays the same for any one Half-Life installation.

You can put as many player WON IDs, names and access levels as needed in the users.ini file.

As you can see, access relies on the equation between two things: name/WONID and password. In the same users.ini file, multiple players can have the same password, and the same player name can have different access levels with different passwords.

Do not write something like this:

```
player:testpass:199
player:testpass:65535
```

The users.ini file is read from top to bottom, so no matter what the second line will either be ignored or will create errors.

To overcome the problem of playername/WONID, you could grant the same access level for a playername and WONID with the same password. For example, if your name id PlayerX, and your WONID is 123445678,
your users.ini file could look like this:

```
12345678:testpass:65535
PlayerX:testpass:65535
```

This will allow a player to change names while at home (i.e. when authenticating via WON ID), and he can use his "normal" name (PlayerX) when on the road. You could also give multiple permissions for different names with the same password.

PERMISSIONS AND ACCESS LEVELS, AND WHAT THEY ALL MEAN

You determine a user’s permissions by **adding up the access levels** for all of the commands you want that person to be able to use on the server. We have provided a friendly little javascript calculator that will do the work for you (if you are reading this in HTML format, that is). The chart below shows the access levels, and the associated Admin Mod commands, followed by some simple instructions covering how to calculate the correct access level for the commands you want to provide to the user.

Explanation of how this all works:

So, If I wanted to give JoeUser access to only the admin_kick, admin_ban and admin_unban commands, I would add the access levels for those commands together:

128+256=384

So, that is the number that I would place in the users.ini file to grant him that access. If I wanted to allow BrunoMan access only to a reserved spot on the server, I would use an access level of 32768. If I decided to give BrunoMan additional access to the admin_say command, then I would add his reserve slot access value to the admin_say command value:

32768+64=32832.

For a handy access level calculator (HTML only) that will help you add up the numbers to get the right access levels, see the Access Levels page in the Admin Mod Reference section of this documentation.

Note that if you give a user access to a given level, you are giving access to *all commands* in that level (level 512 gives access to admin.cfg, admin_servercfg and admin_hostname, for example). You cannot selectively give access to only one command in a given level – they come as a package. They have been grouped in a way that makes good logical sense, though, so you will not likely find the grouping to be a problem. If you are an adventurous scripter, you can modify the access levels for each command and create your own custom security levels, but that is beyond the scope of this configuration document.

Here is a table showing the default Admin Mod access level values and their corresponding commands:

Access Level	**Commands Allowed**
all players (public commands)	admin_listmaps admin_nextmap admin_messagemode admin_nomessagemode admin_timeleft admin_userlist admin_version say currentmap say nextmap say timeleft
1	admin_vote_restart say mapvote say rockthevote say vote <map> admin_vote_kick admin_vote_map
2	admin_cancelvote admin_denymap admin_restartround say cancelvote say denymap

	admin_fraglimit admin_map admin_startvote admin_timelimit
4	admin_prematch admin_reload
8	admin_pause admin_unpause
16	admin_pass admin_nopass
32	admin_friendlyfire admin_gravity admin_teamplay admin_balance
64	admin_chat admin_say admin_ssay admin_csay admin_psay
128	admin_slap admin_slay admin_slayteam admin_kick
256	admin_ban admin_unban
512	admin_cfg admin_servercfg admin_hostname
1024	(unused)
2048	admin_gag admin_ungag
4096	makes player immune to admin commands damage
8192	admin_godmode admin_noclip admin_stack admin_teleport admin_userorigin admin_ct (CS) admin_t (CS) admin_blue (TFC) admin_green (TFC) admin_red (TFC) admin_yellow (TFC) admin_enableallweapons admin_enableequipment

	admin_enablemenu admin_enableweapon admin_restrictallweapons admin_restrictequipment admin_restrictmenu admin_restrictweapon admin_weaponscheck admin_fun admin_disco admin_llama admin_unllama admin_listspawn admin_movespawn admin_removespawn admin_spawn
16384	allow this user to use a reserved nickname
32768	allow this user to use a reserved spot
65536	admin_rcon access (use with caution) admin_execall admin_execclient admin_execteam

Setting up your plugin.ini file

The plugin.ini file is a new addition in Admin Mod 2.50

This lets a server use multiple scripts at the same time. The default installation provides a number of scripts, but has both mod-specific plugins disabled. This means that you must go into the plugin.ini file and enable the Counter-Strike or Team Fortress Classic specific plug-ins.

The default plugin.ini file looks like this when newly-installed:

```
dlls/plugin_base.amx
dlls/plugin_chat.amx
dlls/plugin_cheat.amx
;dlls/plugin_CS.amx
dlls/plugin_hldsld_mapvote.amx
dlls/plugin_message.amx
dlls/plugin_retribution.amx
dlls/plugin_spawn.amx
;dlls/plugin_TFC.amx
dlls/plugin_fun.amx
```

In the stock/default plugin.ini file hown above, note that there are two lines that are commented out by semi-colons (";"). They are:

```
;dlls/plugin_CS.amx
;dlls/plugin_TFC.amx
```

If you play TFC, you should remove the semi-colon in front of the second line. You will therefore have this:

```
;dlls/plugin_CS.amx
dlls/plugin_TFC.amx
```

If you play CS, remove the semi-colon in front of the first line that has one. You will have:

```
dlls/plugin_CS.amx
;dlls/plugin_TFC.amx
```

Obviously, there are lines in the file other than the two shown above.

The default installation uses plugin script files, and retrieves the list of files to use from the plugin.ini file. If you decide to do things differently and use only one script, you can disable the plugins by setting the admin_plugin_file cvar to 0 in your server.cfg. The script_file location must then be specified by the script_file cvar. Unless one of these cvars is defined, Admin Mod will not allow the server to start. If both cvars (admin_plugin_file and script_file) are defined in the server.cfg file, the plugin format will be used and the script_file setting will be ignored.

See also: Admin Mod Configuration Variables (CVARs), Downloading and Installing Additional Plugins and Compiling Admin Mod Scripts 101 elsewhere in this documentation.

Setting up your ips.ini file

The ips.ini will determine which IP addresses have access to reserved slots. IP stands for Internet Protocol. IP addresses are a series of 4 numbers, with each number in the range 0 to 255.

What's my IP address?

If you do not know your IP address, you can get it by running

```
Start >> Run >> winipcfg in Windows 9x.
```

For windows2000/NT users, run a command line interface (CLI):

```
Start >> Run >> cmd and enter "ipconfig" at the command prompt.
```

Linux users get their IP by entering on the command line:

```
 hostname –i or, even better, ifconfig.
```

How do I enter the IP addresses in my ips.ini file?

Single IP numbers can be entered as follows (no port numbers are required):

```
129.49.231.126
172.54.512.7
```

If you use subnets (common in Ethernet Local Area Networks) you can specify your subnet mask (kind of like and IP inside an IP)

```
129.49.231.126/255.255.255.0
```

To cover all IPs between 168.23.21.0 and 168.23.21.255, simply set the last digit to 0

```
168.23.21.0
```

To cover all IPs starting with 168.23, simply add:

```
168.23.0.0
```

Setting up your maps.ini file

This OPTIONAL plain-text file takes the same format as your server's mapcycle.txt, so it's contents would look something like this:

```
cs_aztec
cs_assault
de_dust
```

If you do not have a maps.ini file declared in the maps_file CVAR in server.cfg, all maps in the mapcycle.txt are available for vote. If this file exists and is specified in the server.cfg, it controls what maps may be voted for using Admin Mod.

How to declare the maps_file in server.cfg:

The default server.cfg that is included with Admin Mod and which is copied to your server.cfg file when you run the installation script includes an entry already for maps_file, which you can modify.

Most commonly, people will call their maps_file "maps.ini" and they declare it like this:

```
maps_file "maps.ini"
```

If you do not want to use a maps_file, you would declare as such like this:

```
maps_file 0
```

More information about setting up your server.cfg file can be found elsewhere in this chapter, in Setting up your server.cfg file.

Setting up your models.ini file

The format for this file is model_name:password

To restrict the use of the CS "sas" model without the password "pawsoff"

```
sas:pawsoff
```

Specify passwords for models that you want to reserve - Only users with password clanpass set can use gordon in normal Half-life:

```
gordon:clanpass
```

Reserve hostage models in CS so they cannot be used for cheating:

```
hostage1:off
hostage2:off
hostage3:off
```

If you use Linux you can use encrypted passwords instead
plaintext passwords by setting encrypt_passwords to 1 in
your server.cfg . DON'T mix plain-text and encrypted passwords!

This is an example of an encrypted password

```
barney:HFwILz6hzetcs
```

How to Use Bots with Admin Mod

***NOTICE TO USERS OF VERSIONS 2.50d AND EARLIER:** This section covers using bots with Admin Mod version 2.50e and later only. All Admin Mod versions prior to 2.50e use a different mechanism for supporting and running bots. If you are using a version of Admin Mod prior to 2.50e, you must consult the documentation for the earlier version.*

***Beginning with Admin Mod version 2.50e, the procedure for running bots with Admin Mod has been changed.** Bot support is now started from the command line or Windows shortcut, rather than from an .ini file. The new procedure is described here. If you have bots already running, you are not required to change your implementation, but realize that future support will be given using this new style of implementation.*

What are bots and how do I use them?
Bots are computer-generated players that can be used on a Half-Life server. The are essentially a Half-Life mod, or add-on, which is downloaded and installed just like any other Half-Life mod. Please note that because different bots interact in different ways with the server, your mileage may vary when you try to use bots with Admin Mod. Setting up the bots and Admin Mod to co-exist can be tricky, but in most all cases it can be done.

Metamod, the application that Admin Mod runs with to operate with the Half-Life server, is designed to give you the capability to install plugin applications that interact with your server. Admin Mod and Bots are examples of these types of plugin applications. An application has to be specifically written as a Metamod plugin in order to be used directly with MetaMod.

As Metamod was released in conjunction with Admin Mod 2.50, there are no bot plugins for native Metamod use available at the time of this writing. However, you can use bots with Metamod. If you used bots with an earlier version of Admin Mod the procedure and concepts may be slightly familiar to you.

Installing Bots and Admin Mod:
To use a bot (or other server mod) with Metamod that is not designed as a Metamod plugin, *you should first install the bot program*. The bot's installation will change the liblist.gam file in your mod's directory. That is why you should install the bots *before* you install Metamod/Admin Mod, as the bot installation would overwrite the changes made by the Admin Mod installation otherwise.

Next, install Admin Mod. The Admin Mod installation will alter your liblist.gam file by exchanging the bot's "gamedll" line with a new one that points to the Metamod

DLL.It will also create a backup of your original liblist.gam file for your reference and in case you want to return the file to its prior version. Now your server will run with Admin Mod when you start it. But since the bots' DLL file is no longer specified to run (because it is no longer listed in the liblist.gam file), your bots don't load. Never fear - there's one more thing you need to do.

To get the bots working, you will need to alter the command line used to start your server. If you are running a Windows server or listen-server, this means you make a change to the Windows shortcut used to start your dedicated server or game. For Linux servers, you will add additional arguments to the command line:

For Linux-based servers, add the section in bold to your server startup command line, modified to point to your bot's .so file:

./hlds_run -game cstrike **+localinfo mm_gamedll dlls/pod_bot.so**

For Windows-based dedicated servers, add the following section shown in bold to your server's startup shortcut, modified to point to your bot's .dll file:

C:\SIERRA\Half-Life\hlds.exe -game cstrike **+localinfo mm_gamedll dlls/pod_bot.dll**

For Windows-based listen-servers, add the following section shown in bold to your game's startup shortcut, modified to point to your bot's .dll file:

C:\SIERRA\Half-Life\hl.exe -game cstrike **+localinfo mm_gamedll dlls/pod_bot.dll**

- Note that you *must use forward slashes* in this command line section -- Back-slashes will not work in the +localinfo arguments!
- Your shortcut or command line may contain additional or different content than shown here. Our point is that the sections shown in bold need to be *added* to your existing startup shortcut or command line.
- The name and location of your bot's .dll file can often be determined from looking at the file called Admin Mod backup of liblist.gam in your game mod directory, assuming you installed Admin Mod after installing your server's bots. If you did not follow those instructions, consult the documentation provided with your Bot software for more information.

Protecting bots (and your server) from crashing:
When you are using bots and you use admin_ commands on them like admin_execclient, you server is likely to crash. If this is the case you can set the admin_bot_protection cvar in the server.cfg file to 1. This should prevent you from executing client commands on bots (which can be quite hard on your server).

If you start your server you should now have Admin Mod running together with the bots. If you have problems after having followed all these instructions, head on over to the forums and let us know.

Setting up Clients to Use Admin Mod

Client configuration via setinfo commands

The Basics of Client Configuration

Maybe you already have some sort of idea how this all works, and you are just trying to figure out exactly what to configure and where without the explanation?

To summarize this section, here's the basics of what you can do to get your clients up and going.

Step 1. In the server.cfg file on your server, make sure you have this line:

```
password_field "pw-home"
```

Step 2. In the users.ini file on your server, make sure you have this line:

```
UserName:PassWord:#####
```

Note that in the above example, "UserName" must be replaced with the player's name, "PassWord" must be replaced with the desired password, and "#####" must be replaced with the numeric access level to be granted to that user. More information about access levels can be found in "Setting up your users.ini file."

Step 3. And in your adminpass.cfg file on your CLIENT machine, make sure you have this line:

```
setinfo "pw-home" "password"
```

Note that "pw-home" in the above line must match the value assigned to the password_field cvar in your server.cfg file from step 1 above.

Step 4. Create a shortcut on your desktop to the game you will be playing, or copy an existing shortcut. Edit the shortcut, and in the "Target" field, add the this text:

```
+exec adminpass.cfg
```

For example, if your Counter-Strike shortcut looked like this:

```
C:\SIERRA\Half-Life\hl.exe -console -game cstrike
```

you would change it to say:

```
C:\SIERRA\Half-Life\hl.exe -console -game cstrike +exec adminpass.cfg
```

(to set up your shortcut to allow you to automatically connect to your server, add: +connect 0.0.0.0 - and put your server's IP address in place of the zeros)

Remember: Never use the shortcut for admin privileges to connect to a server other than the one you are an admin on, or *you will risk exposing your password to the outside world*.

If you need to load your password configuration file again while in the game, you can type:

```
exec adminpass.cfg
```

in the console. This loads the information in the file in the same way the startup shortcut does.

In order for a person to successfully log onto your server and gain access through Admin Mod, you must prepare the client computer ahead of time. That is what this section covers.

The setinfo line is the most important part of client access to the Admin Mod on a server. It is also the part that tends to cause the most difficulty, but it doesn't need to. It requires some level of knowledge about what is going on when you connect to a server running Admin Mod. We'll try to explain that here in a fairly simple fashion.

When you ran the Windows installation script for Admin Mod, you may have specified some information about your username and password, in which case a text file called adminpass.cfg was probably created in your mod directory on the server. Although the file was created on the server, it is *not supposed to be used there*. You should now **move** the adminpass.cfg file to your **client machine** (the client is the computer that you play your game on).

Note: You may be running a listen server , in which case your client and server are on the same computer. In that case, the file can stay right where it is.

What is this file for? Basically, when you connect to the server, the Admin Mod looks at your player name that you have specified in your game settings. If your name is listed in the users.ini file on the server, it recognizes that you are connecting as an admin, and then checks your password against the one it has in it's users.ini file to make sure it's really you.

The way the server interacts with the client, in simplified terms, is like this: Once the server recognizes your player name as being on the admin list (in users.ini), it requests information about your password form your client machine (that's the machine you play on and are using to log into the server).

In order to request that information from your client machine, it makes a request based on where it expects the password to be stored, sort of like you going to your mailbox to get your mail, since the client's password storage mechanism has a unique identity and can hold information that you put in there. Let's say your mailbox is called "pw-home" and you have specified that in the server.cfg file, like this:

```
password_field "pw-home"
```

If that is the case, the server is going to send your game a message asking the client machine to send anything that is stored in the "mailbox" called "pw-home." For those who prefer to use accurate terms rather than metaphors, it's actually a "variable," or some would call it a "field."

Therefore, you need to store your player password in the field called pw-home on your game machine. You do this using a "setinfo" command, which is simply a command that Half-Life uses to store information in specified places. In tech terms, "pw-home" is a variable for storing data, and "setinfo" is a command used to assign values to Half-Life game variables. In this case the data being stored is your Admin Mod password.

The setinfo line that needs to be used on the client machine looks like this:

```
setinfo "pw-home" "password"
```

with "pw-home" being the mailbox variable defined by the password_field setting in the server.cfg or listenserver.cfg , and the "password" being the same one listed in the server's users.ini file for your player name or WON ID.

For example, if your users.ini file looked like this:

```
PlayerX:testpass:65535
```

and the (listen)server.cfg file had:

```
password_field "pw-home"
```

the setinfo line for PlayerX would be:

```
setinfo "pw-home" "testpass"
```

For any player attempting to gain access to your server, the first item in quotes in the line above (again, as defined in password_field) will always be the same. The second quoted item - the player's password - will give the player access to your server, as long as the password provided here matches the one set on the client machine. Once again, in our "mailbox" metaphor, the mailbox name (field where info is stored) is called "pw-home" and the user's password is "testpass"

The setinfo line can be specified in many places. Every person that will connect to your server as an Admin needs to follow these instructions. Note that we recommend only one safe and secure way to do this. Please follow these instructions carefully.

On the client (game) machine, go into the <mod> directory of the game you will play on the server running Admin Mod.

For example, if you play TFC and your HL is installed on your c:\ drive, it would be:

```
c:\Sierra\Half-Life\tfc
```

Your **adminpass.cfg** file should contain the information you need already, if you created it with the Windows installer for Admin Mod, but check it to be sure. You were instructed to put a copy of this file in your mod directory earlier in this chapter. If the file is not already there, you either need to go back and copy it, or you can create it with a plain-text editor such as Notepad, emacs, vi, Textpad, etc.

First off, you need to edit the **adminpass.cfg** file with a plain text editor such as notepad, and add these lines if they are not already there:

```
setinfo "pw-home" "password"
developer 1
echo [ADMIN] password has been set on client
developer 0
```

Make sure you replace "*pw-home*" and "*password*" in the example above with the appropriate information from the password_field in your server.cfg and the password listed for your user name in the users.ini file. Be sure to include the quotation marks around the words as shown above - you need to keep those to ensure the file will be read properly by the game when it loads.

Finally, **create a shortcut** on your desktop to the game you will be playing, or copy an existing shortcut to your game. Edit the shortcut (right-click on it and choose "Properties," and then click on the "Shortcut" tab), and in the "Target" field, add the this text: "+ exec adminpass.cfg"

For example, if your Counter-Strike shortcut originally looks like this:

```
C:\SIERRA\Half-Life\hl.exe -console -game cstrike
```

you should change it to read:

```
C:\SIERRA\Half-Life\hl.exe -console -game cstrike +exec adminpass.cfg
```

To set up your shortcut to allow you to automatically connect to your server, you can also add:

```
+connect 0.0.0.0:[port]
```

to the shortcut entry - but remember to put your server's IP address in place of the "0.0.0.0" and the port number for your server if needed in place of "[port]" in the example:

```
C:\SIERRA\Half-Life\hl.exe -console -game cstrike +exec adminpass.cfg +connect
192.168.0.1:27015
```

Remember: Use this new shortcut to connect **only** to the server you have admin privileges on, or you will risk exposing your password to the outside world. Rename this shortcut to something obvious that will identify it as the one that invokes your Admin Mod password. You might call it, for example, "ADMIN Counter-Strike."

If you need to load your password configuration file again at any time, you can type:

```
exec adminpass.cfg
```

in the console. This loads the information in the file in the same way the startup shortcut does.

If all the passwords and variables are in the right place and are (of course) correctly entered, you automatically have access to the server whenever you connect to it. You can also play on any other servers without changing anything. Remember that everything, passwords and names, are case sensitive.

Admin Mod Plugins

The Admin Mod Default Plugins

The following plugins are provided with Admin Mod. A brief description of each one is provided as well. To get into the meat of what each plugin does, do to the Admin\scripting\examples directory, where you will find the text .sma plugins in their original, uncompiled format (for example, plugin_base.sma is the uncompiled version of plugin_base.amx). Opening these .sma files in a text editor will allow you to examine the contents of each plugin and to learn specifically about how they work.

(Note: Don't ask us to make a command list for you - read the files yourself. This is not kindergarten)

:)

plugin_base.amx
This plugin contacins the main, core admin mod functions, such as kicking and banning, admin_say and csay (etc.), user lists, rcon access, and the like.

plugin_chat.amx
Contains a few commands for responding to chat messages (currentmap, timeleft, nextmap).

plugin_cheat.amx
Includes actions which would generally be considered cheating by a player during a game. Includes godmode, noclip, stack, teleport, etc.

plugin_CS.amx
Contains commands to be used specifically and only with Counter-Strike game servers. Includes commands like weapon restrictions, ability to force players to join either the CT or T teams, restartround, etc.

plugin_hldsld_mapvote.amx
Contains AM commands to cancel vote, deny map and start vote, as well as public "say" style commands to do similar tasks plus actually participate in the votes.

plugin_message.amx
Shows message on connect as well as the periodic center_say message. There are no commands provided in this plugin, just functionality.

plugin_retribution.amx

Contains commands for dealing with cheaters, llamas, and the like. Includes such commands as slay, slap, gag, llama, ungag, unllama, etc.

plugin_spawn.amx
As it says in its name, provides commands for spawning things.

plugin_TFC.amx
Provides commands specific to and to be used with TFC game servers. Includes team balancing, changing teams, prematch setting.

plugin_fun.amx
Fun mode commands are provided in this plugin, including (but not limited to) disco mode, glow commands, and - well - anyhow....

Downloading and Installing Additional Plugins

Having trouble integrating those %@#&$^! plugins?

Yes, we know, plugins were supposed to resolve all the problems for those who were not hard core programmers. so here is a step-by-step plugin for dummies tutorial:

> *These instructions are geared toward a Windows user compiling the plugins. If you are using Linux, please note that the procedure is similar, but where we refer to Windows batch file, you will need to use the equivalent Unix shell script.*

There are really only two ways to download a plugin:

- from the plugin page of the Admin Mod web site
- from the Admin Mod forums

In either case, the task is simple to perform:

- From the plugin page, right-click (for Windows) on the Download link, and choose "Save Target As."
- Go to the ..\Admin\scripting\myscripts folder. (It is a folder that you extracted when you extracted Admin Mod from the zip file.)
- Change the "Save as type" from "Text Document" to "All files". Press save and watch the file download.

If you download the script from the forum, the file will either be in the zip format, or in a plain text format.

- If the first case presents itself, unzip the CONTENTS of the file into the ..\Admin\scripting\myscripts folder.
- Otherwise, right click on the link, and do as mentioned above.

So, you now have the plugin you wish to implement on your server in the scripting\myscripts folder. Go to that folder and double click on the compile_all.bat file. (Only files that start with "plugin_" and end in a .sma entension will be compiled automatically.)

Your scripts will be "translated," or "compiled" into a file that the program can understand. These compiled files will be placed in the ..\Admin\scripting\mybinaries folder automatically by the compile_all program. You will notice that the extensions of your compiled scripts will be different. They no longer end with ".sma" but with ".amx" instead. That is how you know it is a compiled plugin.

- Copy all the .amx files into the ..\half-life\<mod>\dlls folder (<mod> being your game folder).

The last step in integrating the files is adding their locations in the plugin.ini file. This file was created when you installed Admin Mod 2.5.

- Find the plugin.ini file in the mod directory (i.e. half-life\cstrike or half-life\tfc).
- Open this file in a text editor and add the location of the plugins you want to add.

  At the end of the file, add an entry in this format:

  ```
  dlls/plugin_nameofplugin.amx
  ```

  For example, if I downloaded the plugin_coffee.sma file, the compiled file would be plugin_coffee.amx. To add that file to your plugin.ini, simply add this to the last line in the file:

  ```
  dlls/plugin_coffee.amx
  ```

Now you can fire up your server, and try the new plugin. You must restart your server or change maps for new plugins to take effect.

Scripting Basics

A Beginner's Guide to Scripting

A SCRIPTING CRASH-COURSE

This is not intended to be a complete guide to programming in the Small language. For that, there is a good set of complete documentation available at:

http://www.compuphase.com/small.htm

The point of this section is to give new and fledgling coders a chance to learn the basics of authoring and compiling plugins for Admin Mod. For information about compiling completed plugins, please take a look at Compiling Admin Mod Scripts 101 in this documentation.

Admin Mod uses a programming language called Small, which is a derivative of the C programming language. So, technically this isn't scripting - it's programming. When you write Small code, you eventually compile it into a binary form. So it's not really scripting. But we call it that. :0)

This section takes for granted that you have some basic programming fundamentals of some sort under your belt.

Declaring variables:

To declare a variable, you must use the "new" declaration. And, unlike C/C++, You can use this to declare any variable, may it be a string (a lot of characters put together...kind of like a phrase) or a number. So if you wanted to declare a new number, you would use:

```
new iValue = 0;
```

If you wanted to declare a string, you would use:

```
new sName[MAX_DATA_LENGTH];
```

where MAX_DATA_LENGTH is the number of characters that string can have. It is predefined by Admin Mod to 200. Augmenting the number could cause clients to crash, so playing with this is not recommended.

If you wished to give a value to that String when you define it, simply add its value:

```
new sName[MAX_DATA_LENGTH] = "Hello world.";
```

For those of you who have no idea what is going on, just keep in mind that the one thing that you can ALWAYS edit is whatever is found between quotes. In other words, where a plugin has text output, such as in the "Hello World" example above, you could easily replace the text inside the quotes without to much fear of ruining things. Doing so will not create any errors unless an unknown symbol is used.

Commenting:

Comments are VERY valuable. Something that is "commented out" is completely disregarded by the compiler. This means that you can do absolutely anything in a commented-out area. Authors use this to make comments on their code and explain to other users how the code works. Comments come in two formats:

```
//
```

and

```
/* */
```

The first type of comment (//) tells the compiler to disregard everything that comes after it in a line. For example, if you wrote:

```
// new iVal1=0;
  new iVal2=0;
```

the variable iVal1 would NOT be declared. Since "new iVal2=0;" is on a different line, iVal2, on the contrary will be declared.

The second (/* */) comments out everything between /* and */. For instance:

```
new iVal1=0; /*
new iVal2=0;
*/ new iVal3=0;
```

the declaration of iVal2 would be disregarded because it is between /* and */ unlike the declaration of iVal1 and iVal2. This type of comment is often used like this:

```
/****************************
Admin Mod 2.5
****************************/
```

Here, everything is commented out, because when you look closely, this is the same as:

```
/*
***************************
Admin Mod 2.5
***************************
*/
```

Note, that /* */ comments CANNOT be nested, i.e. you should not place comment inside of comments. It is safe to have //-style comments in /* */- style commented regions.

Using functions:

Using functions (methods or snippets) is easy. For example, the streq method, which compares two values. Here is the code:

```
streq(String1,String2,iMaxlength) {

 new iNum;

// be careful about how much we check
if (iMaxlength > strlen(String1)) iMaxlength=strlen(String1);
if (iMaxlength > strlen(String2)) iMaxlength=strlen(String2);
for(iNum=0;iNum<iMaxlength;iNum++)
if (String1[iNum]!=String2[iNum]) return 0;
return 1;
}
```

To use this code, you would write another function:

```
admin_foo ()  {
{
new String1[MAX_DATA_LENGTH] = "Hello world.";
new String2[MAX_DATA_LENGTH] = "Little Bunny Foo Foo";
streq(String1, String2, MAX_DATA_LENGTH);
}
```

The result would obviously be false (0) because these two strings are different. Had the two been the same, the result would be 1. You could also just add this is inside an existing method, but be careful not to use a variable that might already be in use for something else. All of the methods in Admin Mod are (should be) clearly explained by their authors.

Statements and Loops:

Statements and loops are the basic parts of any computer program. They are the "thinking" portion of a program.

There are 3 basic statements and loops that are used in Admin Mod. They are:

- if/else
- while
- for

The first is quite self explanatory:

```
if (condition)
{
//do something here
}
```

This means that if the condition is fulfilled than the program will execute what is between the braces. If the condition is not met, the code inside the braces will be ignored. You can add to the if statement with "else"s. For example:

```
if (iNum = = 0 )
{
iNum = 34;
}
else {
iNum = 94;
    }
```

If iNum has a value of 0, iNum will be reassigned to 34. If iNum is not 0, then iNum will be assigned to 94. To spice things up, you can add a bunch of if's inside each else. For example:

```
if ( iNum = = 0 ) {
          iVal = 2;
} else if ( iNum == 1 ) {
          iVal = 3;
} else if ( iNum < 0 ) {
           streq( sStringA, sStringB, MAX_DATA_LENGTH);
} else {
           iVal = 10;
}
```

The while loop is the less used loop, but it can be very useful. Its construction is:

```
while (condition)
{
//do something
}
```

The program will keep on doing that something until the condition IS met. For example:

```
while (iNum<0)
{
say("Please enter a positive number");
}
```

The program will keep on prompting for a positive number as long as the number imputed is not positive. If the user puts a positive number in on the first try, then the program will ignore the loop right away. The for loop is the most used loop. It is in almost every complicated method or function out there. Its structure is:

```
for(starting number; condition; implements number)
{
// do something
}
```

The program will go through the loop, starting at the starting number, and will do something with it, then restart the whole loop, after implementing the number. An example is necessary for this loop:

```
for(iNum=0; iNum<15; iNum++)
{
iCount++;
}
```

This is a simple for loop, just for explanation. The loop will start with iNum=0 (you must have previously defined iNum) and, since iNum=0 is smaller than 15 will add 1 to count (which must have also been defined previously). Since there is nothing left to do inside the loop, iNum will be incremented by 1, and the loop will continue. iNum=1 is smaller than 15, so count is incremented by 1 and since there is nothing else, iNum is incremented. The loop will break when iNum=15, because iNum=15 is not smaller than 15. The result of this loop is obvious, but normally, it is not easy to understand for loops.

Other random things:

There are a multitude of other functions that this small crash-course tutorial will not cover. The most basic will be briefly explained:

break
The break statement ends the loop in which it is nested (inside). No matter the circumstances, if the program comes upon a break statement, the loop will be instantly terminated.

return
The return statement ends the method or function in which it is nested. Like the break statement, it will automatically end all processes that are currently underway. For example:

```
admin_foo() {
            new iNum;
            new iCount;
            new sString1[MAX_FATA_LENGTH] = “Hello world”;
            new sString2[MAX_DATA_LENGTH] = “LittleBunnyFooFoo”;
            if ( streq(sString1, sString2, MAX_DATA_LENGTH) ) {
            for( iNum = 0; iNum < 15; iNum++ ) {
                        if ( iNum > 5 ) {
                                    break;
                        }
                        iCount++;
}
if  ( iCount < 343 ) {
                        return 1;
}
if ( iNum == iCount ) {
                         do_something();
}
}
```

In this example, as soon as, in the for loop, iNum is bigger than 5, the loop will end, without increasing iNum or count. In this far-fetched example, count will always be smaller than 343 so the method admin_foo will end and return a value of 1.

Compiling Admin Mod Scripts 101

The point of this section is to let anyone create their own version of the Admin Mod script. This handy little program is open-source, meaning that the authors only want groveling and absolute submission from you. Unlike Bill Gates, they believe you should keep your money so that you can actually do something with your nerdy, pointless, pathetic lives.

:0)

Okay - enough of that - back to the topic at hand... You may also want to check out the section called A Beginner's Guide to Scripting in this document for more information about the scripting language works from a programmer's point of view.

This section assumes you know something about file extensions and how to properly view them on your computer. Nothing personal, but if you don't know how to do this, you probably shouldn't be running a server and chances are you don't have one running yet. Go do some learning and come back once you're ready.

DOWNLOADING ADMIN MOD PLUGINS

There are really only two ways to download a plugin:

- From the plugin page of the Admin Mod web site
- From the Admin Mod forums

In either case, the task is simple to perform. From the plugin page, right-click (for Windows) on the Download link, and choose "Save Target As."

Go to the ..\Admin\scripting\myscripts folder. (It is a folder that was created when you extracted Admin Mod from the zip file.) Change the "Save as type" from "Text Document" to "All files" if you are prompted. Press save and watch the file download. The result should be a file in the "myscripts" folder that ends with ".sma"

If you download the script from the forum, the file will either be in the zip format, or in a .txt extension. If the first case presents itself, unzip the CONTENTS of the zip file into the ..\Admin\scripting\myscripts folder. Otherwise, right click on the link, and do as mentioned above.

A BASIC COMPILING TUTORIAL

First of all, the scripts that you write or make changes to are **plain text files** with a file name "extension" of ".sma". The "extension" part of a filename is the three letters after the period at the end of the file name.

Compiling is the process of changing coding language into assembly line language, or a language that the computer can understand and use. When you compile your *.sma file, you will that the output will be a *.amx file. The *.amx file is the **compiled plugin file** that Admin Mod uses. There is no way to retrieve the original script from a compiled file, so do not delete your *.sma file. Keep it around just in case you might need it or want to make changes in the future.

Compiling files is really very simple. The Admin Mod v2.50 release offers unprecedented ease of use. Everything is already set up for you.

To compile your own script (or add-on scripts that you've downloaded), simply place your new .sma file in the Admin\scripting\myscripts directory. Note that your file name must end with. sma for all this to work, so if it ends with something like .txt or the like, you will need to rename it.

Run the compile_all.bat in your "myscripts directory" - This nifty program looks for scripts you have dropped in the "myscripts" folder (again, it expects *.sma files) and compiles them.

It then places the compiled plugins that resulted from the compiling process in the "mybinaries" folder. So, after compiling, just look in the "mybinaries" folder for your compiled plugin files, which will have the same name, except that they will end with ".amx" - which denotes the file is a compiled binary version of your script.

You now need to copy all the .amx files you just compiled into the ..\Half-Life\\dlls folder (being cstrike, tfc, or any other modification you are hosting). Unless you copy them here, your game server cannot use them.

The final step in making the plugin files work is adding their locations to the plugin.ini file. This file was created when you first installed Admin Mod. Find the file in the mod directory (i.e. half-life\cstrike or half-life\tfc).

Open the plugin.ini file and add the location of the plugins you want to add. Usually you can just add the plugin to the bottom of the list of plugins already in your plugin.ini file. The format used to specify a plugin is:

```
dlls/plugin_nameofplugin.amx
```

For example, if I downloaded the plugin_coffee.sma file, the file name after I compiled it would be "plugin_coffee.amx" to add that file to your plugins, simply add dlls/plugin_coffee.amx

ERRORS AND WARNINGS DURING COMPILING

Disregard any "Warnings" in the text that is displayed on the console screen during compiling sessions, as they are simply warnings.

However, if you get "errors" during compiling, you are in trouble. The compiling process will not produce a valid *.amx file when it encounters errors (or if it does, the .amx file will have a file size of 0 bytes). So, in order to come up with a working plugin, you have to go and find the errors in your code and fix them.

Remember that errors can occur due to faults in lines that come both before and after the line number indicated by the compiler. The line number indicated in an error message is simply the one where the compiler realized it could not make sense of what was going on. The problem that the compiler choked on could have occurred somewhere else in the script, but as the compiler read the code it did not recognize the error until it ran across something else that didn't work. The most common errors are spelling, loose/missing/extra braces, and undefined variables.

In the code used to make Admin Mod plugins, one loose brace (braces are the **{** and **}** characters) can create a great deal of errors, so after every error you correct, recompile and see if that solved the problem. Keep on searching for mistakes until the compiling process returns a valid *.amx file.

Then have fun!

See also: A Beginner's Guide to Scripting

Compiling scripts by hand

You can compile your script at the command line simply enough, assuming you are comfortable working at the command line and that you have a basic knowledge of directory traversal in text mode.

Start in the "myscripts" directory of your Admin Mod installation, and compile using sc (or sc.exe) with the command:

```
..\compiler\sc -i..\include xxx.sma
```

where xxx.sma is replaced with your script name.

Note: This is for Win32, for Linux replace all backslashes (“ \ “) with forward slashes (“ / ”)

You might see either warnings or errors, or both, displayed. Warnings are okay – one of the more common is a warning that your indentation is “loose” – Okay, so you’re a little sloppy, not that big of a deal. However, errors are a problem – they indicate that there is a flaw that is keeping the compiler from successfully parsing and compiling your script. The key thing to look for is for the compiler to show you no errors. If you see the message “Compilation aborted” at the end of it’s output, it did not work. If you get nothing but warnings and maybe even a message that says, “Done,” you have a compiled script that is ready to test in the server environment.

A note about working with the compiler output:
Small Compiler errors and warnings contain some useful information that can help you in determining what is wrong with your script when it generates them. The first item on an error or warning line is the name of the file you are compiling, followed by a number in parentheses, such as “(217)” – This is the line number where the compiler encountered the problem, which is useful information if you are trying to find out where the problem might be.

Keep in mind that a list of 20 errors on 20 lines might just be dependant upon one problematic line in the script. For example, if I accidentally typed three curly-brackets ({{{) in a row instead of the proper single curly bracket, I might get errors on multiple lines according to the compiler. So, I would choose to go to the first listed line with an error and look there, make my fix, and then recompile and see what happens.

Remember: recompile often to check to see where you’re at, and always make periodic backups of files, so that you can go back to an older version of your script when your brain gets so mushy from vapor lock that you can’t remember what changes you’ve made.

If all goes well, the compiler produces a file called yourscriptname.amx, which will be located in the same directory as your original yourscriptname.sma file. Now copy the .amx to the /dlls directory for your mod (i.e. half-life\cstrike\dll) and fire up the server and test. Rinse and repeat until you get the desired results.

NOTE - Linux and windows .amx files ARE DIFFERENT, You MUST recompile for each platform. It's the same source code file, just slightly different binaries.

More on Manual Compiling …
Isn't there an easier way to do these compiling things? What's all this dot-dot-slash stuff??? Sheez!

The simple answer is, "Yes." You could copy all of the files from the \include directory, along with the sc.exe and your .sma files, into a new directory and run it from there, thus negating the need to do all the dot-dot-slash stuff. But you're learning here, and if you're going to be doing this sort of stuff it helps to understand what's going on, and so here's some information about the dot-dot-slash concept for those who are command-line challenged:

The dots and slashes are part of how we communicate with the computer to tell it where we want to go, or where we want it to go to find something on the drive. Here is an attempt to illustrate what those things mean:

.	(a single dot)	This refers to the directory you are in now
..	(two dots)	This tells the computer to look in the "parent" directory
../mydirectory	(dot-dot-slash-dir_name)	Tells the computer to go to the parent directory (one level above) and then look for a directory called "mydirectory" and go there.
../mydirectory/cool.exe	(dot-dot-slash-dir_name-slash-program_name)	Tells the computer to go to the parent directory (one level above) and then look for a directory called "mydirectory" and go there, and run the program called "cool.exe"

Admin Mod Reference

Materials for reference when installing, configuring and using Admin Mod

- Admin Mod Commands
- Access Levels
- Admin Mod Configuration Variables (CVARs)
- Admin Mod Error Codes
- Frequently Asked Questions (FAQ)

Admin Mod Commands

Note: This list of commands includes all the commands that are provided with Admin Mod "out of the box." It does not include any commands that are created by the use of additional plugins. Please see the new plugin's documentation and/or admin_help for assistance with any additional plugins you add to your server's setup.

admin_help <keyword or string | # >: Displays information about available commands

admin_ban <target or WONID or IP> [<minutes>]: Bans target for specified number of minutes. 0 minutes is a permanent ban.

admin_cancelvote: Cancels the current HLFD-style vote (Votes done with "say")

admin_cfg <config file>: Executes the named config file on server.

admin_chat <msg>: Sends a private message to all other admins on the server.

admin_csay <color> <msg>: Shows message in center of screen. <color> is optional.

admin_ct <target>: Force targeted player to change to CT team (CS only)

admin_denymap <map>: Removes all votes for map during vote.

admin_disco: Starts disco fever on all players. Requires fun mode be on.

admin_enableallweapons: (CS plugin only) Allows all weapons to be purchased. Use to remove all active weapon restrictions.

admin_enableequipment: (CS plugin only) Allows all equipment to be purchased. Use to remove all active equipment purchase restrictions.

admin_enablemenu <menu #>: (CS plugin only) Remove restriction from specified menu.

admin_enableweapon <menu #> <weapon #>: (CS plugin only) Remove restriction from specified weapon.

admin_execall <command>: Force every player to execute a command locally (limited).

admin_execclient <target> <command>: Force specified player to execute local command.

admin_execteam <team> <command>: Force everyone on team to execute local command.

admin_fraglimit <fraglimit>: Sets the server's mp_fraglimit cvar.

admin_friendlyfire <on | off>: Sets the server's mp_friendlyfire cvar.

admin_fun <on | 1 | off | 0>: Turns fun mode on (1) or off (0), allowing a variety of other commands to be used on the server.

admin_gag <target> [<minutes>]: Gag the specified target for the specified number of minutes. Specifying 0 minutes results in a permanent gag. (see admin_ungag)

admin_glow <color | "off">: Causes you to glow that color. Requires fun mode.

admin_godmode <target> <"on" | "off">: Sets godmode on target.

admin_gravity <gravity>: Sets the sv_gravity cvar.

admin_hostname <name>: Sets the hostname cvar.

admin_kick <target> [<reason>]: Kicks target.

admin_listmaps: Shows maps in mapcycle.

admin_listspawn: Lists all spawned entities.

admin_llama <target>: Llama-fy target. Changes name to Llama and makes them speak in a new language.

admin_map <map_name>: Changes map without disconnecting players, same as using changelevel via rcon on a server.

admin_messagemode <command>: Will treat things you 'say' as command.

admin_movespawn <identity> <X> <Y> <Z> <XAngle> <YAngle> <ZAngle>: Moves a spawned item.

admin_nextmap: Shows next map in the cycle.

admin_noclip <target> <"on" | "off">: Sets noclip on target.

admin_nomessagemode: Will treat 'say' as 'say' (removes messagemode).

admin_nopass: Clears the server's password if set.

admin_pass <password>: Sets the server's password.

admin_pause: Sets the pausable cvar to 1.

admin_psay <target> <msg>: Sends a private msg to target player.

admin_rcon <cmd>: Executes specified rcon command on server.

admin_reload: Reloads Admin Mod files.

admin_removespawn <identity>: Removes a spawned item.

admin_restartround <#>: Restarts the round on the same map in # seconds

admin_restrictallweapons: Disallow purchase of all weapons on server

admin_restrictequipment: Disallow purchase of all equipment on server

admin_restrictmenu <menu #>: Disallow purchase of anything from specified menu

admin_restrictweapon <menu #> <weapon #>: Disallow purchase of specified weapon

admin_say <msg>: Shows a text message from you as admin.

admin_servercfg <config file>: Sets the specified config file as the server's default.

admin_slap <target>: Slaps target player.

admin_slay <target>: Slays (kills) target player.

admin_slayteam <team>: Slays everyone on the specified team.

admin_spawn <class> <X> <Y> <Z> <XAngle> <YAngle> <ZAngle>: Spawns a new item.

admin_ssay <msg>: Shows a text message from admin without identification.

admin_stack: Will stack everyone on top of you.

admin_startvote: Starts an HLFD-style vote.

admin_t <target>: Forces targeted player to join the Terrorist team (CS only)

admin_teamplay <teamplay>: Sets the mp_teamplay cvar. (TFC only)

admin_teleport <target> <X> <Y> <Z>: Teleports target to the given coordinates. See admin_userorigin for related command.

admin_timeleft: Shows the time left on the map.

admin_timelimit <timelimit>: Sets the mp_timelimit cvar. Value = minutes.

admin_tsay [color] <msg>: Prints msg on lower right of screen in the color specified.

admin_unban <WONID or IP>: Unbans specified WON ID or IP address..

admin_ungag <target>: Ungag target.

admin_unllama <target>: Unllama-fy target.

admin_unpause: Sets the pausable cvar to 0, making the server non-pausable by players

admin_userlist [<name>]: Shows a list of players on the server.

admin_userorigin <target>: Returns the X, Y, Z coordinates of target player.

admin_vote_kick <target>: Starts a public vote to kick target.

admin_vote_map <map>: Starts a public vote to change the map.

admin_vsay <question>: Presents the specified question as a vote.

say rockthevote: Starts an HLFD vote by typing the command in "say" mode

say vote <map>: Places a vote for the map.

say glow <color | "off">: Causes you to glow that color. Works in fun mode only.

Admin Mod Configuration Variables (CVARs)

This is a listing of all of the server configuration variables (cvars) specific to Admin Mod. You set these like any other server cvar (such as mp_timelimit, etc), can place them in your server.cfg, and the like.

NOTE: This is not a list of Admin Mod commands. Rather, this is a list of all the server-side configuration variables you can use when you are configuring your Half-Life server.

Please note that a 'dedicated' server will, unless told otherwise, read it's initial settings from 'server.cfg'. A 'listen' server, on the other hand, reads them from 'listenserver.cfg' Setting variables in server.cfg when you're running a listen server, and vice versa, does nothing. A listen server is one that you start from within Half-Life itself: it starts when you join, it ends when you quit, and you have a 0 ping. A dedicated server is started from the HLDS program, and appears on the server machine as just lots of text information in a DOS-like console window.

Document notation standards
A cvar can either accept numeric ('123') or string ('abc') data. If it can accept string data, it will look like this in this document:

```
cvar_name "<data>"
```

If it can accept numeric data, it will look like this:

```
cvar_name <#>
```

In most cases, numeric data will either be '0' (for no or disable), or anything else (for yes or enable). If the description of a cvar talks about it being 'enabled', that means non-zero. Exceptions are noted. Note that if a cvar is not explicitly set, it defaults to 0 (usually disabled).

In the descriptions below, examples of how to write the cvar are provided. reading this entire document is a good idea if you are planning to do much at all with the Admin Mod.

About "relative directories" and files
Cvars that represent files are said to be 'relative' to a directory (usually the <mod> directory). This means the file name should be treated as if looking from the specified 'relative' directory. As an example, admin_plugin_file is relative to the directory; if I'm running a TFC server, my directory might be C:\HLServer\TFC. If I wanted my plugin file to be at 'C:\HLServer\TFC\plugin.ini',

I would simply set admin_plugin_file to 'plugin.ini'. If I wanted my plugin file to be at 'C:\HLServer\TFC\AdminMod\plugin.ini', I would set admin_plugin_file to 'AdminMod\plugin.ini'.

Miscellaneous Notes

- If vote_freq is not defined voting will be disabled.
- If map_ratio is not defined map voting is disabled.
- If kick_ratio is not defined kicking will be disabled.
- If models_file is not defined model passwords will be disabled.
- If nicks_file is not defined nick reservation will be disabled.
- If maps_file is not defined then users can vote for any map in the mapcycle.txt file. If it is defined, all votable maps must be included in the file indicate by the maps_file cvar (typically this file would be named maps.ini)
- You can un-define or make null any variable by setting to 0 like this: nicks_file 0

---

admin_balance_teams <#>

Used by the TFC plugin. If enabled, an attempt to switch teams will be rejected if it would unbalance the teams (unless the team being switched to, or the team being switched from, has less than two people). If disabled, people can switch teams freely.

admin_bot_protection <#>

When running bots on the server, setting this variable to "1" will protect bots from exec_client commands in Admin Mod, and thus prevent server crashing problems.

admin_connect_msg "<data>"
admin_connect_msg "All your base are belong to Jaguar."

Used by the message plugin. This is the message that will be displayed to everyone thirty seconds after connecting to the server. Setting admin_connect_msg to "0" will not display a connect message, although the timer is run.

admin_cs_restrict <#>

Used by the CS plugin. If enabled, the CS plugin's weapon restrictions go into effect (if any are set, that is). If disabled, the CS weapon restrictions are ignored (if any are set).

admin_debug <#>

If enabled, your logs will be filled with lots and lots of debugging messages related to Admin Mod. Not recommended for general use.

admin_fun_mode <#>
Used by the Fun plugin. Determines whether or not the fun commands are allowed.

admin_fx <#>
Enables special effects to accompany certain commands, such as teleport and slap.

admin_gag_name <#>
Used by the retribution plugin. If enabled, people who are gagged will be unable to change their name while gagged. If disabled, people who are gagged will be able to change their names as normal.

admin_gag_sayteam <#>
Used by the retribution plugin. If enabled, people who are gagged will not be able to use the say_team command. If disabled, people who are gagged will be able to use the say_team command as normal.

admin_highlander <#>
There can be only one! Normally (when disabled), everyone gets the access assigned to them. If admin_highlander is enabled, only the person with the highest access actually gets their's; everyone else gets the default. Thus, only one admin (the one with the highest access level)will be able to execute commands at any given time.

admin_ignore_immunity <#>
This is referenced by CheckImmunity() in adminlib.inc. Normally (when disabled), people with ACCESS_IMMUNITY (4096) become immune to many of the other admin commands (though not to straight rcon). If admin_ignore_immunity is enabled, ACCESS_IMMUNITY is ignored and does nothing.

admin_plugin_file "<data>"
admin_plugin_file "plugin.ini"
This is the file to load plugins from. It should be relative compared to the <mod> directory (e.g., for TFC, it should be relative compared to the 'half-life\tfc' directory). If this cvar exists, the plugin-style of scripting will be used. If it does not exist, the single-script style of scripting will

be used (see script_file). Note that, regardless of where
this file is situated, the plugins it referenced are relative
to the <mod> dir, _not_ the location of the plugin file
(eg, if the admin_plugin_file is '<mod>\AdminMod\plugin.ini',
the plugins referenced will still be relative to '<mod>',
NOT '<mod>\AdminMod').

admin_reconnect_timeout <int>
admin_reconnect_timeout 300

This is the time that a password is considered to stay valid
after the player has disconnected from the server. If the
player reconnects to the server within <int> seconds using the
same name and the same IP, he does not have to reenter his
password first. Don't set this value too high.

admin_reject_msg "<data>"
admin_reject_msg "This is a restricted command which you cannot use."

This message is displayed to users who try to execute
commands that they don't have the appropriate access rights
for.

admin_repeat_freq <int>
admin_repeat_freq 300

Used with the admin_repeat_message cvar. This determines how often
(in seconds) the "pretty say" repeat message appears on the screen. The
repeat message frequency can be set with admin_repeat_freq. The units
are seconds. The minimum value is 15 seconds. Setting it to 0 will not
start the timer. The timer cannot be started by setting it to a different value
during a map. It will only be started at a map change.

admin_repeat_msg "<data>"
admin_repeat_msg "Don't look now! They're right behind you!"

Used by the message plugin. This is the message that is shown
to everyone on the server in the center of the screen. Setting
admin_repeat_msg to "0" will not display a repeat message, but
the timer is not killed if it was started. It will continue to run, and
setting admin_repeat_msg to a different value will again display that string.

admin_quiet <#>
admin_quiet 0
admin_quiet 1
admin_quiet 2

This is referenced by SayCommand() in adminlib.inc. Here's how it
functions by default:

- **Certain commands override admin_quiet** (such as those found in the cheating plugin). These over-riding commands, when used, will *always* show the message 'ADMIN Command: <Player> used <command>" regardless of what value this cvar is set to.

- **If admin_quiet is 0**, executing admin command on others will result in the command being executed and the message 'ADMIN Command: <Player> used <command>' being displayed to all players.
- **If admin_quiet is 1**, the same effect occurs as with the "0" value: Amin Mod will display the message 'ADMIN Command: Admin used <command>' but will not identify the admin by name.
- **For any other value of admin_quiet** (for example, "admin_quiet 2"), all commands will not display a message at all, but will only get logged, *with the exception of the commands that over-ride the cvar as mentioned above* (generally these are commands that would be considered cheating if no one knew they were being executed).

admin_vault_file "<data>"
admin_vault_file "vault.cfg"

This file will be used to store configuration data across maps and even across server incarnations.

admin_version "<data>"

Returns the current Admin Mod version of the DLL. Setting this does nothing.

admin_vote_autostart <#>

Used by the HLFD map vote plugin. If enabled, a HLFD-style map vote will automatically start five minutes before the end of a map. If disabled, it won't.

admin_vote_echo <#>

If set to 1, when a vote is in progress the option a user voted for is echoed to all clients in the format:

<player> voted for option #<option number>

admin_vote_freq <#>

This is the number of seconds that must elapse after the start of the map, or the end of another vote, before another HLFD-style map vote can be called by someone without the ACCESS_CONTROL_VOTE access. NOTE: If you don't have it explicitly set in in your .cfg, it defaults to 600 (ten minutes). If 0, then _only_ those people with ACCESS_CONTROL_VOTE may call for votes. Note that this only controls the HLFD-style votes; for the HL menu-style votes, see vote_freq.

admin_vote_maxextend <#>

Used by the HLFD map vote plugin. Controls how many times the current map can be extended for thirty minutes (e.g., admin_vote_maxextend 2 would allow for, at most, two extensions). If set to zero or a negative number, the current map can never be extended.

admin_vote_ratio <#>

Used by the HLFD map vote plugin. Controls the percent of the players who have to vote for a map to get it to win (note that this ratio applies only to the HLFD-style map vote.
For the admin_vote_map, see map_ratio, below). If zero or negative, whichever map gets the most votes wins. Otherwise, a map must get at least (admin_vote_ratio * playercount / 100) votes to win (e.g., if the admin_vote_ratio is 60, and there are 10 people on, a map must get at least (60 * 10 / 100) = 6 votes to win.)

alarm_message "<data>"

This cvar is no longer used.

alarm_time <#>

This cvar is no longer used.

allow_client_exec <#>

This controls whether or not the execclient() scripting function is enabled in the Admin Mod DLL. If enabled, execclient() is enabled; if disabled, execclient() is disabled. Enabling execclient() functionality allows scripts to execute commands remotely on players.

amv_autoban <#>

If set to 1, players who try to crash the server with non-printable characters (aka ~%?%?%?) in mid-game will be banned for 24 hours. In either case are they dropped from the server.

default_access <#>

This controls the default access that people get (aside from being granted special user privileges). It works exactly like the user level access rights, but everyone gets it.

encrypt_password <#>

LINUX ONLY. NOTE: If you don't have it explicitly set in your .cfg, it is ENABLED. This determines whether or not the passwords people enter for their user privileges are encrypted on the server or not.

file_access <#>

This cvar is no longer used. See file_access_read, file_access_write.

file_access_read <#>

This controls whether or not the scripting functions are allowed to read files on the server. If enabled, the scripting functions are; if disabled, they are not.

file_access_write <#>

This controls whether or not the scripting functions are allowed to write to files on the server. If enabled, the scripting functions are; if disabled, they are not.

help_file "<data>"
help_file "admin_help.cfg"

This is not used for the plugin-style scripting; if you have admin_plugin_file defined, this cvar is ignored. If you are using the old single-script style, this is the file to load the help information from. It is relative to the <mod> directory.

ips_file "<data>"
ips_file "ips.ini"

If you're using MySQL, this cvar is ignored (see mysql_dbtable_ips). Otherwise, this is the file (relative to the <mod> dir) that priority IPs are loaded from. NOTE: If you don't have it explicitly set in your .cfg, it is 'ips.ini'. Priority IPs are those that are allowed to take a reserved spot (if any are set up) without a password.

kick_ratio <#>

Used by admin_vote_kick. This is the ratio of players who must vote 'yes' to a kick for it to be successful. e.g. if the kick_ratio is 60, and there are 20 people on the server, 12 of them must vote 'yes' to a kick vote for it to be successful.

map_ratio <#>

Used by admin_vote_map. This is the ratio of players who must vote 'yes' to a map change for it to be successful. e.g., if map_ratio is 40, and there are 20 people on the server, 8 of them must vote 'yes' to a map vote for it to be successful. Note that this cvar does not control the HLFD-style map vote; see admin_vote_ratio.

maps_file "<data>"

maps_file "maps.ini"

A list of maps available during map votes. This list can contain maps not listed in your regular server map cycle, or it can contain a limited number of maps if you want to restrict voting to just a few maps. If you disable it (by setting the value to "0"), the list of votable maps will instead be taken from your mapcycle.txt

models_file "<data>"
models_file "models.ini"

If you're using MySQL, this cvar is ignored (see mysql_dbtable_models). Otherwise, this is the file (relative to the <mod> dir) that reserved models are loaded from.

models_kick_msg "<data>"
models_kick_msg "This model is reserved."

This is the message shown to someone who gets kicked for trying to use a reserved model.

mysql_dbtable_ips "<data>"
mysql_dbtable_ips "ips"

If you're not using MySQL, this cvar is ignored (see ips_file). Otherwise, this is the database table to get the priority IP information from. Priority IPs are those that are allowed to take a reserved spot (if any are set up) without a password.

mysql_dbtable_models "<data>"
mysql_dbtable_models "models"

If you're not using MySQL, this cvar is ignored (see models_file). Otherwise, this is the database table to load the reserved model names from.

mysql_dbtable_plugins "<data>"
mysql_dbtable_plugins "plugins"

If you're not using MySQL, this cvar is ignored (see users_file). Otherwise, this is the database table to load the plugins from.

mysql_dbtable_users "<data>"
mysql_dbtable_users "users"

If you're not using MySQL, this cvar is ignored (see users_file). Otherwise, this is the database table to load the privileged names from.

mysql_dbtable_words "<data>"
mysql_dbtable_words "words"

If you're not using MySQL, this cvar is ignored (see words_file). Otherwise, this is the database table to load the swear words from.

mysql_host "<data>"
mysql_host "127.0.0.1"
If you're not using MySQL, this cvar is ignored. Otherwise, it's the address of the host the MySQL database exists on.

mysql_pass "<data>"
mysql_pass "milk"
If you're not using MySQL, this cvar is ignored. Otherwise, it's the password used to connect to the MySQL database.

mysql_user "<data>"
mysql_user "moocow"
If you're not using MySQL, this cvar is ignored. Otherwise, it's the username used to connect to the MySQL database.

nicks_kick_msg "<data>"
nicks_kick_msg "This name is reserved."
This is the message shown to someone who gets kicked for trying to use a name that has reserved access (16384).

password_field "<data>"
password_field "pwd-home"
This is the setinfo field that people need to enter their password into on the client before connecting to the server, if they wished to be authorized at connect time. NOTE: If you don't have it explicitly set in your .cfg, it is 'pw'. As an example, if the password_field is 'pw-home', and there is a user whose password is "milk", then the setinfo line in the user's adminpass.cfg would be 'setinfo "pw-home" "milk"'.

password_timeout <#>
This cvar is no longer used.

pretty_say <#>
This cvar controls how the centersay() script function works. NOTE: If you don't have it explicitly set in your .cfg, it is ENABLED. If pretty_say is enabled, centersay() fades in and out and does some other tricks. If it's disabled, centersay() just appears as normal text in the middle of the screen.

public_slots_free <#>

Returns the number of slots free on the server (after taking into account reserved slots). Setting this does nothing.

reserve_slots <#>

Controls how many of the server's slots are reserved. This is useful only if reserve_type, below, is either 0 or 2.

reserve_slots_msg "<data>"
reserve_slots_msg "There are no free slots available."

This is the message shown to someone who gets kicked when there are only reserved slots left on the server, and they do not have reserved access.

reserve_type <#>

This controls how reserve slots work on the server (the default is 0).

* **reserve_type 0:** Public slots are used in preference to reserved slots. Reserved slots are freed before public slots.
* **reserve_type 1:** One slot is always reserved (regardless of reserve_slots). If someone with reserve access joins into that slot, the highest pinger without reserve access is kicked to make room. Thus, one slot always remains free.
* **reserve_type 2:** Reserve slots are used in preference to public slots. Public slots are freed before reserved slots.

The difference between reserve_type 0 and reserve_type 2 may not be immediately apparent. Here's an example: Suppose there is a 16 player server, with 2 reserved slots. Thus, with no one on, there are 14 public slots, and 2 reserved slots:

* **reserve_type 0:** Someone with reserved access joins. There are now 13 public slots, and 2 reserved slots (the person does not take a reserved slot, because those are used only when necessary). The server later fills up, so there are 0 public slots and 0 reserved slots. If anyone leaves, whether they have reserved access or not, it is a reserved slot that is freed, not a public one.
* **reserve_type 2:** Someone with reserved access joins. There are now 14 public slots, and 1 reserved slot (the person takes a reserved slot). The server later fills up, so there are 0 public slots and 0 reserved slots. If someone with reserved access leaves, it frees up a reserved slot; if someone without reserved access leaves, it frees up a public

slot.

script_file "<data>"
script_file "tfc\dlls\admin_win32.amx"
If you have admin_plugin_file set, this cvar is ignored, as it is used to set up the older-style single scripts used in versions of Admin Mod prior to v2.50.

.

Otherwise, this is the file (note the path is relative to the <half-life> directory...NOT the <mod> directory) that the compiled script is loaded from. If this cvar is disabled and admin_plugin_file is disabled, Admin Mod won't run.

use_regex <#>
If enabled, names are compared to those who have privileges with regular expressions; otherwise, they are compared with a straight match. Using regular expressions is useful for matching parts of names (such as clan tags), whereas, without use_regex, you would have to add each name with that tag you wanted to assign privileges to.

users_file "<data>"
users_file "users.ini"
If you're using MySQL, this cvar is ignored (see mysql_dbtable_users). Otherwise, this is the file (relative to the <mod> directory) that privileged names are loaded from.

vote_freq <#>
This is the minimum number of seconds allowed between votes called with the vote() scripting function (such as admin_vote_kick and admin_vote_map). If 0 or disabled, the vote() scripting function is disabled. Note that this delay also applies to the beginning of the map; if vote_freq is 240 (240 seconds = 4 minutes), admin_vote_kick and admin_vote_map won't work for the first four minutes of the map, as well as for the four minutes following any previous admin_vote_kick or admin_vote_map. Note that this only applies to HL vote-style votes; for HLFD-style votes, see admin_vote_freq.

vote_freq_kick <#>
This cvar is no longer used. See vote_freq.

vote_freq_map <#>
This cvar is no longer used. See vote_freq.

words_file "<data>"

words_file "wordfile.txt"

This is the file containing the list of words that will match the check_words() and censor_words() functions (useful for swear word detection). Note that these words are case-insensitive.

User Access Levels - Version 2.50d

For information about setting access levels for your users, please see: Setting up your users.ini file

Note: Some commands changed access levels beginning with v2.51d (Voting control commands moved out of public access levels). This chart lists only the default Admin Mod commands. If you add additional functionality via plugins, the commands provided by those plugins will also be assigned access levels. Consult the plugin text files and/or any documentation packaged with each plugin.

To use the calculator (web documentation version only), check the boxes for the access levels you want to give to your users. Once you have selected al the access levels for a user, press the "Calculate" button at the bottom of the page. The correct value to be inserted in to the users.ini file will be displayed below. Please note that adding plugins or altering the standard scripts will result in either additional or fewer commands being available than shown in this table, as this calculator pertains only to the default access levels and commands.

Grant Access?	**Access Level**	**Commands Allowed**	
	public commands	admin_listmaps admin_nextmap admin_messagemode admin_nomessagemode admin_timeleft admin_userlist admin_version	say currentmap say nextmap say timeleft
☐	1	admin_vote_restart say mapvote say rockthevote	say vote <map> admin_vote_kick admin_vote_map
☐	2	admin_cancelvote admin_denymap	admin_fraglimit admin_map admin_startvote admin_timelimit

		admin_restartround say cancelvote say denymap	
☐	4	admin_prematch	admin_reload
☐	8	admin_pause	admin_unpause
☐	16	admin_pass	admin_nopass
☐	32	admin_friendlyfire admin_gravity	admin_teamplay admin_balance
☐	64	admin_chat admin_say admin_ssay	admin_csay admin_psay
☐	128	admin_slap admin_slay	admin_slayteam admin_kick
☐	256	admin_ban	admin_unban
☐	512	admin_cfg admin_servercfg	admin_hostname
☐	1024	(unused)	
☐	2048	admin_gag	admin_ungag
☐	4096	*makes player immune to admin commands damage*	
☐	8192	admin_godmode admin_noclip admin_stack admin_teleport admin_userorigin admin_ct (CS) admin_t (CS) admin_blue (TFC) admin_green (TFC) admin_red (TFC) admin_yellow (TFC) admin_enableallweapons admin_enableequi	admin_restrictallweapons admin_restrictequipment admin_restrictmenu admin_restrictweapon admin_weaponscheck admin_fun admin_disco admin_llama admin_unllama admin_listspawn

		pment admin_enablemenu admin_enableweapon	admin_movespawn admin_removespawn admin_spawn
☐	16384	*flags this user name as a reserved nickname*	
☐	32768	*allow this user to use a reserved server spot*	
☐	65536	admin_rcon (use with caution) admin_execall admin_execclient admin_execteam	

Spawning HL Entities

This is a quick and dirty tutorial on spawning things. Why is a spawning tutorial in the Admin Mod documentation, you ask? Well, we get a lot of questions about how to use the spawning commands, and the almighty second god of Admin Mod, Jaguar, wrote this tutorial back when he first gave Admin Mod spawning capabilities. It is still completely valid today, and barely edited. Praise Jaguar! (We're not worthy!)

Jag speaks:

So, this tutorial's broken down into two parts:

- Adding stuff onto a map ahead of time
- Adding stuff onto a map at run time

Adding stuff onto a map ahead of time
First, you need a map. I play TFC, so I'm going to use 2fort as an example. How many people haven't dreamed of plopping down a Xen tree or 3 on the bridge of 2fort? Well, ok...more of you than I expected, but that's what we're going to do anyways. =P

For you CS people, you can just follow along (it's pretty easy), or one of you can write me up an example for CS, which I can include.

Anyways, you need 2fort.bsp. This is found in all server installs, so you should have it if you have a server, in hlserver\tfc\maps. Before doing anything else, BACK UP THE MAP! Copy it to 2fort.bak or something.

Now, you've got two options: you can edit the .bsp file by hand, or you can export the entities and edit those.

Option 1: Editing by hand requires you to have a decent text editor; one that doesn't barf at characters above ASCII 127. What you need to do is look for the entity information. 2fort.bsp is roughly 2.5 megs; in it, the entity information begins at offset 255800h. You should see this:

```
{
"wad" "\tffree\tfc\tfc.wad;\tffree\valve\halflife.wad;\tffree\valve\liquids.wad;"
"mapversion" "340"
"mapversion" "220"
"sounds" "1"
"message" "2 Forts Classic"
"skyname" "dusk"
"MaxRange" "4096"
"classname" "worldspawn"
"classname" "worldspawn"
}
```

All of the entities follow in the same format: enclosed in braces ({}), with key/value pairs describing the data.

Option 2: Exporting the entity list requires Zoner's Half-Life Editing Tools (ZHLET), which you can find here:

http://halflife.gamedesign.net/resources/zhlt.shtml

Once you've downloaded the tools (and compiled them, if need be), run the ripent.exe tool:

```
ripent -export (map)

  ripent -export d:\hlserver\tfc\maps\2fort.bsp
```

This will create a file in the same directory as the .bsp, but with the extension .ent (eg, in the example, it would create d:\hlserver\tfc\maps\2fort.ent). This .ent file is the entity list from the .bsp file, and can be edited (IMHO) much easier, allows for easier searching, etc.

Moving on...

What do these key/value pairs do? Some of them are obvious. Some of them are not. Some of them _seem_ obvious, but are actually not (who would guess that 'maxammo_shells' key on a 'tf_detect' entity actually controls the class limits for team 1? I wouldn't...)

The most comprehensive entity list I've found so far is on Wavelength:
http://www.planethalflife.com/wavelength/levels/entities/index.html

(It covers TFC entities...but not CS ones. Sorry.) Probably the easiest way to start is to find a map that already has what you want in it, and take a look at its entities, copying as you go. But for this little example, I'll just give you the entity for a xen_tree:

```
{
"renderamt" "255"
"rendercolor" "0 10 200"
"origin" "0 0 0"
"angles" "0 0 0"
"classname" "xen_tree"
}
```

Oh, but there's a problem. What are the coordinates of where we want to put it?

You've got three ways to get them:

- Using a program to decompile the .bsp into a .map file (such as winbspc for Win32 machines), open the map in Worldcraft and locate the coordinates that way.
- Load the map up into a dedicated server on your machine. Walk around until you find the spot, and then use the 'status' cmd to get your coordinates.
- If you have Admin Mod, and are using the plugin scripts, load the map up into a server and then user the 'admin_userorigin (player)' cmd to get your coordinates.

Using one of these methods, get the coordinates of where you want to place the xen_tree. I'm going to go ahead and just use ones I've already gotten: X:0, Y:0, Z:-480.

So, update the entity info with your new coordinates (origin) data:

```
{
"renderamt" "255"
"rendercolor" "0 10 200"
"origin" "0 0 -460"
"angles" "0 0 0"
"classname" "xen_tree"
}
```

Put that at the end of the entity list (making sure to be very careful if you're editing the .bsp by hand). If you're using ripent, then you have to import the .ent file back in ripent -import (map)

```
  ripent -import d:\hlserver\tfc\maps\2fort.bsp
```

Once this is done, load the map up into a server. A cute li'l xen tree should be awaiting you on the bridge.
The tree can be removed easily...just remove the entity information from the bsp or the ent (being sure to import when done if editing an ent). Similarly, other trees can be added at other spots, or other entities entirely could be added (more spawn points in CS, ammo packs for TFC in the hldm maps, etc).

Adding stuff onto a map at run time
Adding stuff on the fly requires one of two things:

- sv_cheats 1 enabled.
- Admin Mod, latest version

However, before you can add anything onto a map, (and this is the important bit) it has to already have been precached. Things can only be precached when the map loads. In general, this means that the only things you can add onto a map at run-time are things that already exist on the map.

It boils down to this: If you want to add a xen tree somewhere at run time, a xen tree must already exist on the map. It doesn't matter where (it doesn't have to be 'in the map')...but it's got to exist.

So, getting back to 2fort, remove the original xen_tree (if necessary), and add this in:

```
{
"renderamt" "255"
"rendercolor" "0 10 200"
"origin" "5000 5000 5000"
"angles" "0 0 0"
"classname" "xen_tree"
}
```

Note the origin; 5000 5000 5000 is well outside the map. No one will ever see, touch, or interact with this xen tree in any way. It's only purpose in life is so that we can spawn other xen trees. Without it, we can't spawn any xen trees at all.

With this down, we can now spawn an army of xen trees, anywhere we want.

Using sv_cheats, the cmd is 'give xen_tree'. It'll spawn one wherever you are.

Using Admin Mod, the cmd would be 'admin_spawn xen_tree (X) (Y) (Z) (XAngle) (YAngle) (ZAngle)'. The angles can default to 0, but having a ZAngle of 180 will give you an upside down Xen tree, which is kind of amusing. =)

Developer Change Log - AM plugins

15.06.2001: (2.50e)

- Made the mp_timelimit storing in plugin_base more efficient.
- Added the plugin_exec() native function definition to admin.inc

26.05.2001:

- admin_map stores the mp_timelimit. This is useful for people who don't have mp_timelimit set in their server.cfg.

28.05.2001:

- Added KillGlow() function to turn off glow when admin_fun is set to 0.

02.05.2001:

- Fixed various typos throughout

plugin_CS:
- The restrict message is lowercase to make it narrower.

plugin_TFC:
- Fixed a bug using admin_tfc_balance instead of the correct cvar name admin_balance_teams.

plugin_base:
- admin_hostname uses the whole name, not only the first word.
- admin_map and admin_vote_map store the mp_timelimit before they set it to 1. It is restored after the map change.
- admin_pass will echo the current password if none is supplied.
- admin_vsay checks if a vote is allowed.

plugin_hlds_mapvote:
ACCESS_CONTROL_VOTE replaces ACCESS_ALL for admin_cancelvote, admin_denyvote and admin_startvote.
Fixed an echoing message and the extend vote

plugin_message:
- Setting admin_connect_msg to "0" will not display a connect message, although the timer is run.
- Setting admin_repeat_msg to "0" will not display a repeat message but the timer is not killed if it was started. It will continue to run and setting admin_repeat_msg to a different value will again display that string.

- The repeat message frequency can be set with admin_repeat_freq. The units are seconds. The minimum value is 15 seconds. Setting it to 0 will not start the timer. The timer cannot be started by setting it to a different value during a map. It will only be started at a map change.

plugin_retribution:

- admin_slay uses the new slay() function and is thus independent from the setting of allow_execclient. It will only issue the message and the thunderclap noise if the target was really slain. It respects admin_quiet.
- The same is true for admin_slay_team.

admin.inc:

- Added rainbow() function.
- Added slay() function
- Overriding admin_quiet will not print the admin's name.

Developer Change Log - AM DLL

June 15, 2001: Changes made for version 2.50e

- Added system error reports to CPlugin::LoadFile() function.
- Added the plugin_exec() function.
- Fixed the bot_protection in teleport() not showing special fx.
- Added #pragma pack() to the end of amx.h to stop the packing.
- Tried to move include directives so that amx.h always gets included last. This enabled MySQL support under Windows.
- Fixed a bug in timer.cpp::IsValidTimer() checking for the wrong constant.
- Fixed a bug in the bad name detection. make_friendly replaces % now with spaces instead of deleting them.
- Fixed a bug in the logging functions of centersay et al which messed up the formatting.

June 2, 2001: Changes since the admin-2_50-release CVS tag

CLinkList.cpp:
CLinkList.h:
The linked list and list item classes have been converted to templates. This is necessary because we can't use a void* pointer to store data in the items. Using a void* pointer will cause a memory leak since the destructor of an object stored will not be called. The template declaration includes an argument to specify if the list item object will be used to store a single element or an array of elements. This is necessary to call the correct delete operator on destruction. All other files have been adapted to use the templates. I removed some C-type casts which are unnecessary when using the templates. We should gradually change to C++-type casts if they are necessary.

The insertion was fixed which fixes a bug of registered commands not showing up in admin_help.

Makefile:
- Well, just a little cleanup, removing the double (and wrong) optimization flags. Removed linkfunc.ccp from the MM build.

admin_commands.cpp:
- The wrong time bug in ban() has been fixed. Log string size has been increased to LARGE_BUF_SIZE.
- Added debug message to plugin_register*() functions.
- tsay(), centersay() and rainbow() messages get logged in the logs with newlines replaced with "\n" strings.
- vote_multiple() function displays all options, not only the last one.
- Added const_casts on STRING() calls in order to use the original HLSDK STRING() function and get rid of those compiler warnings.

- Jag moved the contents of the getteamcount() function around. =)
- Added slay() function which is independent from the setting of allow_execclient.
- Added more bot protections if admin_bot_protection is on. It should work now also if admin_fx is turned on.
- Fixed gettarget() and pointto() crashes under Linux. The Linux version of CS and TFC don't like the HLSDK EyePosition() function.

admin_mod.cpp:

- Cvars added: admin_repeat_freq, admin_vote_echo, amv_autoban. Other CVars have changed default settings: admin_connect_msg, admin_repeat_msg.
- AM_ClientConnect(): Setting some structures to 0 seems to have fixed that mysterious crash when compiling optimized.
- Check for player names with non-printable characters. They are denied access. If they change name in mid-game, their HL is closed on the client machine and a 24h ban issued depending on the setting of amv_autoban.
- Metamod stuff changed to match new interface version.
- Added MM function GetEngineFunctions_Post()
- Changed DispatchObjectCollsionBox() to hide the timer entity.

extdll.h:

- Fixed MSVC compiling problems when compiling in DEBUG mode and added #pragma warning(disable : 4018) to hide the signed/unsigned mismatch warnings.
- Adapted to use the original STRING() function and updated functions UTIL_LogPrintfFNL() and make_friendly().

hexport.cpp:

- Catches engine functions to hide the timer entity. This removed the crashes in TFC.

make_pass:

- Updated to better make use of the new encryption scheme, improving usability.

MSVC schtuff:

- Oh well, updated the workspace files to work. Added resource file adminmod.rc which adds a version information to the Windows DLL.

timer.h:
timer.cpp:

- Added MaxVoteChoice member and access functions. Updated the StartVote() function to use ChoiceCount.

users.cpp:

- Removed ' * ' from the list of comment characters.
- MySQL fixes from Jon Paul Nollmann.
- Most changes are to use the new templates.

users.h:

- Added DEBUG_LOG and DEVEL_LOG macros to make it easier to log debug messages.

util.cpp:

- Changed make_friendly() to check for non-printable characters. The check is optional and can be enabled with the second.
- Added UTIL_LogPrintfFNL() function which acts like UTIL_LogPrintf() but converts newlines within the string to "\n" literals, if there is enough space in the array, else they are replaced with blanks.
- UTIL_LogPrintf() uses vsnprintf() instead of vnprintf() to prevent buffer overflow.
- CLIENT_PRINTF() replaced with SystemResponse(), fixing crash when the player lookup function found more than one match on a partial name.

Admin Mod Error Codes

Not that you'd hope to have to deal with the information on this page, but hey - if it's broken, sometimes you need this stuff to fix it. If you get an error code when running Admin Mod , just look it up in this table to cross reference to a name and description.

This is the list of error codes for Admin Mod 2.50.

A note about the error numbers: AMX_ERR_NONE starts at 0, and they count up from there, so that AMX_ERR_SLEEP is 12, and then it takes the jump to 16 for AMX_ERR_MEMORY, where it continues to count. Error Codes 13 through 15 are not used.

Error Code	**Error Name**	**Description**
0	AMX_ERR_NONE	
1	AMX_ERR_EXIT	forced exit
2	AMX_ERR_ASSERT	assertion failed
3	AMX_ERR_STACKERR	stack/heap collision
4	AMX_ERR_BOUNDS	index out of bounds
5	AMX_ERR_MEMACCESS	invalid memory access
6	AMX_ERR_INVINSTR	invalid instruction
7	AMX_ERR_STACKLOW	stack underflow
8	AMX_ERR_HEAPLOW	heap underflow
9	AMX_ERR_CALLBACK	no callback, or invalid callback
10	AMX_ERR_NATIVE	native function failed
11	AMX_ERR_DIVIDE	divide by zero
12	AMX_ERR_SLEEP	go into sleepmode - code can be restarted
16	AMX_ERR_MEMORY	out of memory
17	AMX_ERR_FORMAT	invalid file format
18	AMX_ERR_VERSION	file is for a newer version of the AMX
19	AMX_ERR_NOTFOUND	function not found
20	AMX_ERR_INDEX	invalid index parameter (bad entry point)
21	AMX_ERR_DEBUG	debugger cannot run
22	AMX_ERR_INIT	AMX not initialized (or doubly initialized)
23	AMX_ERR_USERDATA	unable to set user data field (table full)
24	AMX_ERR_INIT_JIT	cannot initialize the JIT
25	AMX_ERR_PARAMS	parameter error

Admin Etc

Frequently Asked Questions (FAQ)

Remember - help is available in the Admin Mod web forums, if your questions cannot be answered in the documentation.

- When I try to start Admin Mod on my server, it doesn't work and I get a message telling me that I "must define script_file or admin_plugin_file on my server." What's wrong?
- How do I get that green thing that appears in the middle of my screen to work and say what I want?
- I can't seem to use the weapon and menu restriction commands?
- How do I get my WON ID?
- Where can I find a list of commands?
- I don’t understand the whole password deal.
- When my server changes maps, all the clients get “dropped.” What the #&^$%?
- 56k-ers lag my server. How can I ban them?
- The map returned by nextmap is not the next map in the cycle. How come?
- How do I uninstall Admin Mod?
- I was kicked from my own server!!!
- I saw a server that let people forgive teammates for team killing. How can I do that?
- I am banned from my server. How do I get back in?
- I am getting Bad Nick Info every time I try to connect!!!
- How can I make map specific configurations?
- My server crashes every 10 minutes/My server crashes when I use admin_csay.
- I run a Linux server and I have a Win32 compiled script? Why wont it work?

Q: When I try to start Admin Mod on my server, it doesn't work and I get a message telling me that I "must define script_file or admin_plugin_file on my server." What's wrong?

There is no one, simple answer to this question. What this error is indicating is that Admin Mod is trying to start, and the DLL has been successfully loaded, but for some reason your server.cg pr listenserver.cfg is either incorrectly configured or is not being read correctly.

Here is a partial list of things to check to try to resolve the problem:

1. Make sure there is only one copy of the admin mod cvars (configuration variables) in your server.cfg or listenserver.cfg file. Note that if you run the installation script more than once, you may end up with multiple copies of this file, which will cause problems when you try to run your server.
2. Make sure your server.cfg/listenserver.cfg file is smaller than 16KB in size. Due to a restriction in the Half-Life server code, .cfg files cannot be larger than this size. Files over 16KB in size will fail to load.
3. If you are running a Windows server, make sure your file names are all spelled correctly. You must have file extensions for all file types viewable in order to be able to accurately determine if the files are named correctly. This is configurable in the options for Windows Explorer. Filenames are required to be spelled correctly and must be located in the correct directories to be found and to function. Almost all the files used with Admin Mod are to be located in the game mod directory. Plugin files that have been compiled and the Admin Mod DLL files are located in the game mod's "dlls" directory.
4. Make sure you are configuring the .cfg file that is located in the game mod directory of the game you are running on your server.
5. The obvious: Make sure either admin_plugin_file or script_file is specified in your server.cfg/listenserver.cfg file, and that your files are named and located where the .cfg file indicates.
6. Reinstall Half-Life and server files, then update to the latest Half-Life and server versions from the Internet, and then reinstall the latest version of Admin Mod, following the directions carefully.

Q: How do I get that green thing that appears in the middle of my screen to work and say what I want?
Ahh, the never-ending question. The answer is simple. In your (listen) server.cfg, enable the pretty_say cvar. Then, in the same file, enter:

```
admin_repeat_msg "This is the XXXXXXXX server.”
```

To make a new line, simply add a ^n where you want the new line. For example:

```
admin_repeat_msg "This will be the first line.^nThis, the second.”
```

Q: I am running a Counter-Strike Server. After I installed Admin Mod, I can't seem to use the weapon and menu restriction commands that Admin Mod is supposed to allow for Counter-Strike servers? Why not???

You need to take a look at the page in this manual called Setting up your plugin.ini file**,** which will explain what you need to do in order to enable Counter-Strike specific plugin capability. It also explains how to enable TFC-specific capability, by the way.

Additionally, you will need to set your "admin_cs_restrict" cvar in server.cfg to "1" or else all will be for not. If all of the following are true, it should work:

1. You are running a Counter-Strike server
2. You have "admin_cs_restrict 1" in your server.cfg
3. You have enabled the plugin_cs.amx in the plugin.ini file
4. You have executed commands to restrict stuff (of course)

The commands and cvars involved are all listed in the reference section of this documentation.

Q: How do I get my WON ID?
There are many ways to view your WON ID. The easiest is to join any random server, pull down the console (press "~") and enter "status" (without the quotes). Then search for your player name and look for a big long number on the same line. That is your WON ID.

Q: Where can I find a list of commands?
Check the Admin Mod Commands page in the Admin Mod Reference section of the documentation.

Q: I don't understand the whole password deal.
A summary: here's what you can do to get going. In the (listen)server.cfg on your server, make sure you have this line:

```
password_field "pw-home"
```

In the users.ini on your server, make sure you have this line:

```
username:password:65535
```

And in your adminpass.cfg on your CLIENT, make sure you have this line:

```
setinfo "pw-home" "password"
```

In each of the above examples, you *must* replace "username" with the player name, and "password" with that player's password. Note that we now advise people not to use autoexec,.cfg for setinfo information storage - You should use adminpass.cfg instead, for security reasons. For more information about configuring these aspects of Admin Mod, refer to "Client configuration via setinfo commands" elsewhere in the documentation.

Q: When my server changes maps, all the clients get "dropped." What the #&^$%?
This is typically not Admin Mod's fault. One of the more common causes of this is an enormously large custom.hpk file.

When players connect with custom decals, these get uploaded to the server. The server stores them in the custom.hpk file and sends them to the other clients. But the custom decals never get deleted from the custom.hpk file which is why it increases in size over time. If the file gets too large it takes the server a long time to load and search it. There have been custom.hpk files over 30MB large. If you delete it the server simply creates a new one. The new HLDS has cvars to control the maximum size of the custom.hpk file.

Aside from that, if you want to point fingers, point yours at that heap of metal under your desk. Admin Mod uses very little system resources on top of the Half-Life server process. HLDS on the other hand requires some horsepower and plenty of RAM to operate most efficiently. There are online resources available for server administrators to learn and troubleshoot problems with HLDS, including the forums at http://server.counter-strike.net/ and the Valve Software dedicated server mailing lists for both the Windows and Linux servers.

Q: 56k-ers lag my server. How can I ban them?
Sorry, but there is no way to ban HPB. Admin Mod does not check the ISP of clients, nor does it find the serial number of the modem, nor does it download credit card numbers (rats!). You can encourage them to leave by setting sv_maxrate and sv_minrate higher, therefore making 56k-ers lag.

Q: The map returned by nextmap is not the next map in the cycle. How come?
There has been some confusion in this area, so just to set things straight: Admin Mod uses the maps.ini file to restrict map votes to a set of maps. It uses the file specified by the HLDS mapcyclefile cvar for map cycle information, usually mapcycle.txt. The next map gets out of sync if a map was changed to by a vote or by a manual map change. Admin Mod reports back the next map in the mapcycle.txt file but the HLDS will jump back into the mapcycle where it left off before the manual change.

Q: How do I uninstall Admin Mod?
You don’t. :D Just kidding. It's really very simple:

1. Remove the Admin Mod specific CVARs from your server.cfg or listenserver.cfg (whichever one applies to your situation).

2. Then edit your liblist.gam file to return the gamedll filed back to "mp.dll" (Windows) or "dlls/cs_i386.dll" (in the case of Linux running under CS). Or, if you used the installation script to originally install Admin Mod, just find the "AdminMod Backup of liblist.gam" file in your mod directory and copy it over the current liblist.gam (assuming your previous liblist.gam was working properly before you installed Admin Mod).

Q: I was kicked from my own server!!!
Your users.ini file and setinfo line may not be working together. Read through the manual to make sure you have set everything up correctly. Help on this topic is available in the sections labeled Client configuration via setinfo commands and Setting up your users.ini file.

Q: I saw a server that let people forgive teammates for team killing. How can I do that?
This option is only available for Linux users at this time, and it is not done by Admin Mod. It can be done on Linux servers with an application called HLFD. There are also some remote console programs that can detect team killing and take action, but again - they are not related to Admin Mod..

Q: I am banned from my server. How do I get back in?
In your mod directory (cstrike, tfc , or other) there is a file called banned.cfg. Inside will be your WONID. If you know your WONID, simply remove yours, and only yours. If you do not know your WONID, simply delete the contents of the file.

Q: I am getting Bad Nick Info every time I try to connect!!!
This means that your users.ini file and/or your setinfo lines are not set up correctly. Read through the manual and make sure you have followed instructions. Help on this topic is available in the sections labeled Client configuration via setinfo commands and Setting up your users.ini file.

Q: How can I make map specific configurations?
Admin Mod 2.50 has a new option that allows Admin Mod commands to be directly entered into the dedicated server CLI (command-line interface), or to be called from a file that is executed by the server. Create a .cfg file with the name of the map (i.e. de_dust.cfg) and enter the commands you want to execute along with that map. By way of example, let's say you wanted to restrict the AWM/P rifle on the map de_dust. You would create a plain text file called de_dust.cfg and inside that file you would simply put:

```
admin_command admin_restrictweapon 4 6
```

That would restrict the AWM/P every time the de_dust map is loaded, assuming you have the proper Admin Mod plugins loaded to perform weapon restrictions.

Q: My server crashes every 10 minutes/My server crashes when I use admin_csay.
This is because the length of the message you have entered is too long to fit on a line. Reduce the size of the message, or make it wrap around to a new line (using “^n”). For example:

```
admin_csay This is line one^nThis is line two^nThis is line three^netc...
```

Q: I run a Linux server and I have a Win32 compiled script? Why wont it work?
Linux and win32 are different platforms, and the Small compiler creates slightly different versions of the .amx files on each platform, so they are not cross-platform compatible. The uncompiled scripts (.sma files) are identical in structure, so you use any uncompiled text plugin file (typically has a .sma extension) that was written on either platform, but you will have to compile it on the same type of system your server operates on (e.g. Linux or Windows).

Where to Go for Help

Admin Mod is far from simple - we know that. While we have done our best to simplify things as much as we can, by it's very nature it will require some real work and understanding to install, configure and administer.

That said, you may need some help getting things running just right. Before you go running for assistance, please make sure you read this documentation. If you ask a question that can be answered here, you will likely be told to come right back here - not because we don't like you, but because it's already been typed out once here, and to do it all over again would be time consuming and tedious.

So, you've read the docs and still have problems? Help is available on the Admin Mod web site, by clicking on the "Forums" link. Please take the time to include this information when you request assistance, so that we can best help you:

1. Operating System (e.g. Windows98, SE, Win2K, Linux Mandrake 7.1, etc.)
2. Admin Mod Version
3. Server software version and type of server (listen, dedicated, etc.)
4. What game MOD you are running (cstrike, TFC, etc.)
5. Method of installation (script or manually)
6. Whether you are using plugin scripts (new in v2.50) or the old-style single script
7. What other mods/bots you have running
8. A detailed explanation of what you have tried so far, listed in clean paragraph format, one paragraph for each thing you have tried and what the result was
9. Any log/console output that illustrates what happens when you encounter the problem (attach a .txt file)
10. When applicable/requested, a copy of your server.cfg, users.ini, adminmod.cfg, etc. files, ATTACHED to your post in TXT format or ZIP'ed. In other words, look for the "Attach File" box below the window where you type your message, and use the "Browse..." button to attach the file, which must be renamed so it has a .txt extension (e.g. server.cfg.txt will work), or it must be ain a ZIP format.

PLEASE do NOT copy and paste your server.cfg or any other long text files into a message - an attachment is a much better option, because it preserves the content of the file exactly, and it keeps the forum a lot cleaner when it comes time to read the thread you are in.

If we can gather this core info on all problems, especially the ones that seem to affect multiple users, we can help you faster, as well as better catch problem patterns and causes. That, in turn, will help us to better provide you, the user, with better help documentation and a better mod down the road.