Admin Mod Version 2.50b

Documentation and Online Help
v2.50b.080401

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

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

Documentation Revision Information:
v2.50b.080401
--

New Admin Mod minor (fix) version, 2.50b
Corrected several errors in spelling, layout and content
Now supporting TFC
Added cross-links to several areas
Improved server.cfg setup information
Added missing default commands for CS to commands list

Table of Contents

What's New in version 2.50?

AdminMod 2.50 has a load of new commands making it more accessible and powerful. Some are obvious, such as the implementation of plugin scripts, and others are less so.

A partial list of changes to this version:

Plug-in Scripts!
You can now use multiple scripts. Yes, that is correct, you can use multiple scripts. Thanks to the plugin.ini file, simply list the path to the plugin script you wish to use. This feature can be 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 numerous other plugins. Check out the Metamod web site for more information.

admin_command
You are now able to enter commands directly into the hlds command line. 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 staring atthe HLDS console, just enter:

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

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

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.

Bye-bye old password_timeout
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, without having to re-enter a password.

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

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

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

admin_reject_msg
All rejection messages are easily configurable by setting this cvar. If somone tries to use an un-authorized comands, 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 AdminMod'" 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.

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

admin_disco
This cannot be explained, and neither can its counterparts. Go see it for yourself. :)

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 complete list of supported game mods appears below.

What is Admin Mod?

This mod 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. It is a plug-in replacement for the following mods:

- Judgement
- FireArms rc2.5
- DoD 1.1
- 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: (untested)
- Freeze: (untested)
- Bot: (untested)
- 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? (2.10/2.10a)
- OpFor:: Bang? (2.10/2.10a)
- Others: (Tell Alfred if it works, it shouldn't)

This is a server-side only mod, one that gives additional functionality to players specifically granted permissions to access the functions it provides.

However, 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.

As Alfred and the team have written this mod they has 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.

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 Admin Mod version 2.50 (released April 1, 2001 - and no, that's not a joke), and assumes that you already have a working Half-Life Server up and running.

We have not attempted to document how to install a HLDS system – This documentation explains how to install, configure and use the Admin Mod.

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

Geting_Ready_to_Install

Introductory Information

Setting up Half-Life Admin Mod isn't 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.

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.

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 suct as Microsoft Word or Wordperfect or anything along those lines. Most Windows server admins 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

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 helper 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 with game mods, the developers of helper applications like admin mod can focus on growing and expanding their programs' capabilities.

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 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 scripts to plug into 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.

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 "Getting Ready to Install 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 2.50 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
- etc.

So, if you have a server running on a Windows-based machine, you can successfully use the admin mod. We can’t guarantee the server will run well, 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:

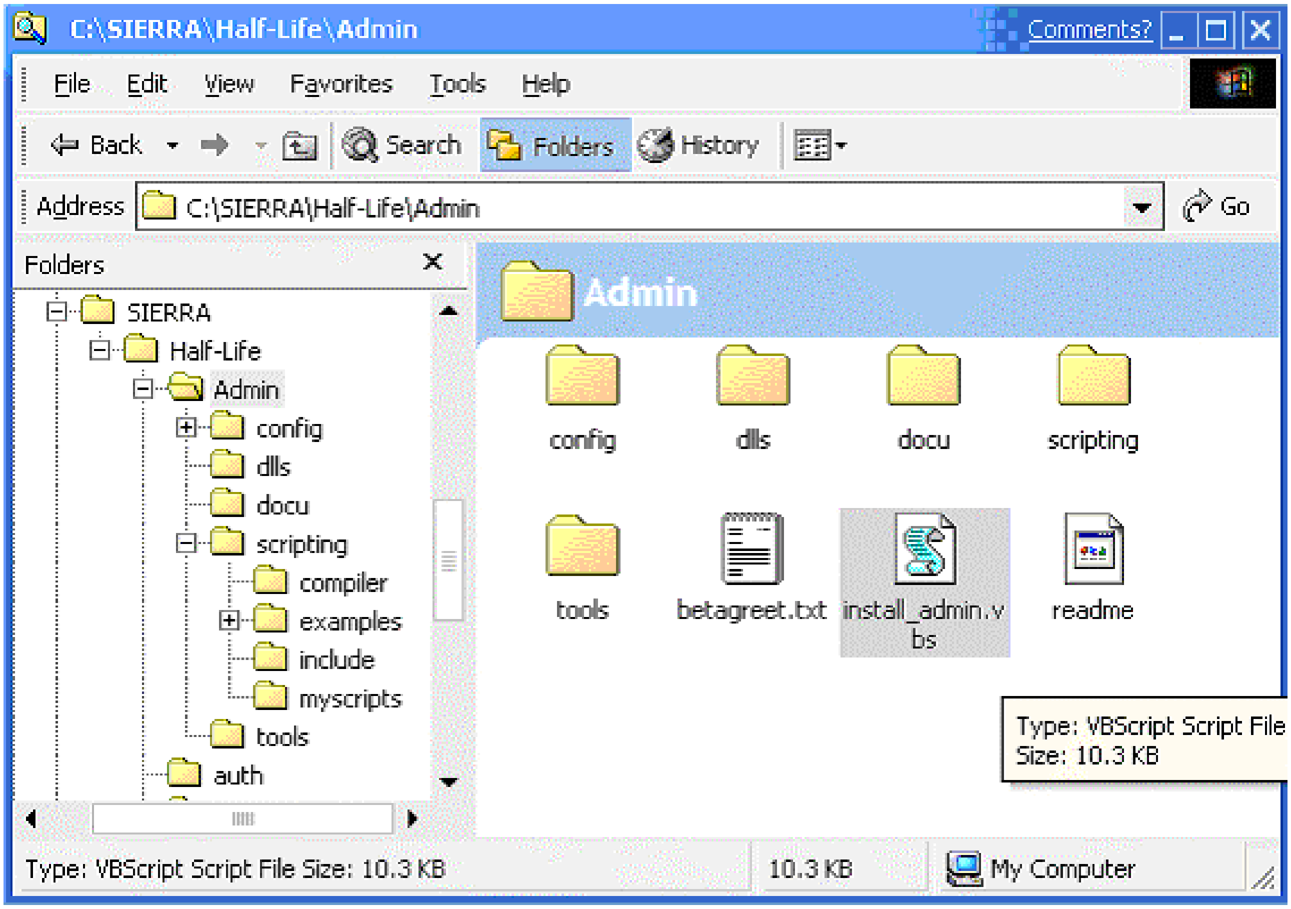
THE FIRST THING YOU SHOULD DO IS BACK UP YOUR CONFIGURATION AND OTHER CUSTOM FILES!!!

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.



Type: VBScript Script File
Size: 10.3 KB

Type: VBScript Script File Size: 10.3 KB

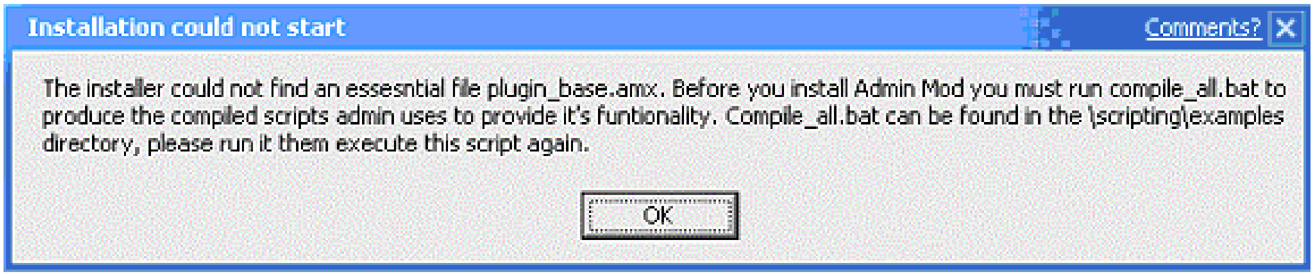
If you don't see these folders/directories after unzipping the file, you have done something wrong, so **STOP NOW** and save yourself the headache. You most likely did not use the folder names included in the ZIP file.

Next: Running the Windows Installation Script

Running the Windows Installation Script

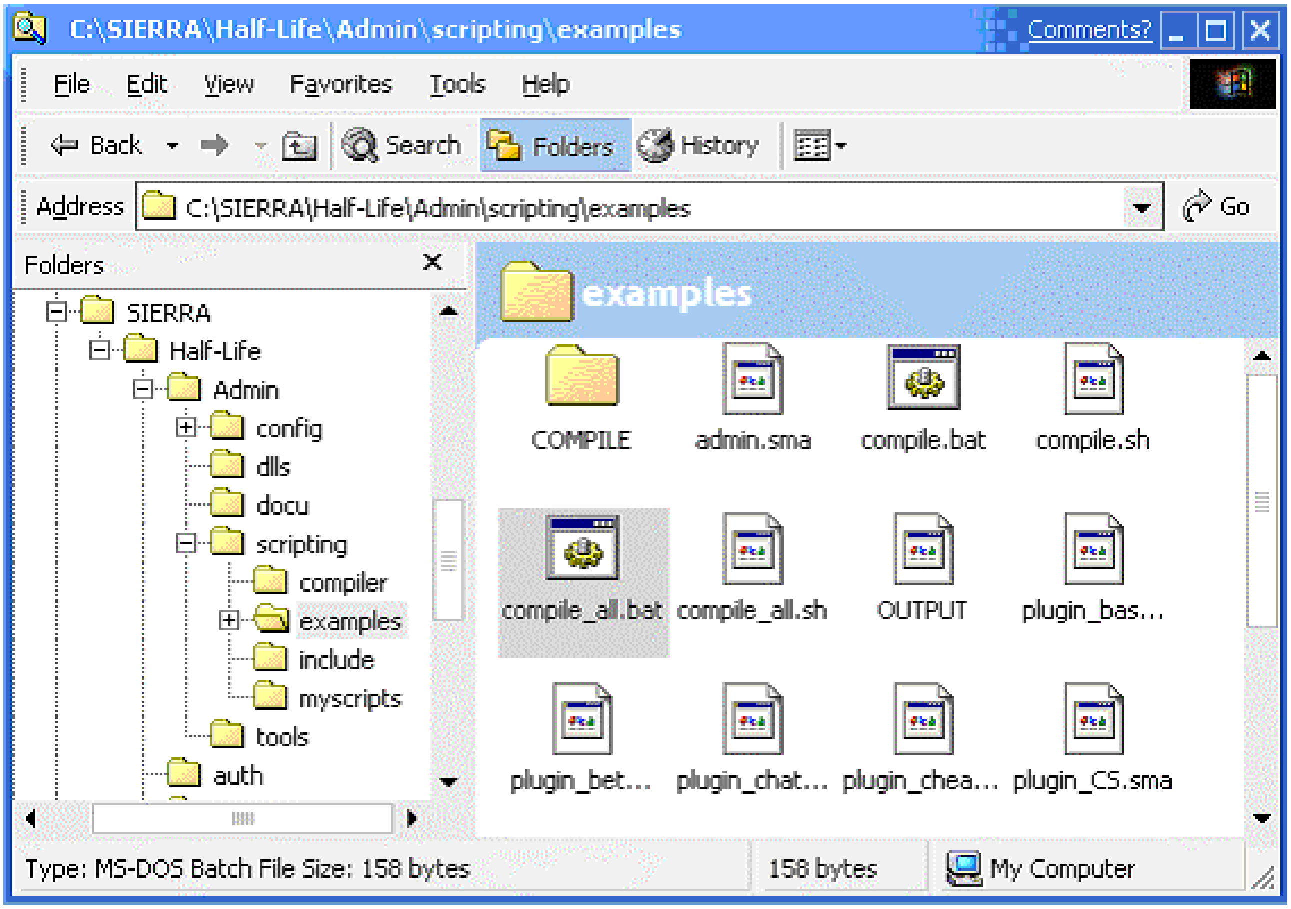
> ***NOTE:*** *If you experience difficulties running the installation script that is supplied with Admin Mod, it is possible you either have an outdated version of the Windows Scripting Host or you do not have it installed on your computer. In either case, the latest version of Windows Scripting can be downloaded from:*
>
> *http://www.microsoft.com/msdownload/vbscript/scripting.asp.*

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.

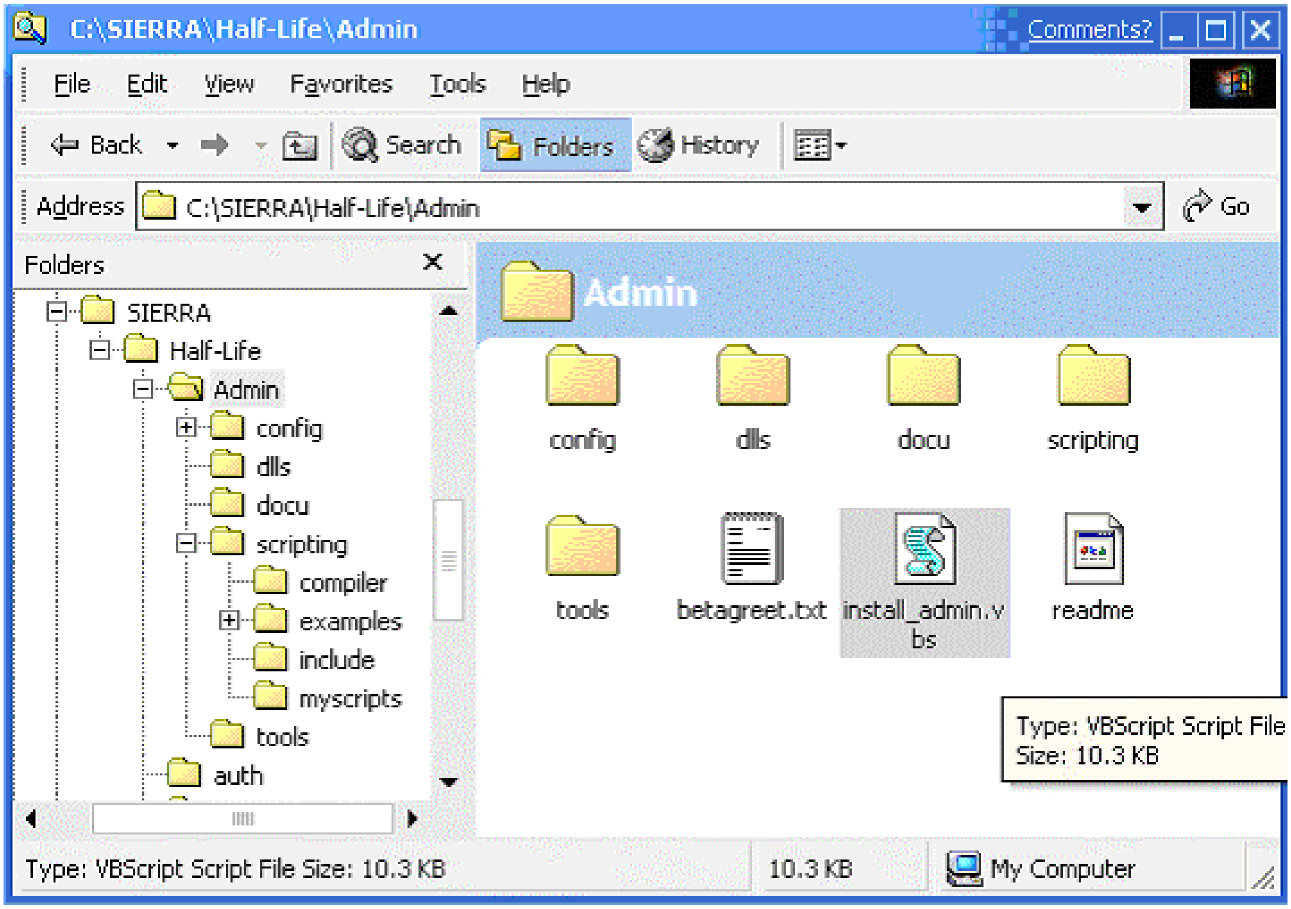

Installation could not start

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.

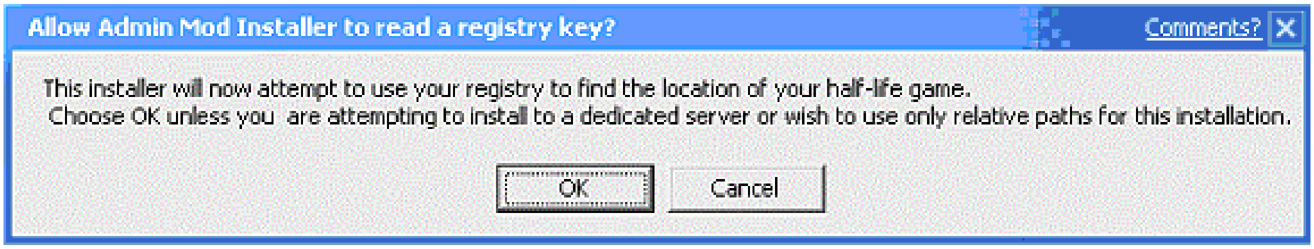
If you do not have the script files compiled ahead of time, or if they are not in the appropriate folder (\admin\scripting\examples), the above window will pop-up. As explained in the window, you simply need to run the file named compile_all.bat in the admin\scripting\examples folder. Doing this will - 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:

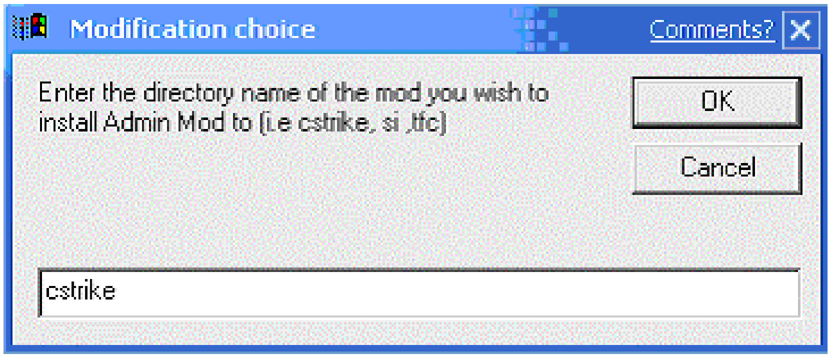


If you have a server that is set up without entering a cd key (dedicated server download), select cancel for the following pop-up. If you have entered a cd key (you are using the server that came on your Half-Life CD, even if it has been upgraded), you have the choice between either option. If you are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



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. The mod directory 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.

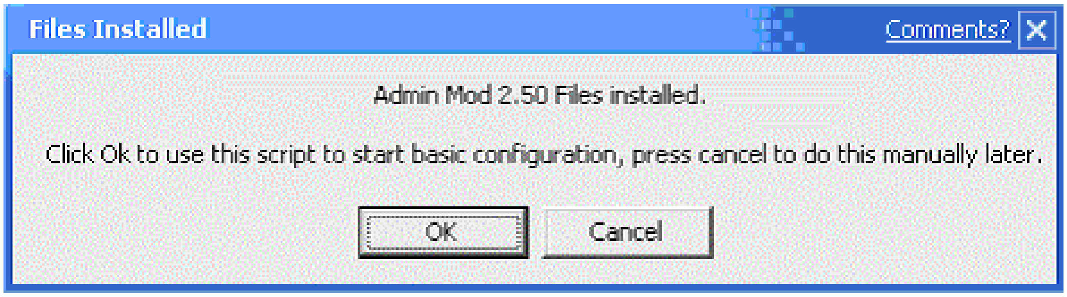


Modification choice

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 nicks.ini and sample autoexec.cfg files, which is nice, since that way you will have a sure-fire matching set of files.



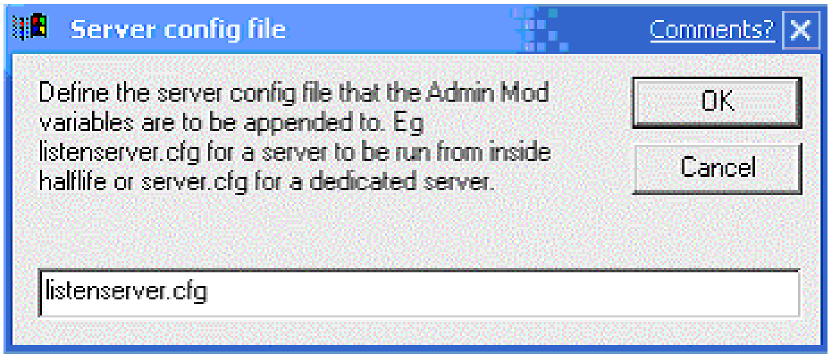
Files Installed

Admin Mod 2.50 Files installed.

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

The two windows shown below will determine which server configuration to use, dedicated or non-dedicated.

If you create your 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:

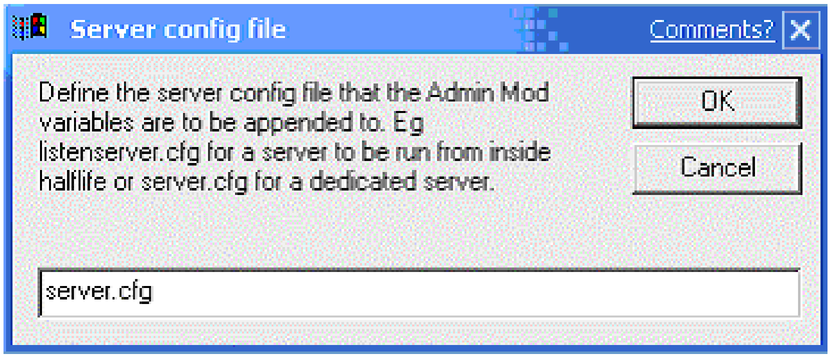


Server config file

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

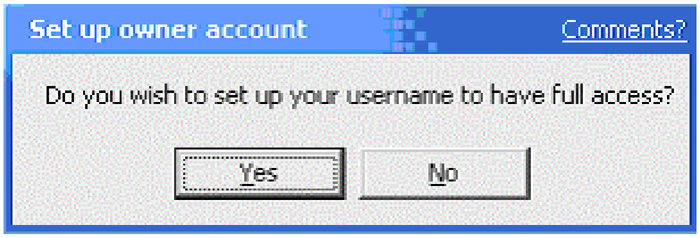
However, if you use a dedicated server (a DOS-like window opens and lots of data scrolls down when you start it), enter "server.cfg in the box pictured below:


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

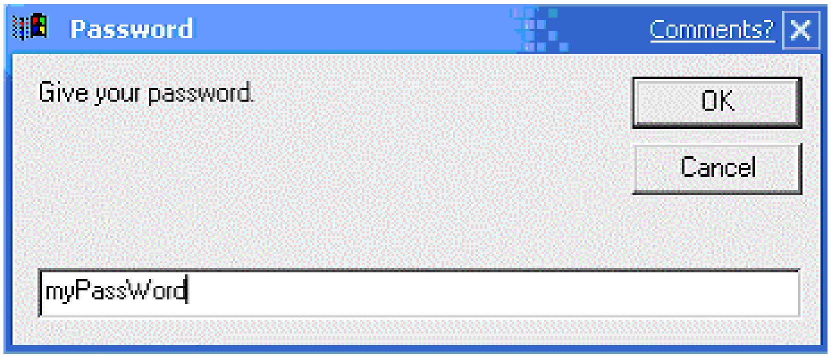
Depending on your setup, you may or may not be asked for the following information:


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 autoexec.cfg file in order to set up your password automatically if you play on this computer:

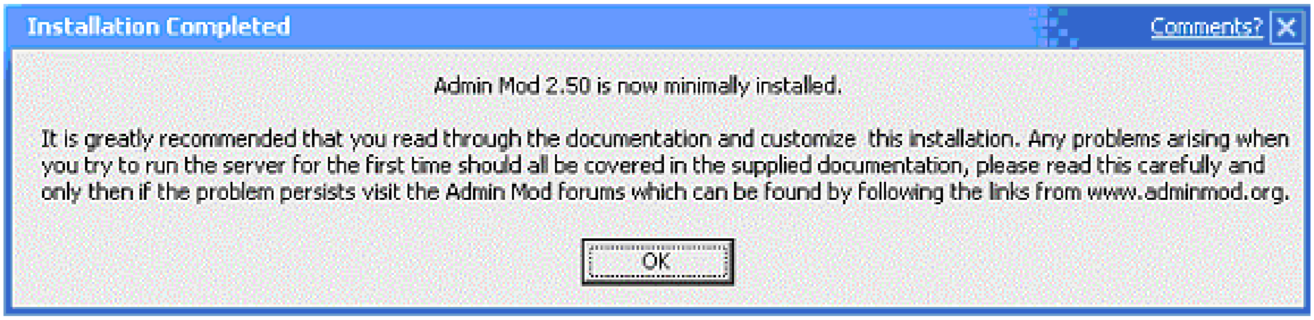
> *NOTE: If you are setting up a dedicated server, this file will need to either be moved or re-created on the computer where you play the game. This file is created on your server during installation, but it is not meant to stay there, unless you are running a listen server. Be sure to see the section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


Password

Give your password.

myPassWord

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



Installation Completed

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.

Not every problem can be found in this documentation, but we suggest strongly that you look here first. If you have problems you can't solve - and that happens - please ask a question in the forums at http://www.adminmod.org.

So what's Next???

You're not done just yet. You will need to set up the following files. You're a goodly portion of the way there! So what are you waiting for - Let's get started!

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

Manual Installation - Walk-Through

or, Snaller's Heavilly-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.

Well it’s fairly simple, and the good news is that if it goes wrong it’s very easy to return to your old working server.

Now, what I did was for CS under WinNT, but I imagine there isn't any great difference for TFC and Linux (if that's what you are using)

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

Now, 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 there into the dll 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 dlls doesn't upset the server at all (it doesn't 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 juts 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* of each of the 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. everything from the directories called "Admin/config" and Admin/config/samples."*

Now you need to edit some of the config files. Start by editing your main server.cfg file (this should be in your mod directory).

Suggestion: Scroll to the bottom of the server.cfg file and make a separating comment line, so you know where the admin 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 don't touch the server.cfg in the examples directory anymore, only the one in your mod directory).

Then you edit the file paths as 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 exension 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"

> **Old Skool (unsupported for development in the future)** -- ONLY IF YOU ARE USING A SINGLE OLD_STYLE SCRIPT, you need to locate the field called "script_file" (still in server.cfg) and make sure it's pointing to the right file, usually called admin.amx
>
> **Plug-In (supported model going forward)** -- 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"

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

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

The wordlist.txt file is a list of terrible words. This is your censor list, if you want that. If you add a word here people can't use that word when they chat (whole line is suppressed). 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 ip.ini file is if you want to give players reserved spots based on their IP number, examples are in the file. If you don't want anything to reserved, just delete all the lines.

The maps.ini file contains a list of maps that the players can vote for, one map per line (of course Counter-Strike has built-in voting, but this enhances voting greatly).

The users.ini file is where you put people who you want to give some kind of admin rights. You should at least put yourself in here and give your username max rights. Note that permission levels are slightly different now than they were in previous versions, and that there is no longer a file to reserve player nicknames. See "Setting up your users.ini file" in this manual for more information.

You plugin scripts are configured in scripts.ini -- Be sure to uncomment the TFC or CS lines if you are using those scripts, otherwise the commands and features from those scripts will not be available to you! You can also add compiled plugin scripts to your mod's dll directory and then specify them here 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 printout will show all the scripts 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. 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 set to point to the correct Admin Mod DLL files necessary for running AdminMod. 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.*

Up to this point, the program still is not in effect if you restart your server. One last thing is still missing:

In your mod directory there is a file called liblist.gam - make a backup of this now, so that if you ever want to remove the Admin Mod, all you have to do is put your backup file back in place.

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. (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 the setinfo commands for the client (game) setup in the section entitled "Client configuration via setinfo commands" in this documentation.*

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

Thanks to Smaller for providing the huge majority of this tutorial, which was adapted heavilly 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.

Linux 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.

Configuration

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.

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's of the people 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.

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 your server.cfg file

> **NOTE:** The server.cfg/listenserver.cfg file is the link between Half-life and Admin Mod. Without a proper setup, Admin Mod will cause more problems than it will solve. This 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 extensivly, 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:

```
// Team Balancing - TFC Only
// Used by the TFC plugin.  If enabled, an attempt to switch
// teams will be rejected if it would unbalance the teams
// (1 = turn feature on, 0 = turn feature off)

admin_balance_teams 0
```

```
// If you are using bots with adminmod you can set this to 1
// It will protect bots from recieving client commands which
// would crash the server.
// (1 = turn feature on, 0 = turn feature off)

admin_bot_protection 0
```

```
// This is the message displayed to everyone after connecting.
// It displays in plain text in the center of the player's screen.

admin_connect_msg "Welcome to the Real World..."
```

```
// This controls the availability of weapon restriction from the CS plugin script.
// 0 to disable weapon restriction, 1 to enable

admin_cs_restrict 0
```

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

admin_debug 0
```

```
// Determines whether or not the fun commands are allowed.
// Fun mode includes things like disco mode and ability to glow, etc.
// (1 = turn feature on, 0 = turn feature off)

admin_fun_mode 0
```

```
// Enable to get special effects with certain commands
// like teleport or slap.
// (1 = turn feature on, 0 = turn feature off)
```

admin_fx 0

// If enabled, people who are gagged will be unable to change
// their name while gagged
// (1 = turn feature on, 0 = turn feature off)

admin_gag_name 0

// If enabled, people who are gagged will not be able to use
// the say_team command.
// (1 = turn feature on, 0 = turn feature off)

admin_gag_sayteam 0

// Makes the admin with the highest access level the only admin in power.
// for example: if multiple admins are present, only the one with the highest access
// level will have admin access.
// (1 = turn feature on, 0 = turn feature off)

admin_highlander 0

// If admin_ignore_immunity is enabled, ACCESS_IMMUNITY is
// ignored and does nothing. In other words, if you set this to 1,
// the access level giving immunity from certain admin commands
// will have no effect and admins will not be immune. Set to 0 to
// use the immunity capabilities.
// (1 = turn feature on, 0 = turn feature off)

admin_ignore_immunity 0

// This file specifies which script plugins get loaded.
// It should be relative from the <mod> directory
// NOTE: EITHER THIS FILE OR THE "script_file" CVAR MUST BE DEFINED
// FOR ADMIN MOD TO WORK. If you are using the plugin scripts that were
// introduced in Admin Mod 2.50, then you must define this cvar to point at
// your .ini file containing the list of plugin scripts you want to enable (by
// default this is the plugin.ini file in your game mod directory).

admin_plugin_file "plugin.ini"

// The time 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."

// 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 by [ADMIN] <user> used command <command>

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.
// (1 = turn feature on, 0 = turn feature off)

admin_vote_autostart 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 by a vote of the players

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. Use a value of 0-100.

```
admin_vote_ratio 50
```

```
// Ability to make clients execute commands on their computers (in-game only)
// (1 = turn feature on, 0 = turn feature off)

allow_client_exec 0
```

```
// Default access rights for players not in the users.ini file.
// See Access Levels section of the documentation for more information

default_access 1
```

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

encrypt_password 0
```

```
// If enabled, the scripting file functions have read
// access to files
// (1 = turn feature on, 0 = turn feature off)

file_access_read 0
```

```
// If enabled, the scripting file functions have write
// access to files.
// (1 = turn feature on, 0 = turn feature off)

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. A percentage value, from 0-100.

kick_ratio 60

// Ratio of players who must vote 'yes' to a map change
// for it to be successful. A percentage value, from 0-100.

map_ratio 80

// List of maps people are allowed to vote for. (Typically maps.ini)
// This file takes the same format as the mapcycle.txt file.
// Disable to enable all maps. 0 to disable.

maps_file 0

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

models_file 0

// 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 pasword 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.
// Turn this one for the fade-in color text in the center of the screen
// (1 = turn feature on, 0 = turn feature off)

pretty_say 1

// 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 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 0

// This controls how reserve slots work on the server.

// reserve_type 0

// This cvar allows you to use the older-style single admin scripts.
// NOTE: If you have admin_plugin_file set, 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.
// (1 = turn feature on, 0 = turn feature off)

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() 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 "words.txt" if enabled

words_file 0
```

Setting up your users.ini file

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:password1:131071
Gerg:password1:65535
Bud-froggy:password2:65535
```

```
        12345678:password3:199
```

This would give PlayerX an access level of 131071 (all commands) if, in his setinfo line, he had the appropriate passwords. For more information, see Client configuration via setinfo commands

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 an appropriate setinfo line.

You can put as many 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 at will while at home, and he can use his "normal" name 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. Here is a friendly little calculator that will do the work for you (if you are reading the HTML docs, 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.

Access Level	Commands Allowed
all players (public commands)	admin_cancelvote admin_denymap admin_startvote admin_listmaps admin_nextmap ambt_greeting admin_messagemode admin_nomessagemode say currentmap say nextmap say timeleft admin_timeleft

	admin_userlist admin_version
1	admin_vote_restart say mapvote say rockthevote say vote <map> admin_vote_kick admin_vote_map
2	admin_restartround say cancelvote say denymap admin_fraglimit admin_map 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_execall admin_execclient admin_execteam 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

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), see the Access Levels page in the Admin Mod Reference section.

Note that if you give 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 for each level. 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.

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 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. DONT mix plaintext and encrypted passwords!

This is an example of an ecrypted password

```
barney:HFwILz6hzetcs
```

Setting up your maps.ini file

This OPTIONAL file takes the same format as your server's mapcycle.txt:

```
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.

Setting up your plugin.ini file

The plugin.ini file is a new addition in AdminMod 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.

In the stock/default plugin.ini file 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 will be lines in the file other than the ones shown above.

The default installation retrieves files from the plugin.ini file. If you wish to use only one script, disable the plugins by setting admin_plugin_file to 0. The script_file location must then be specified by the script_file cvar.

See also: Admin Mod Configuration Variables (CVARs)

How to Use Bots with Metamod (AM 2.50)

Metamod is designed to give you the capability to install various plugins that interact with your Half-Life server, for example Admin Mod or Bots. An application has to be specifically written as a Metamod plugin to be used with metamod.

As Metamod only came out with the release of Adminmod 2.50, there are no bot plugins for Metamod available at the time of writing. You can use bots with metamod nevertheless. If you used bots with an earlier vesrion of Admin Mod the procedure will be nothing new to you.

To use a bot (or other server mod) with Metamod that is not designed as a Metamod plugin, you should first install the bot. The bots installation will change the liblist.gam file in your mod's directory. That is why you should install the mod before you install Metamod/Admin Mod, as it will overwrite the changes made by the AM/MM installation. Then install Admin Mod/Metamod. The installation will alter your liblist.gam file by exchanging the gamedll line with on that points to the Metamod DLL. Now your server should run with Metamod.

To get the bots working, create a new text file and name it metagame.ini -- In this file you put one line with the location and name of the bot's DLL, e.g. if you installed the HPB_Bot for Counter-Strike and the bot's DLL is called hpb_bot.dll, the DLL will be in the cstrike\dlls directory.

Then the metagame.ini file needs to have this line in it:

```
dlls\hpb_bot.dll
```

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 metamod 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.

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:

```
               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
   install this version of Adminmod 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 Adminmod lines to your server.cfg file? (y/n) [y]

   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 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:

```
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!
```

Setting_up_Clients_to_Use_Admin_Mod

Client configuration via setinfo commands

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 because it requires some knowledge of 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 file called autoexec.cfg was probably created in your mod directory on the server. You will now be moving that file to your client machine (the computer that you play your game on), unless you are running a listen server, in which case your client and server are on the same computer, so the file can stay right where it is. The autoexec.cfg file contains information similar to the information we will discuss here on this page.

Basically, when you connect to the server, the Admin Mod looks at your name that you have specified in your game settings. If your name is listed in the users.ini file, 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 does this, in simplified terms, is like this: Once it 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 specifies a place that it expects the password to be stored, sort of like a mail box. Let's say your mailbox is called "pw-pass" and you have specified that in the server configuration file. If that is the case, the server is going to send your game a message requesting anything that is stored in the "mailbox" (it's actually a variable) called "pw-pass".

So, you need to store your player password in the field called pw-pass 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, it is a variable for storing data. In this case that data is your Admin Mod password.

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

```
setinfo "pw-pass" "password"
```

with "pw-pass" being the mailbox variable defined by the password_field setting in the server.cfg or listenserver.cfg, and the "password" being defined by the permission line in users.ini for your player name or WONID.

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

```
PlayerX:testpass:65535
```

and the (listen)server.cfg file had:

```
password_field "pwd-admin"
```

the setinfo line for PlayerX would be:

```
        setinfo "pwd-admin" "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, if correct, give the player access to your server. Once again, in our mailbox metaphor, the mailbox name (field where info is stored) is called "pwd-admin" and the user's password is "testpass".

The setinfo line can be specified in many places. However, we recommend one easy way to do it. 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\Hlaf-Life\tfc
```

You autoexec.cfg 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 need to edit this file with a plain text editor such as notepad:

```
setinfo "pwd-admin" "password"
developer 1
echo [ADMIN] password has been set
developer 0
```

Make sure you replace "pwd-admin" and "password" in the example above with the appropriate information from password_field and the users.ini. Do not remove the quotation marks - you need to keep those.

In the autoexec.cfg file, which should be located in the mod directory on the client machine (make a plain text file with this name if it does not exist), add this line:

```
exec adminpass.cfg
```

If all the passwords are in the right place and are, of course, the correct passwords, 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.

Scripting_Basics

A Beginner's Guide to Scripting

A SCRIPTING CRASH-COURSE

This is not intended to be a complete guide to scripting 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 Newbies a possibility to learn the basics of scripting and compiling, which are the basics of coding in general. For information about compiling completed scripts, 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 C/C++. This section takes for granted that you know basic programming of some sort.

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 put:

```
new iValue = 0;
```

If you wanted to declare a string, you would put

```
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, the one thing that you can ALWAYS edit it what is between quotes. It 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 (//) comments out 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:

```
/*****************************
AdminMod 2.5
*****************************/
```

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

```
/*
****************************
AdminMod 2.5
****************************
*/
```

Note, that /* */ comments CANNOT be nested, i.e. you should not place comment into comments. It is safe to have //-sytle 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 FooFoo";
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 andloops 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

A BASIC COMPILING TUTORIAL

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.

First of all, the scripts that you write or make changes to are plain text files with an extension of ".sma".

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 file that Admin Mod uses. Once compiled, there is no way to retrieve the original script, so do not delete your *.sma file. Keep it there just in case. The 2.5 release offers unprecedented ease of use. Everything is already set up for you.

To compile your own personal script with all of your added snippets and so on, simply place the new plugin_myscript.sma file in the admin\scripting\myscripts directory. Note that your file name must end with.sma for this to work.

Run the compile_all.bat - This nifty program looks for scripts you have dropped in the "myscripts" folder (again, it expects *.sma) and compiles them, and then places them in the "binaries" folder. After compiling, just look in the "binaries" folder for your compiled 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.

Disregard any "Warnings" in the text that is kicked out to 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 *.amx file when it encounters errors (or if it does, the .amx file will have a length of 0 bytes), so you have to go and find the errors in your script and fix them.

Remember that errors can be due to faults in lines that come before and after the one indicated. The line number indicated in a error message is simply where the compiler could not make sense of what was going on, but the problem that stopped it there could have started somewhere else in the script. The most common errors are spelling, loose/missing/extra braces, and undefined variables.

One loose brace 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 /dll 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

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

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

admin_cancelvote: Cancels the current hlds_ld vote

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

admin_chat <msg>: Shows message only to other admins.

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

admin_denymap <map>: Removes all votes for map.

admin_disco: Starts disco fever. Fun mode only.

admin_enableallweapons: Allows all weapons to be purchased. Use to remove all active weapon restrictions.

admin_enableequipment: Allows all equipment to be purchased. Use to remove all active equipment purchase restrictions.

admin_enablemenu <menu #>: Remove restriction from specified menu.

admin_enableweapon <menu #> <weapon #>: Remove restriction from specified weapon.

admin_execall <command>: Force everyone to execute command.

admin_execclient <target> <command>: Force target to execute command.

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

admin_fraglimit <fraglimit>: Sets the mp_fraglimit cvar.

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

admin_fun <"on" | "off">: Turns fun mode on or off.

admin_gag <target> [<minutes>]: Gag target. 0 minutes is a permanent gag.

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

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.

admin_map <map_name>: Changes map.

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

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

admin_nextmap: Shows next map in cycle.

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

admin_nomessagemode: Will treat 'say' as 'say'.

admin_nopass: Clears the server's password.

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.

admin_rcon <cmd>: Executes rcon command.

admin_reload: Reloads Admin Mod files.

admin_removespawn <identity>: Removes a spawned item.

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 message from you as admin.

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

admin_slap <target>: Slaps target.

admin_slay <target>: Slays target.

admin_slayteam <team>: Slays everyone on team.

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

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

admin_stack: Will stack everyone on top of you.

admin_startvote: Starts an hlds_ld vote.

admin_teamplay <teamplay>: Sets the mp_teamplay cvar.

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

admin_timeleft: Shows the time left.

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

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

admin_unban <WONID or IP>: Unbans target.

admin_ungag <target>: Ungag target.

admin_unllama <target>: Unllama-fy target.

admin_unpause: Sets the pausable cvar to 0.

admin_userlist [<name>]: Shows a list of users.

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

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

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

admin_vsay <question>: Presents question as a vote.

admin_ungag <target>: Ungag target.

admin_unllama <target>: Unllama-fy target.

admin_unpause: Sets the pausable cvar to 0.

admin_userlist [<name>]: Shows a list of users.

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

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

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

admin_vsay <question>: Presents question as a vote.

say rockthevote: Starts an hlds_ld vote.

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

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

User Access Levels

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

To use the calculator (web 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_cancelvote admin_denymap admin_startvote admin_listmaps admin_nextmap ambt_greeting admin_messagemode	admin_nomessagemode say currentmap say nextmap say timeleft admin_timeleft admin_userlist admin_version
□	1	admin_vote_restart say mapvote say rockthevote	say vote <map> admin_vote_kick admin_vote_map
□	2	admin_restartround say cancelvote say denymap	admin_fraglimit admin_map 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_restrictallweapons admin_restrictequipment admin_restrictmenu admin_restrictweapon

		admin_blue (TFC) admin_green (TFC) admin_red (TFC) admin_yellow (TFC) admin_enableallweapons admin_enableequipment admin_enablemenu admin_enableweapon	admin_weaponscheck admin_fun admin_disco admin_execall admin_execclient admin_execteam admin_llama admin_unllama admin_listspawn 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	

Calculate Levels **Access Level for users.ini:** □

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 <mod> directory; if I'm running a TFC server, my <mod> 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 passwording 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 undefine 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 commnds 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.

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_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 (eg, 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_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 every ten minutes.

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 messages, when used, will always show the message 'ADMIN Command: <Player> used <command>".
* If admin_quiet is 0 or disabled, other commands will show the message 'ADMIN Command: <Player> used <command>'

* If admin_quiet is 1, other commands will show the message 'ADMIN Command: Admin used <command>', but will not name the admin.
* For any other value of admin_quiet, other commands will not display a message at all, but will only get logged.

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

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 hlds_ld-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 hlds_ld-style votes; for the HL menu-style votes, see vote_freq.

admin_vote_maxextend <#>

Used by the hlds_ld map vote plugin. Controls how many times the current map can be extended for thirty minutes (eg, 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 hlds_ld 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 hlds_ld-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 (eg, 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.

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 relatived 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. Eg, if 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. Eg, 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 hlds_ld-style map vote; see admin_vote_ratio.

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 'pwd-home', and there is a user whose password is "milk", then the setinfo line in the user's autoexec.cfg would be 'setinfo "pwd-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 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. Otherwise, this is the file (relative to the <half-life> dir...NOT the <mod> dir) 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> dir) 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 hlds_ld-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.

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

Frequently Asked Questions (FAQ)

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 pretty_say (pretty_say 1). 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."

I am runing a Counter-Strike Server. After I installed Admin Mod, I can't seem to use the weapon and menu restriction commands? 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.

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

Where can I find a list of commands?
Check the Admin Mod Reference section of the documentation.

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"
```

When my server changes maps, all the clients get "dropped." What the #&^$%?
This is not Admin Mod's fault. If you want to point fingers, point yours at that heap of metal under your desk. Admin Mod uses very little system resources (0.1%), so do not blame your computer's shortcomings on Admin Mod.

56k-ers lag my server. How can I ban them?
Sorry, but there is no way to ban HPB. AdminMod 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.

The map returned my nextmap is not the next map in the cycle. How come?
Admin Mod uses the maps.ini file for its map cycle information. Half-life refers to mapcycle.txt, so if the two files are not identical, Admin Mod and Half-life will not be in synergy.

How do I uninstall Admin Mod?
You don't. :D Just kidding. Remove the Admin Mod specific CVARs from your server.cfg or listenserver.cfg (whichever one applies to your situation) and 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 - hopefully you were a good scount and made a backup of your original files just in case..).

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.

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. It can be done on Linux servers with an application called hlds_ld.

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.

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.

How can I make map specific configurations?
AdminMod 2.5 has a new command that can be directly entered into the CLI (command line interface). Create a cfg file with the name of the map (i.e. de_dust.cfg). By way of example, simply write in:

```
admin_command admin_restrictweapon 4 6
```

This will restrict the AWM/P.

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

I run a Linux server and I have a win32 script? Why wont it work?
Linux and win32 are different platforms, and they therefore compile differently. The uncompiled scripts are the same, so you can get one that was written on a win32 machine, but you will have to compile on a Linux machine. The same applies in reverse, by the way.

Index