

The Official Admin Mod Help and Installation Guide for Newbies and Hard-Core admins, too

Compiled from the vast resource and knowledge of admin mod users from around the world

ADMIN MOD VERSION 2.07

DOCUMENT DATE: OCTOBER 22, 2000

Welcome to Admin Mod

Admin Mod is a modification add-on for the Half-Life Dedicated server, commonly used for CounterStrike 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 this document covers (and what it doesn’t)

This documentation assumes that you already have a running 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.

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:

- Phineas (v0.21): The normal Half-Life mod with bots!
- Firearms (rc2.3): If you haven't played it you must!
- Golden Eye (1.5): You loved it on the N64, now its on the pc!
- Counter-Strike (b7.1): The cool counter terrorist mod
- Action Half-Life: (untested)
- Freeze: (untested)
- Bot: (untested)
- TFC (v1.1.0.0): The Oh High-N-Mighty Team Fortress
- Valve (v1.1.0.0): 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 (v0.46): Join the battle on the front lines.
- WizardWars (beta 2): Zap, kapowi, spin up a spell.
- GangstaWars (beta 2.5): Become the don!
- Arg! (beta 0.5): Become a pirate!
- 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. As Alfred has written this mod he 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 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.

Introduction

Setting up Half-Life Admin Mod isn't difficult or tedious; it just requires some understanding of what's going on. It actually may 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.

Here's a quick description of what each of the admin mod files are for:

users.ini

The users.ini file is used to specify who will have special rights on your server (by name) and what commands do they have permissions to use.

nicks.ini

The nicks.ini file is basically used for name reservation. It prevents players from using the specified names unless they set the correct password. It makes sense to add all the player names that are in the users.ini to this file also. It's like added protection. Just make sure that you set the same password for the corresponding name or problems will arise.

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.

models.ini

The models.ini file is basically used for model reservation. It prevents players from using the specified model unless they set the correct password. It lets you protect your clan's model from other unauthorized players using it.

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.

There is also an admin.ini file but it is usually not needed (read, DON'T have it).
If you are using a non-supported mod or auto detection fails, contact Alfred.

Default Basic Installation of Admin Mod – Windows 32-bit

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):

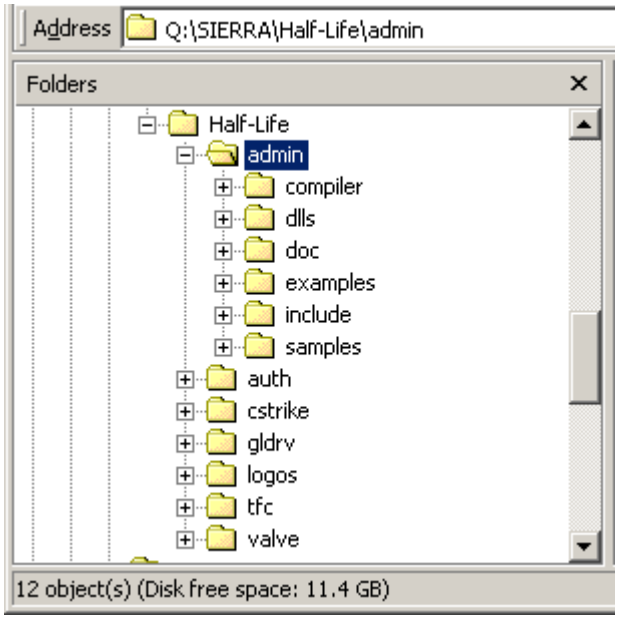
Windows 95
Windows 98
Windows NT4
Windows ME
Windows 2000

So, if you have a server running on a Windows-based machine, you can successfully use the admin mod.

For 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.

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

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 at right. 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. See above.



What's in those folders? What are they all for? Where does everything go? Well here is a brief description of what you are looking at:

admin - This is the main folder containing the entire admin mod distribution, comprised of several files and subfolders. In here you will find the Windows installation script (install_admin.vbs) as well as the admin help files and a few other files. Don't worry about any of these yet – we will get to the files and configuration I a bit.

compiler – This folder contains the small compiler program. You won't need to worry about this for a standard installation, but just so you know what it's for, this is the program that is used compiler custom script files that you can use to modify the

way admin mod works. Pretty cool, but not important for now.

dlls – This folder contains some of the files used to run the admin mod. The script we will run in a minute will take care of copying these files to the proper locations for you.

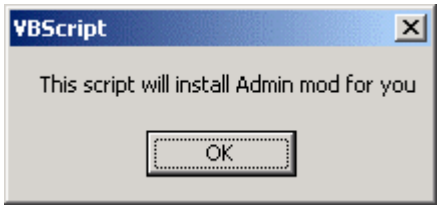
doc – This folder has a few text files that describe the mod, and how to do things with it. There is a file with a list of available commands, how to configure the mod, a set of frequently asked questions, some information on using admin mod with mySQL (for all you database-types out there) and a file that talks about writing scripts for the mod.

examples – This directory contains a copy of the default script, which can be modified and then compiled to customize admin mod. We really don't want to mess with this until we are sure we have a working mod. Once your admin mod installation is verified to be operational, you might want to take a look at the scripts (they are text files) and join the online forums to learn about how to customize the mod for your server.

include – Every time you compile a custom script, there are certain library files that are sucked up and included with the main script. They live in this folder. You won't want to mess with these files unless you're a true power programmer type.

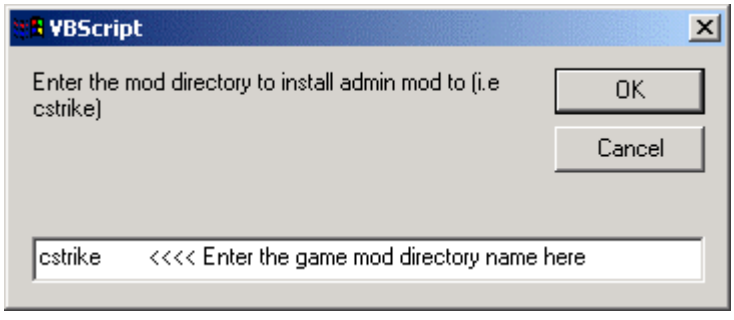
samples – We will be coming back to this folder in a short time, because it is where we can find examples of how to set up our server.cfg and a few other configuration files required by the admin mod.

Running the Installation script



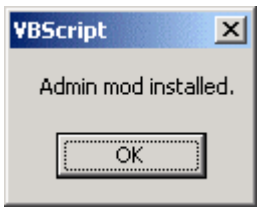
Okay, we have our ZIP file unzipped and everything looks like it has been described above (if it doesn't, go back – something is wrong).

Now it's time to run the *admin_install.vbs* script, which will copy our admin mod files to the proper locations and make certain changes to the game environment. There is one question you need to know the answer to before you start, though:

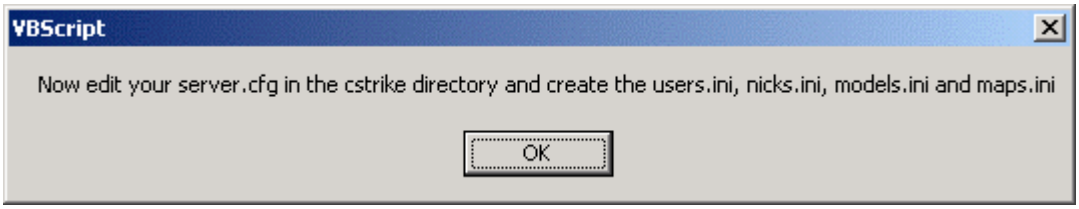


What game are you installing this mod on? *You need to know the directory name of the game mod.* For example, if you are installing the admin mod to work with counterstrike (and you have a default installation of counterstrike), the folder name is **cstrike**, and it is located in the half-life directory (the same directory the admin folder is located in). When we run the script, we will have to enter that exact folder name into a text box that will pop up on the screen.

Once you have entered the mod name and clicked the OK button, and if all goes smoothly, you will be presented with a happy (although somewhat misleading) box telling you that the mod has been installed. Technically this is correct – It is installed, but is not yet ready for prime time; so let's continue on.



The mod still needs to be fully configured before you can actually use it on your server. You will need to work with a set of configuration files before you can use the mod. The next box that pops up lists the files that we need to be concerned with configuring before we can actually us the admin mod.



server.cfg – This file already exists in your cstrike directory, and if you are already running a server, then you certainly are familiar with this file.

Note: IF YOU ARE RUNNING A LISTEN SERVER, YOU NEED TO EDIT THE LISTENSERVER.CFG FILE in place of the server.cfg file. We will be referring only to server.cfg here, so if the listen server configuration file applies to you, make these changes to that file.

Users.ini, nicks.ini, models.ini and maps.ini – Each of these will need to be created in the cstrike directory, as they do not exist there yet. Don't sweat it – it's easy.

Setting up server.cfg and the .ini files

It's time to look back to the *admin\samples* directory. In that folder 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.

DO NOT copy the server.cfg to you cstrike directory – it contains only the admin mod portions that you will need to copy into your existing server.cfg.

Now is the time to copy all of the .ini files and the wordslist.txt file to your game mod directory (i.e. for CounterStrike you copy these to the "cstrike" directory). If you have previously installed admin mod and do not want to lose your configuration files, be sure to back them up before copying! Once you have placed copies of the files in your game mod directory, forget about the ones in the samples directory – you should make all your changes to the copies that you just placed in your game mod directory.

It is recommended that you **open the server.cfg** in the samples directory and copy the contents, then paste them into your existing server.cfg file from your game mod directory (i.e. the one on cstrike if that is the game you are installing the mod for). Most people paste these lines of code in at the bottom end of their server.cfg file so they can keep track of it there.

The settings from the sample **server.cfg** you just copied will work in most cases for a default installation of your mod on a Windows machine, meaning the mod is installed in the half-life folder (i.e. c:\SIERRA\Half-Life\cstrike\ ) Read through the comments in the file to determine if the default settings will work for you – they are pretty self-explanatory. There is also a table of all server.cfg cvars at the end of this document.

Here are the contents of the server.cfg example file with a brief discussion of what each line does:

You need to tell the mod what file contains the user information for your administrators:

```
// users are in this file
users_file "users.ini"
```

How many seconds must pass after a vote has taken place before another vote is allowed to start?

```
// how often can each vote happen
vote_freq 300
```

What percentage of the players on the server have to vote in favor of a map change or to kick a user?

```
// 50% yes to change maps
map_ratio 50

// 80% yes to kick
kick_ratio 80
```

What is the name of the file containing reserved nicknames for your server? (useful for clan name reservations. See information that follows regarding similarities and requirements for using this file in combination with the users.ini file)

```
// reserved nicknames in the file
nicks_file "nicks.ini"
```

How many player spots should be help open for players with permission to take a reserved spot on the server?

```
// reserve 4 slots for people with this flag define in users_file
reserve_slots 4
```

The following line will allow you to make admin mod spit out all kinds of useful information in the event that things are not working as smoothly as you might like. You uncomment the line to make it work – to do that just remove the two forward slashes (“//”).

```
// print out debug info, use if you have a problem
//admin_debug 1
```

If this variable is set to 1, all players on the server will be allowed to use the admin_vote_map and admin_vote_kick commands.

```
// let anyone use admin_vote_kick and admin_vote_map
default_access 1
```

You can password protect player models in the file listed in this next option.

```
// list of model passwords
models_file "models.ini"
```

This next field is the one that stores the password on the client machine for access to models. It works pretty much the same as the password_field, above.

```
// the setinfo field the client must store the password in
models_password_field "pw-model"
```

A nifty custom message sent to the client console when they try to access a reserved model:

```
// tell clients this when they have an incorrect password
models_kick_message "go away, model reserved"
```

By default, leave reg_ex off, just like it is below. It’s the ol’ if you don’t need it, don’t use it rule… Regular Expressions are now supported under Windows beginning with admin mod v2.06 – 95% of password problems can be resolved by setting this as described here:

```
// use regular expression to match names ( 0=off, 1=on) (if you
have pw problems set this to 0)
use_regex 0
```

Here is where you define of the name of the file that will contain the maps that can be voted for using the admin_vote_map command:

```
// list of maps people are allowed to vote for. Comment out to
enable all maps
maps_file "maps.ini"
```

If you have a default installation of CounterStrike, this will already be set up correctly for you. If you have another mod or have CounterStrike set up in some unusual fashion, you will need to edit this line, or your mod won’t work. Also, Linux users need to comment out the Windows line and uncomment the Linux line:

```
// Change this to point to your version of admin.amx
// (i.e. change cstrike to the mod you are using)
script_file "cstrike\dlls\admin_win32.amx"
```

```
// for Linux use this file
// script_file "cstrike/dlls/admin_linux.amx"
```

The name of the admin mod help file and its location:

```
// where the help file is located, Linux guys flip the \
help_file "admin_help.cfg"
```

Only Linux people need to worry about this next line

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

Leave this on if you have not seen it yet – it makes for a nice fade-in message every so often in the game (and you can customize it too if you like):

```
// do the pretty centersay method :)
pretty_say 1
```

The name of the file containing bad words you don't want people to be able to say on your server. Usually vulgarities:

```
words_file "wordlist.txt"
```

A security option for you – if you are not sure, just leave it as it is.

```
// let scripts read and write files
file_access 1
```

Starting with v2.07, any given user's admin password will time-out after the number of seconds listed below – that means they have to re-enter it. This was added as a security measure.

```
// the amount of time a users admin_password entry remains valid
(default = 30mins)
password_timeout 1800
```

Formats of the .ini files:

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. The format of the users.ini like this:

username:password:access

Each user is listed on a new line. Passwords are case-sensitive, usernames are not. Here is an example of how one line of the file might look:

```
JoeUser:PaSsWoRd:127
```

That line defines a player names “JoeUser” who is assigned a password of “PaSsWoRd” and an admin user access level of 127. That last item – the user access level – deserves some explanation, since it can be a bit confusing how to come up with the right number, and what it all means.

In admin mod, there are different classes of permissions, which means that as the server owner you can give access to only those commands you want to allow, and you can pick and choose the commands available to individual users of the admin mod. That’s pretty cool – if you have an admin who seems to be abusing his ability to ban people from your server, you have the ability to remove that person’s ability to ban players, while still allowing him to use other commands.

You determine a user’s permissions by adding up the access levels for al of the commands you want that person to be able to use on the server. The chart below shows the access levels, and the associated admin mod commands.

access	commands that can be used
all players	admin_help admin_userlist admin_version admin_timeleft
1	admin_vote_kick admin_vote_map
2	admin_map admin_timelimit admin_fraglimit
4	admin_prematch admin_reload
8	admin_pause admin_unpause
16	admin_pass admin_nopass
32	admin_teamplay admin_friendlyfire
64	admin_say
128	admin_kick
256	admin_ban admin_unban
512	admin_cfg admin_status admin_rcon
1024	
2048	
4096	
8192	
16384	
32768	allow this user to use a reserved spot

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.

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_status and admin_rcon, 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.

The **nicks.ini** file format changed with the release of v2.07 of the admin mod, in order to improve security. The nicks.ini file is simple; all you have to put in it is a list of players and Won ID's of those who don't want others using their names. The basic concept is "I password protected my name". That's all, its not that hard. Here is the format of the nicks.ini:

username:wonid

Where **username** is the players name and WonID is their WonID. You can have more than once instance of **username** with different WonID's so people can use multiple computers to play with a name.

What's the WonID? It's the unique (almost) identifier assigned by the WON gods to identify a player. In reality, any given player can be assigned two different WonID's depending on which of the two WON servers they authenticate off of prior to joining your server. The server logs WonID's, so you can look there to determine your WonID and those of your admins and friends.

The **ips.ini** file is very simple also; it just specifies what IP addresses have access to take up one of the reserved slots. IP address are defined in the IP_ADDRESS/NETMASK format:

192.168.1.0/255.255.255.0

which allows 192.168.1.0 through 192.168.1.255 to have access.

The **models.ini** file is simple; all you have to put in it is a list of models and passwords of those who don't want others using their models. The basic concept is "I password protected my model". That's all, it's not that hard. Here is the format of the models.ini:

model_name:password

Where **model_name** is the name of the model to protect (i.e. "sas") and **password** is the password used to protect that model (same as users.ini).

You can use this function to reserve your clan's models only for use by clan members on your server.

The **maps.ini** file contains a list of maps that players are allowed to vote for. These maps might be ones that are not in your mapcycle.txt, for example. Similarly, you might have maps in your rotation that you don't allow people to vote on. It's a really, really simple file, and uses the exact same format as your server's mapcycle.txt file. Here is the format:

map_name

Where **map_name** is the name of the map which users can vote for with admin_vote_map (i.e. "2fort")

You can use this function to restrict what maps users can vote for on your server. This is so you can have special maps that cannot be accessed through voting.

Smaller's Adapted Step-by-step Guide To Installing Admin MOD manually

> ***Note:*** *This portion of the guide discusses a* ***manual installation*** *of the admin mod – for a script-based, more automated installation, please read the Basic Default Installation section. No matter what type of installation you are undertaking, there is a lot of good information here, so be sure to read it at some point, 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)

I did change one thing from the original plan: The mod uses several config files, and the default was to scatter them in your mod directory, which I thought was messy, so I created a directory called admin and put them in there.

The counterstrike 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).

Unpack/unzip the file you downloaded. Inside the files and directory structure you just unpacked (by default the top level directory will be called "admin"), there is a dll directory. Copy everything into the dll directory inside your MOD directory:

- *If you are running* Counterstrike*, you should copy everything in the half-life\admin\dll directory to the half-life\cstrike\dll directory.*
- *If you are running* TFC*, you should copy everything in the half-life\admin\dll directory to the half-life\TFC\dll 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 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).

Now in your mod directory, make a dir called Admin. Into this you copy:

- The file called admin_help.cfg
- Everything from the directory called "samples" (strictly speaking you don't need to copy server.cfg but it doesn't take up much space)

Now you need to edit some of the config files, they are all in Unix "format" so if you are on Windows you need an editor which understands Unix line ends and stuff. *Programmers File Editor* is neat little one, it has been discontinued but is free, and its small, one source is here: http://www.winsite.com/info/pc/win95/misc/pfe101i.zip

Start by editing your main server.cfg file (this should now be in your mod directory).

Suggestion: Scroll to the bottom of the server.cfg file and make a separating comment 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:

```
users_file "admin/users.ini"
nicks_file "admin/nicks.ini"
models_file "admin/models.ini"
maps_file "admin/maps.ini"
help_file "admin/admin_help.cfg"
words_file "admin/wordlist.txt"
```

(Assuming you put these in the *name_of_your_mod*\admin dir as suggested earlier)

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

Then you locate the field called "script_file" (still in server.cfg) and make sure it's pointing to the right file, depending your operating system:

- For Windows, usually called admin_win32.amx
- For Linux, usually called admin_linux.amx

Now save the server.cfg file. 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 in the Admin subdirectory that you created earlier in your mod dir.

wordlist.txt 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 delete all the words, I don't think it's safe to just remove the file.

The **ip.in**i 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 CounterStrike has built-in voting)

The **nicks.ini** file is so that people can reserve nicknames on the server (but YOU have to write them in. ☺ Check the file for examples.

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 you max rights. (Put yourself in the nicks.ini file also)

After you have written all this, the program still is not in effect if you restart your server. One last thing 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.

Open the liblist.gam file and find gamedll_linux or gamedll (depending on your OS) and replace the text to either "dlls/admin_i386.so" or "dlls\admin.dll" respectively. (i.e. for Windows it's *gamedll "dlls\admin.dll"*)

NOW when you restart your server it will start with the admin mod. (And restart means completely stopping it - quit in the console and then restarting it)

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 somewhat for this publication.

server.cfg [sample]

Here is a list of what variables go in your server.cfg and what they do:
The easiest way to edit your server.cfg is to just copy the text from this sample server.cfg and just change the values.

variable	description
users_file "users.ini"	Defines the name of the file where the users info is stored
ips_file "ips.ini"	Defines the name of the file where the list of reserved IP's are stored
nicks_file "nicks.ini"	Defines the name of the file where the nick reservation info is stored
maps_file "maps.ini"	Defines the name of the file where the maps users are allowed to vote for is stored
vote_freq 240	Defines how often a vote can happen
nicks_kick_msg "msg"	The message displayed when a player is kicked because their credentials don't match for that reserved nick
default_access 1	Sets the access level for everyone that joins the server (1 gives admin_vote_map and admin_vote_kick to everyone)
reserve_slots 2	Keep a minimum of 2 slots free for people who have reserved access
admin_debug	Print out debug info, use if you have a problem
use_regex 0	Whether to use regex expression when matching user/model names. 1 = on, 0 = off. Note: applies to both Windows and Linux versions beginning with v2.06
models_file "models.ini"	Defines the name of the file where the list of model passwords is stored
models_kick_msg "msg"	The message displayed when a player is kicked because their model password doesn't match the password from the models.ini.
script_file cstrike/dlls/admin_win32.amx	The location of the compiled script file.
help_file admin_help.cfg	The location of the help file.
encrypt_password 1	(LINUX only) Whether to use encrypted passwords in Linux or not. (1=on, 0=off)

pretty_say 0	Whether to use the pretty centersay method. (1=on, 0=off). NOTE - Not all mods support this. If things crash set it to 0.
map_ratio 0	The percent of people that must vote YES for a map vote (valid values are 0-100)
kick_ratio 0	The percent of people that must vote YES for a kick vote (valid values are 0-100)
reserve_type 0	The reserve slots method used. 0 means just keep free for reserved slots users. 1 means kick the player with the highest ping when a person with reserved slots access joins. This method will still keep 1 slot open so people can still join to use their reserved slot. The default is 0. If using method 1 reserved_slots MUST still be greater than 0.
words_file 0	This is a list of swear words that the server will not allow. It is a new-line delimited list, which is case insensitive.
password_timeout 1800	This is the amount of time (in secs) that a password entered via admin_password remains valid.
file_access 0	This controls whether a script can write or read files. 1 is on, 0 is off.

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.

You can undefine a variable by setting to 0 like this:

nicks_file 0

users.ini [sample]

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. The format of the users.ini like this:

username:password:access

Where:

- **username** : is just the players name exactly how it is when they play the game. If you want to use a colon (":") in the players name then backslash it, like this. [CLA]\:Jodie, this will add user [CLA]:Jodie.
- **password** : is the password you want the client to have. For Win32 passwords are plain text (i.e. “butthead”), but for Linux, passwords can be either plain text or encrypted using make_pass, which is included in the .tgz file.
- **access** : is just the number you get when you add up all the numbers that correspond to the commands that you want to give this user. (See sample users.ini for examples)

access	commands that can be used
all players	admin_help,admin_userlist,admin_version,admin_timeleft
1	admin_vote_kick,admin_vote_map
2	admin_map, admin_timelimit, admin_fraglimit
4	admin_prematch,admin_reload
8	admin_pause,admin_unpause
16	admin_pass,admin_nopass
32	admin_teamplay,admin_friendlyfire
64	admin_say
128	admin_kick
256	admin_ban,admin_unban
512	admin_cfg,admin_status ,admin_rcon
1024	
2048	
4096	
8192	
16384	
32768	let this user use a reserve spot

nicks.ini [sample]

The nicks.ini file is simple; all you have to put in it is a list of players and passwords of those who don't want others using their names. The basic concept is "I password protected my name". That's all; it's not that hard. Here is the format of the nicks.ini:

username:wonid

Where username is the players name and WonID is their WonID. You can have more than once instance of username with different WonID's, so people can use multiple computers to play with a single name and multiple WonID's can be assigned to a single users, which is important since any given player can be assigned two WonID's depending on which WON server they authenticate off of.

It is a good idea to reserve nicks for all players that are in the users.ini, it just prevents confusion.

ips.ini [sample]

The ips.ini file is very simple also; it just specifies what IP addresses have access to take up one of the reserved slots.

IP addresses are defined in the IP_ADDRESS/NETMASK format (i.e. 192.168.1.0/255.255.255.0 to match 192.168.1.0-255)

models.ini [sample]

The models.ini file is simple; all you have to put in it is a list of models and passwords of those who don't want others using their models. The basic concept is "I password protected my model". That's all; it's not that hard. Here is the format of the models.ini:

model_name:password

Where **model_name** is the name of the model to protect (i.e. "sas") and **password** is the password to protect that model (same as users.ini).

> ***Note*** *that the default models – the ones that come with the game – should* ***never*** *be password protected.*

You can use this function to reserve your clan's models only for use by clan members on your server.

maps.ini [sample]

This file contains a list of maps that players are allowed to vote for. Here is the format of the maps.ini (note that it is identical to the format of your mapcycle.txt file that you already run with your server):

map_name

Where **map_name** is the name of the map which users can vote for with admin_vote_map (i.e. "2fort"). One map listed per line in the file, just like mapcycle.txt.

You can use this function to restrict that maps users can vote for on your server.

This is so you can have special maps, which cannot be accessed through voting.

make_pass [applies to Linux servers only]

For all you penguins... (In other words – Linux people only!)

make_pass is just a perl script that uses libcrypt to generate encrypted passwords. If you are using Linux then I shouldn't even explain to you how to use it, but I will just in case you're new to it or something like that. Here is the format of the make_pass command:

make_pass [-n] <username> <password>

Where:
- **-n**: with this flag the output is read to be pasted into any of the ini files
- **<username>**: is the players name when they are playing the game
- **<password>**: is the desired password in plain text

For example, if I want a user name Jodie with password bob, this is how I do it. Just cd in to your hlds_l directory then type: **make_pass Jodie bob** at prompt. It should output **jog2guNs5Pi3w**. So there is your password, just use that in the users.ini and the nicks.ini. Just remember that the client still has to type in the plain text password, not the encrypted one.

Custom Scripting

A scripting engine controls all the commands in admin mod. There are three parts to the scripting: Writing/modifying the admin script, compiling the script, and then installing it in admin mod. The scripting engine is called Small and a copy is shipped with admin mod. It can also be found online at: http://www.compuphase.com/small.htm - That page has a good description of the language.

The Basics:

Read **admin/docs/admin_mod_scripting.txt** in your installation for a list of admin mod functions. Then, look at the admin.sma file, which is located in the admin\examples directory, to see the source script that is used to define the commands in a default installation of the admin mod. Once you have this worked out, you can change the source so it does what you want.

Then, compile is using sc (or sc.exe) with the command:

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

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 Complier 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 admin.amx, which will be located in the same directory as your original admin.sma file. Now copy the admin.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 admin.amx files ARE DIFFERENT, You MUST recompile for each platform. It's the same source code file, just slightly different binaries.

Script command syntax reference:

(Updated for version 2.07)

Command	Description
say(*string to say*)	Say to everyone
changelevel(*mapname*)	Well...quite obvious, instant change, just like a rcon changelevel.
convert_string(*value, string*)	Used to convert strings from the hl engine into small strings. Value is passed and turned into a string.
auth()	Return whether this user is valid (i.e in users.ini with correct password). Takes a string parameter (i.e. "auth(a)" where "a" is the player name to auth against. "" (no name) means auth against the current player.
valid_map(*map_name*)	Return whether this map is a valid map name.
message(*user,string*)	Send the string to user as a message.
access(*a*,*username*)	Return whether the user *username* has access to level a. Use "" for the current user.
set_timer(*command to issue, delay, repeat number*)	*"command to issue"* is a public function from the script file. *"delay"* is how many seconds from now, "*repeat number"* is how often. Put 99999 for infinite repeats.
vote(*question, option 1, option 2*	Display question and then option1

command to run, variable to pass)	option 2, ... (up to 9). Command to run is a public function to run when the vote finished. Variable to pass is the variable to provide to the public function (see admin.sma for examples)
kick*(nickname/playerid (not WON id)*)	kick that person if you have access.
vote_allowed(*vote_num*))	Return whether that vote number is allowed to happen right now.
check_user(*username*))	Return whether username is a valid current user on the server.
exec(*command*)	Run "command" on the server
log(*string*)	Log string to the log file.
ban(*nickname/WONid/playerID*)	Quick ban of player. You can determine a WONid from a nickname etc.
reload()	Reload admin mods config files.
version()	Print out the current version of admin mod.
timeleft()	Print out AND return the timeleft on the current map.
nextmap(mapname,length)	Print out the next map after mapname n the map cycle. If length is zero the next map after the current one is printed.
centersay(*string,time to display,red,green,blue*)	Print out string to all clients for "time to display" seconds with the color (r,g,b).
getvar(*string*)	Get the value of the half-life variable "string". (i.e. mp_footsteps)
getstrvar(*string,string to store in, string length*)	Get the string value of the half-life variable "string". (i.e. admin_version)
strtonum(*string*)	Returns the integer value stored in *string*
check_words(*string*)	Returns 1 if *string* contains one of the words in the file defined by the words_file cvar. Returns 0 otherwise.
get_username(*string*,*string1*,*n*)	Copies the username of the player defined in *string* (either userid or wonid) into *string1*, for a maximum of *n* characters.
help(*keyword*)	Have an admin_help.cfg with a list of

	help topics. Do this so we can add help topics in for the commands we add, i.e.: admin_help.cfg keyword: text text text text text..... keyword: text text text text text.... Keyword is optional. If blank the whole file is printed.
currentmap(*map*,*length*)	Return the current map name into *map* with a maximum of *length* characters.
maxplayercount()	Returns the maximum number of players allowed on the server.
playercount()	Returns the current number of players on the server.
playerinfo(*PlayerIndex*,*Name*,*Length*,*&UserID*,*&WONID*)	*Playerindex* is a number between 0 and maxplayercount() that denotes a player on the server. The name, userid and wonid of the user *playerindex* is returned. *Length* is the maximum length of *Name*.
getteamcount(*Team*)	Returns the number of players on team number *Team*.
writefile(*filename*,*data*)	Writes the string *data* to *filename*, where *filename* is relative to the mods base dir.
resetfile(*filename*)	The file *filename* is emptied (all the data removed from it).
readfile(*filename*,*string*,*lineno*,*len*)	Line number *lineno* is read into *string* from *filename* for a maximum of *len* characters.
messageex(*user*,*string*,*type*)	Say private message *string* to *user* with *type* where type can be: • print_console • print_center • print_chat

Check out admin.sma for examples of how to use the functions.

More on 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 admin.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"

Where to go for more help

Alfred, the creator of the admin mod, can be reached either through the online discussion forums on the admin mod web site at http://halflifeadmin.sourceforge.net/ or through email at alfred@mazuma.net.au -- Keep in mind that Alfred's dedication to this application is a labor of love. He works hard to produce an application used by many people, and on top of that it's an open-source project, which means he is not compensated financially!

You can help by participating in the discussions, helping others, and contributing your ideas and skills to the future of the admin mod. Visit the web site at http://halflifeadmin.sourceforge.net/ and join the community!