

# Phasor – For Halo PC

Made By: Oxide aka Genocide, urbanyoung.

*Rewritten for 01.00.10.057*

<http://haloapps.wordpress.com>

## Introduction:

This is a guide to the basics of the Phasor specifically version 01.00.10.057. Phasor is a server modification designed for Halo PC version 01.00.09.620. I plan to add support for other versions and Halo CE, but it is still currently under development. Phasor has two main aspects:

1. The internal, compiled stuff that a programmer could edit.
2. The scripting interface that (hopefully) anyone can edit.

I'll go over the internal (core) aspects now and the scripting stuff later. The scripting stuff is where all the cool mods come into play. However, you **do not** need to script to use Phasor.

A list of the main aspects is below:

- Hash based rcon system. This is pretty self-explanatory I think, in addition to an rcon password admins are recognized by their unique cd key hash. This means that even if someone gets your rcon password, they can't do anything because their hash isn't recognized. You can also give different admins different access levels. This is done through the included program, RconAccessEditor (see later). This feature is disabled when no admins are added.
- Map voting. When a game ends the players can (if enabled) vote for the next map. This is done through Halo's chat system. The idea/usage is basically the same as Gandanur's although the code is obviously different.
- Support for non-default maps. Phasor gives you the ability to run any Halo PC map file regardless of its name (ie bloodgulch1.map) so long as it's less than 32 characters long. Currently the server only loads maps that it recognizes, or those with original names. But because Phasor lets you load other maps you can now run various, modded, maps from the same install directory.
- Automated server messages. Phasor lets you send the server a message after a specified time. You can send as many messages as you like using whatever delay suits you. You can also do event based messages, which you can find out about later on. Sending events based messages requires using scripts.
- Logging. Phasor logs server events including rcon usage, game events and Phasor specific messages.
- More server commands. As you may have guessed, in order to do all this stuff I needed to add a few more servers commands. They are listed at the bottom of this section.

- **AFK Kicking.** You can specify a time after which AFK players are kicked. A player is deemed AFK (away) if they don't move their camera (look around) or chat for 1 minute. After xx minutes (specified via `sv_kickafk`) the player is kicked. Each minute they are inactive they are warned via a chat message. Admins are not kicked.
- **Alias system.** Some people may be familiar with an alias system; I've used the term in other programs I've released (although there weren't for Halo). Basically this tracks the names that a hash uses. As a hash is unique to a cd key, it tracks all names a player uses (generally) for one computer. You can use `sv_alias` for a list of these names. You can also search for the names a hash has used and search for hashes associated with player names.
- **Teleporting.** You can now teleport players (or objects) using `rcon` commands and/or the script interface. You can teleport players to predefined locations (ie `redbase`) or specify the x/y/z coordinates manually.
- **Object management.** As of 01.00.10.057 Phasor lets scripts manage objects. This includes the creation and deletion of objects, assigning weapons to players and even forcing players in/out of vehicles.

### **Included Tools:**

I've included a couple of tools with this new release; you can find them in the Tools folder of the archive you downloaded. I'll briefly describe what each one does.

- **AliasConversion.exe** – This program can be used to convert old alias files (produced by versions before 01.00.10.057) to the new format used from 01.00.10.057 and later.
- **Multiclient.exe** – This program is useful when you want to test scripts, but need more than one player. It will let you launch numerous instances of Halo so you can join your server. Each client is given a random cd key hash and as such any servers you want to join will need to have `sv_public` set to 0. It only works for Halo PC 1.09 and should be placed in your Halo game folder. Any command line arguments passed to the multiclient will be forwarded to the Halo process. The commands I use are:  
"C:\Program Files (x86)\Microsoft Games\Halo\Multiclient.exe" -console -windowed -novideo -nosound -connect 127.0.0.1:2302 -vidmode 640,480,30
- **RConAccessEditor.exe** – This program is used to create and modify access levels. You can read more about it later.

### Commandline Options:

Phasor supports a few command line options. You can specify the path that Phasor (and the server) searches for data, using `–path`. There is one additional parameter which is specific to Phasor, `-mappath`. This can be used to specify the path where Phasor searches for map files.

### Installation:

I assume you've downloaded the necessary files, if not head over to [Halo Apps](#) and you'll find what you need.

Extract the files within Phasor\_010010057.zip somewhere. Before installing Phasor I recommend you backup the **strings.dll** that you'll find in your **server folder**. This can be used later for uninstalling Phasor (simply replace the modified dll I provide with the original). Once you've made a backup, copy 'strings.dll', 'Phasor.dll' and 'sqlite3.dll' into your Halo Server folder. You may be asked whether or not what want to overwrite **strings.dll**, just say yes because you made a backup.

Once you've done that, browse to the path that halo uses to store data. By default, this is C:\Users\<username> \Documents\My Games\Halo, but if you use the `–path` command line parameter the directory is what you specify. This folder is where you need to copy the included scripts folder. I will refer to this directory as the server's profile path. Phasor saves a lot of data in this folder. Logs are stored in \logs, alias and teleport data is stored in \data.

Finally, if you want to use the hash based admin system you will need to create an access file using the included RconAccessEditor. The access file is used to specify the commands that a certain admin level can use. Usage of the program should be self explanatory. Once you've created an access file, it needs to be copied into the server's profile path.

Once you've done all of that, Phasor is installed and you can use the new commands.

### Server Commands:

Many of these commands aren't saved and as such should be put into your init file that the server executes while loading. Commands that save data (and don't need to be in the init file) are: admin commands, teleport commands.

### Admin Commands:

1. `sv_admin_add` – This function adds a new player to the admin list. Doing so gives said player the ability to use server commands (up to the specified level).  
Syntax: `sv_admin_add <hash or player number> <auth name> <access level>`
2. `sv_admin_del` – This function removes a previously saved from player the admin list.  
Syntax: `sv_admin_del <admin id>`

3. `sv_admin_list` – This function lists the current admins (along with their admin ids)  
Syntax: `sv_admin_list`
4. `sv_admin_cur` – This function lists the admins who are currently in the server.  
Syntax: `sv_admin_cur`
5. `sv_viewadmins` – This function shows the names of the current admins in the server.  
Syntax: `sv_viewadmins`
6. `sv_reloadaccess` – This function reloads the access list file.  
Syntax: `sv_reloadaccess`

#### Alias Commands:

1. `sv_alias` – This function is used to enable/disable the alias system. By default, the alias system is active.  
Syntax: `sv_alias <status>` where status can be true, false, 1, 0.
2. `sv_alias_search` – This function searches the alias database for a player name.  
Syntax: `sv_alias_search <data to find>` Note: Wildcard searches are supported and can be specified with a %. ie `sv_alias_search Ox%`
3. `sv_alias_hash` – This function searches the alias database for players whose hash matches that specified.  
Syntax: `sv_alias_hash <hash to find>`

#### Logging Commands:

*Note, the valid log types are either game or phasor. Use them for <log type>*

1. `sv_logname` – This function sets a custom name for a specific type of log.  
Syntax: `sv_logname <log type> <file name>`
2. `sv_stoplog` – This function disables a specific type of log.  
Syntax: `sv_stoplog <log type>`
3. `sv_savelog` – This function forces the specified type of log to save (and clear cache).  
Syntax: `sv_savelog <log type>`
4. `sv_loglimit` – This function specifies the maximum size (in kB) a log is allowed to be. Once it is exceeded the log is moved to the 'old' folder and a new log is started.  
Syntax: `sv_loglimit <type> <size>`

#### Map Vote Commands:

1. `sv_mapvote` – This function enables/disables map voting.  
Syntax: `sv_mapvote <true or false, 1 or 0>`
2. `sv_mapvote_add` – This function adds an option to the vote list.  
Syntax: `sv_mapvote_add <map> <gametype> <description> opt: <script 1> <script 2> . . .`
3. `sv_mapvote_del` – This function removes the specified option from the vote list.  
Syntax: `sv_mapvote_del <map vote index>`

4. `sv_mapvote_list` – This function displays all map vote options, with their indices.  
Syntax: `sv_mapvote_list`

#### Message Commands:

1. `sv_welcome_add` – This function adds a message to the welcome list, which is sent to players as they join the game.  
Syntax: `sv_welcome_add <message>`
2. `sv_welcome_del` – This function removes a message from the welcome list.  
Syntax: `sv_welcome_del <index>`
3. `sv_welcome_list` – This function lists the welcome messages. The output indices can be used with `sv_welcome_del` to remove messages.  
Syntax: `sv_welcome_list`
4. `sv_msg_add` – This function adds a message to the automated message list, which is sent to players after a specified delay.  
Syntax: `sv_msg_add <message>`
5. `sv_msg_del` – This function removes a message from the automated message list.  
Syntax: `sv_msg_del <index>`
6. `sv_msg_list` – This function lists the messages in the automated message list. The output indices can be used in `sv_msg_del`.  
Syntax: `sv_msg_list`
7. `sv_msg_start` – This function begins the sending of automated messages.  
Syntax: `sv_msg_start <delay in minutes>`
8. `sv_msg_end` – This function stops the sending of automated messages.  
Syntax: `sv_msg_end`

#### Team Commands:

1. `sv_teams_balance` – This function balances the team (only based on numbers, not score etc).  
Syntax: `sv_teams_balance`
2. `sv_teams_lock` – This function locks the teams, disallowing players from changing teams themselves.  
Syntax: `sv_teams_lock`
3. `sv_teams_unlock` – This function unlocks the teams, allowing players to change team themselves.  
Syntax: `sv_teams_unlock`
4. `sv_changeteam` – This function change's a player's team.  
Syntax: `sv_changeteam <player>`

### Teleport Commands:

1. **sv\_teleport** – This function ‘teleports’ a player to the specified location. The location can either be coordinates (x, y, z) or the name of a saved location.  
Syntax: **sv\_teleport** <player> <location name or <x> <y> <z>>
2. **sv\_teleport\_add** – This function adds your current location (or specified coordinates) to the location list. It can be referred to by the name you choose. Locations are saved using the current maps base name (ie bloodgulch1 would save to bloodgulch) and can only be used within any other variant of that map.  
Syntax: **sv\_teleport\_add** <name> opt: <x> <y> <z>
3. **sv\_teleport\_del** – This function removes a previously saved teleport location.  
Syntax: **sv\_teleport\_del** <index>
4. **sv\_teleport\_list** – This function lists all teleport locations for the currently loaded map.  
Syntax: **sv\_teleport\_list**

### General Commands:

1. **sv\_kickafk** – This function enables/disables the kicking of inactive players. When enabled, each minute a player is AFK they receive a warning. After the specified time runs out they are kicked. Admins are immune to being both warned and kicked.  
Syntax: **sv\_kickafk** <time> note: The time is in minutes and a time of 0 disables kicking.
2. **sv\_commands** – This function lists the commands you’re authorized to use.  
Syntax: **sv\_commands**
3. **sv\_say** – This function sends a message to the server.  
Syntax: **sv\_say** “message”
4. **sv\_gethash** – This function gets the cd key hash of the specified player.  
Syntax: **sv\_gethash** <player>
5. **sv\_kill** – This function kills the specified player  
Syntax: **sv\_kill** <player>
6. **sv\_reloadscripts** – This function reloads all currently loaded scripts.  
Syntax: **sv\_reloadscripts**
7. **sv\_setspeed** – This function modifies the specified player’s movement speed.  
Syntax: **sv\_setspeed** <player> <speed>
8. **sv\_invis** – This function makes the specified player invis for the (optionally) specified duration.  
Syntax: **sv\_invis** <player> opt: <duration>
9. **sv\_chatids** – This function can be used to enable/disable the displaying of a player’s id in chat.  
Syntax: **sv\_chatids** <status>

10. sv\_host – This function sets the data that is sent to my server. You can use it to set information about who hosts the server.

Syntax: sv\_host <info>

#### Modified Default Commands:

I don't want to list all the commands, but I think it's important to mention this. Phasor modifies how some map commands behave. I added this so you could specify scripts within the mapcycle (and when using sv\_map). As such functions like sv\_mapcycle\_add and sv\_map (along with others) have been modified. If you don't specify any scripts then none will be loaded and they will work as normal.

That's all of the commands in this build. For an example of how to setup the init file, see the next page

#### Access Levels:

Phasor uses various levels to classify admins. These levels specify what an admin can and cannot do. You can create access levels using the included RconAccessEditor. It should be self explanatory. After you've made your levels you need to make sure that the file it created (access.ini) **is in your profile directory**. Without access levels Phasor cannot use its hash based admin system.

*Note: Your profile directory is specified via the -path command line parameter. If no path is specified the directory is where the ban list is stored. This goes for all file reading/writing (including gametypes, excluding maps).*

#### Script Interface:

By now I hope you know the basics because I'm moving onto the script interface. This is what makes Phasor powerful. Phasor uses Lua 5.14 (*Lua is a scripting language*) to notify scripts of various events that happen throughout the game. These scripts can **modify** and **control** how these events behave. For example, you could send welcome messages as a player joins or you could stop a player from changing team. As an introduction to the scripting system I put all these functions together and made an **Infection** gametype. I also included a chat filter, server commands and spawn protection example. You can get help and find more scripts at the Phasor forum. <http://phasor.proboards.com>.

I understand that you may not like how I've setup my scripts. If you feel this way you can modify them. It's really easy to do. Please don't "pigeon hole" Phasor as "that zombie mod" as it's more than that, and I really hope people will show me what they can do.

Prior to 01.00.03.104 Phasor only supported one script being loaded at a time. I didn't really like this and as such, Phasor can now load various scripts at a time. It's important to note



that some scripts can interfere with each other, so be careful. *As a rule, only the first acting script's return value is considered.* That basically means first come first serve. Scripts are loaded in the order you specify them. If the first script returns a non-default value for a function then the return values of all subsequent scripts will be ignored for that function call.

Scripting may look scary at first but I encourage you to try it. If you spend an hour or two getting familiar with Lua I'm sure you can make some really cool stuff. I've released a Phasor scripting guide which can help you on your way. If you think scripting may interest you then you should definitely check it out.

### **Final stuff:**

That's all the stuff I really want to explain here. I hope it's helped. I rewrote this file for the 01.00.10.057 release and I hope it makes this a lot clearer. If you have any issues feel free to contact me at the Phasor bug forum (<http://phasor.proboards.com>).

My example scripts (along with Phasor) are distributed under the terms of the GNU General Public License:

*Phasor is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. Phasor is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.*

*You should have received a copy of the GNU General Public License along with Phasor. If not, see <<http://www.gnu.org/licenses/>>.*

Phasor uses [Lua](#) for its scripting interface; Lua is used in compliance with the following licence.

Copyright © 1994–2010 Lua.org, PUC-Rio.

*Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:*

*The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.*

*THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.*

Phasor also uses cURL, which is used in compliance with the following licence.

Copyright (C) 1998 - 2011, Daniel Stenberg, <daniel@haxx.se>, et al.

This software is licensed as described in the file COPYING, which you should have received as part of this distribution. The terms are also available at <http://curl.haxx.se/docs/copyright.html>.

You may opt to use, copy, modify, merge, publish, distribute and/or sell copies of the Software, and permit persons to whom the Software is furnished to do so, under the terms of the COPYING file.

This software is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.

With that said there's really only one more thing I want to mention. I like feedback about my work, but I don't like people complaining. Feel free to give me constructive feedback. A lot of time and effort has gone into Phasor. If you'd like to say thanks there's a donation link below. Donations and feedback keep a programmer motivated.

#### [Support Phasor](#)

*Phasor was developed by Oxide and uses Lua 5.14. It was programmed in C++ using Visual Studio 2008 SP1. It is **completely open source**. You should have received a copy of the source code when you downloaded Phasor.*

#### Credits and Thanks:

I'd like to take the chance to thank people that helped contribute to Phasor. Please take the time to read over this. They are listed in no particular order.

1. TCP/IP – He has given me heaps of ideas and shown a lot of interest in Phasor. Look out for some cool scripts he makes.
2. Church/Wyatt – He probably gave me the idea to make Phasor and reminded me of Roulette. What I **didn't** want Phasor to become. He gave me guidance and helped with a little bit of the code.
3. Smiley – Well, he really transformed Phasor. Before I read his post about making an Infection gametype I was really just focusing on core stuff. He gave me the idea for a script interface and made Phasor what it is.

4. [DWM]Wizard – He hosted a server so that I could really test Phasor out. Without him Phasor wouldn't be close to completion.
5. Goemitar – While he didn't directly help me I got to learn what people wanted from Gandanur. It gave me ideas. Also, to keep stuff familiar to players a few things in Phasor are modelled after his implementation (ie map voting).
6. UrbanGrafix – Testing the shit out of it and making various debug reports.
7. Modhalo.net community – I learned quite a bit about map structure from them, check 'em out.
8. Everyone who's created scripts, given me ideas and helped with testing.

I'm sorry if I forgot anyone, if I did then thanks to them too.

And finally, thanks to you for downloading. Thanks to you for playing on my development servers even though they had their fair share of downtime.

Remember Phasor is just a tool that **YOU** can use to make your ideas a reality. I really hope people create **AND SHARE** some Phasor scripts. The whole reason I gave Phasor the script interface was so that you could modify it to meet your needs.

#### **Change log:**

01.00.02.468 – *Changed the admin system a little, if no admins are in the list then only the rcon password is used. Fixed a buffer allocation error, this should fix Phasor (sometimes) crashing when loading/unloading. Should also fix update bug. Statically link CRT library so the C++ runtime isn't necessary. Also fixed the rcon access level creator, it saves to access.ini not AccessList.ini.*

01.00.03.104 – This build has significant structural changes to Phasor. All scripts require updating to run on this build. Includes many bug fixes and new features.

01.00.10.057 – This build is a complete rewrite of Phasor. Once again all scripts require updating. This build introduces a lot of new features. I don't have a list of them but you should notice a lot of new stuff throughout Phasor.

#### **Updating scripts to 01.00.10.057:**

This build has quite a few significant changes to the scripting system. They are detailed below.

1. registertimer has changed its behaviour. In previous versions its syntax was 'registertimer(id, delay, userdata, callback)'. In this release the id parameter is removed, instead registertimer returns the id for the created timer. The order of parameters has also changed and it can accept multiple userdata parameters.  
Definition: registertimer(delay, callback, <opt>: userdata1, userdata2 ...)  
Return value: id of created timer

Examples: `timer_id = registertimer(1000, "MyTimer", data1, data2, data3)`  
`registertimer(1000, "MyTimer")`

Because the register timer call has changed, the definition of the callback functions needs to change too. The definition is dependent upon how `registertimer` is called, specifically how many userdatas were specified. A couple examples of how to define the callback are below.

```
function MyTimer(id, count)
```

```
function PlayerChangeTimer(id, count, data1, data2, data3)
```

A "bug" has also been fixed with timers, the callback function is no longer called with a count of 0 when the timer is created. It is first called once the delay runs down and with count of 1.

2. The return values for Phasor exported functions have changed. In previous builds the return value on error varied, it could have been -1, 0 or something else. Starting from this build return values are more consistent. If a function fails to execute as expected (ie couldn't find an object, player etc) a value of nil is returned. As such all functions should be checked against nil to determined if it successfully executed.
3. `OnWeaponCreation` has been replaced with `OnObjectCreation` and as such any code that used to occur in `OnWeaponCreation` should be move to `OnPlayerSpawn`. This means that any changes to ammo counts won't sync unless the new function 'updateammo(weaponId)' is called. `OnObjectCreation` is defined as:
  - `function OnObjectCreation(m_objectId, player_owner, tag)`
4. `rresolveplayer` has been fixed, the input should now be 1 based (that is don't subtract 1 from the input value).
5. Other script functions definitions have changed, they are listed below:
  - `function OnVehicleEntry(relevant, player, vehicleId, vehicle_tag, seat)`
6. Finally, there are new functions:
  - `function OnVehicleEject(player, forceEject)`
  - `function OnClientUpdate(player, m_objectId)`

For details about new functions, read the updated scripting guide.