Creating a script file for use in a scenario tag:

1. Open up your favorite text editor
2. Save the file in the "data" folder in your Halo Custom Edition folder with the "hsc" as your file extension
3. Write some code
4. Save it again, with the new code
5. Run Kornman00.exe (or any other hacked Guerilla with everything unlocked), and open your scenario file (back a backup now if you wish)
6. You can either add the script to the AI Script References block or Source Files block, depending on what the script was for.
7. Press "add" on which ever script reference block you choosed.
8. Enter in your scripts name in to the textbox marked "Source name"\"Name", you don't have to add the file extension (hsc). (if you saved it in a subfolder in the "data" folder, you would add the relative path name with you script name. IE "global_script.hsc" is your script name, and its saved in the "data\scripts\tutorial\" folder, so what you would enter is "scripts\tutorial\global_scripts")
9. Repeat steps 8 & 9 if you have more than 1 script
10. Save your scenario file and close your tag editor.
11. Run Sapien and open up your scenario file.
12. Once Sapien gets done loading it will compile your script(s) you've made and added and will report any errors to the game window.
13. It will then merge the source file with your scenario file (if everything went ok) then you can save your scenario file.
14. Make a map file out of your new scenario file.
15. Test it in the game!

; *semi-colon*

The semi-colon marks the beginning of a *comment*. A comment is a block of text that LISP ignores. LISP ignores all text after a semi-colon until the end of the line. LISP programmers often use one, two or three semicolons to indicate the importance of the comment. Three semi-colons means a globally important comment. Two semi-colons introduces a comment about a block of code. One semi-colon introduces a single line comment. Here is an example that shows all three levels

```
;;; the detune function return a value randomized by some percentage

(defun detune (hz pct)
  ;; should this function allow keynums????
  (+ hz
     (random (* pct hz))))  ; uniform distribution
```

' *quote*

The quote character is used to block LISP evaluation. When LISP evaluates a quoted thing the quote is "stripped off" and the expression following the quote is returnd.

```
? (+ 3 4)    ; LISP evalutates the list
7

? '(+ 3 4)   ; quote blocks evaluation
```

```
(+ 3 4)       ; the quote is stripped off

? PI         ; LISP evalutates the symbol
3.141592653589793

? ''PI       ; the first quote is stripped off
'PI       ; leaving the second quote intact
```

: *colon*

The colon introduces a special type of symbol called a *keyword*. A keyword is *self-evaluating*, that is, it is constant data.

```
? :woof-woof!
:WOOF-WOOF!
```

*cross*

The cross character is used to introduce a special interpretation of the subsequent expression. Here are several examples:

`#xFF00`
: Introduces digits in hexidecimal radix.

`#(1 2 3)`
: Creates a vector (array) from a list of elements.

`#'list`
: Returns the function associated with the symbol `list`. If no function is associated with the symbol LISP signals an error.

`#|`
: Introduces a comment that lasts more than one line. A `#|` must be balanced by a terminating `|#`.

Function Call Argument Syntax

In order to call a LISP function a programmer must first know the *syntax*, or calling convention, of the function. This syntax is determined by the programmer when the function is defined. Each function definition provides a list of the function's *lambda parameters*, the "windows" through which the function receives its input arguments. The lambda parameter list declares both the number and type of arguments the function takes:

Required Arguments
: Required arguments are arguments that must be specified in order for the function to work. Required arguments appear first in the function call list. LISP signals an error if too few or too many required arguments are specified.

Optional Arguments (`&optional` parameter specifier)
: Optional arguments may be specified or not. If specified, they appear directly after the required arguments. An optional argument need only be specified if it is different than the *default argument value* defined by the function. Function parameters that accept optional arguments are marked `&optional` in the function definition. ("&optional" is pronounced "and optional".)

Keyword Arguments (&key parameter specifier)
: If keyword arguments are defined they appear after all the required and optional arguments. The syntax of a keyword argument is:

  `:keyword value`

  where *`:keyword`* is the name of the keyword argument and *`value`* is its value. A keyword argument need only be specified if it is different than the default value defined by the function. Keyword arguments may appear in any order since each keyword is mentioned by name in the set. Function parameters that accept keyword arguments are marked `&key` in the function definition. ("&key" is pronounced "and key".)

Rest Arguments (`&rest` parameter specifier)
: Rest arguments may appear after all the required, optional and keyword arguments. The term Rest means "the rest of the arguments". Function parameters that accept rest arguments are marked `&rest` in the function definition. ("&rest" is

pronounced "and rest".)

It is quite common for a function to use mixtures of different argument types. For example, the arithmetic operators + and * use just Rest arguments, which means that both + and * accept any number of inputs, including none. The - and / operators, on the other hand, are defined to accept 1 required argument followed by any number or Rest arguments.

Script Type	Details	Example
startup	Performed only on map startup	(script startup init_shit)
dormant	Performed when something happens?	(script dormant wait_for_it)
continuous	Always being performed	(script continuous spawn_warthog)
static	Performed when called from another script	(script static multiply_5)
stub	Sort of like a prototype. Doesn't do anything	(script stub wtf)

Return Type	Details	Example
special form	Something special indeed	
function name	Script name?	
passthrough	returns nothing?	
void	returns nothing	
boolean	A value that is true or false	true false
real	Floating point value Value Range: 3.4E +/- 38 (6 digits)	3.000000
short	Short integer value Lowest Value: -32767 Highest Value: 32768	2
long	Long integer value Lowest Value: –2,147,483,648 Highest Value: 2,147,483,648	2000000000
string	String of characters in double quotes Max number of characters: 32?	"This is a string"
trigger_volume	A "Trigger Volumes" value (a subcatagory in the scenario tag)	
cutscene_flag	A "Cutscene Flags" value (a subcatagory in the scenario tag)	
cutscene_camera_point	A "Cutscene Camera Points" value (a subcatagory in the scenario tag)	
cutscene_title	A "Cutscene Titles" value (a subcatagory in the scenario tag)	
cutscene_recording	A "Cutscene Recording" value that isn't in the public HaloCE scenario tag?	
device_group	A "Device Groups" value that isn't in the public HaloCE scenario tag?	
ai	A "Encounters" value? (a subcatagory in the scenario tag)	
ai_command_list	A "Command Lists" value (a subcatagory in the scenario tag)	
starting_profile	A "Player Starting Profile" value (a subcatagory in the scenario tag)	

conversation	A "AI Conversations" value (a subcatagory in the scenario tag)	
navpoint		
hud_message	Hud_message tag [hmt]	
object_list	A object list	
sound	Sound tag [snd!]	
effect	Effect tag [effe]	
damage	Damage tag [jpt!]	
looping_sound	Looping_sound tag [sndl]	
animation_graph	Animation_graph tag [antr]	
actor_variant	Actor_variant tag [actv]	
damage_effect	Damage_effect tag	
object_definition		
game_difficulty	easy normal hard impossible	
team	player human covenant flood sentinal	
ai_default_state	none sleeping alert moving guarding searching fleeing	
actor_type	name of an actor	jacktest
hud_corner	top_left top_right bottom_left bottom_right	
object	Object tag [obj]	
unit	Unit tag [unit]	
vehicle	Vehicle tag [vech]	
weapon	Weapon tag [weap]	
device	Device tag [devc]	
scenery	Scenery tag [scen]	
object_name	A "Object Names" value (a subcatagory in the scenario tag)	
unit_name	A "Bipeds" value (a subcatagory in the scenario tag)	
vehicle_name	A "Vehicles" value (a subcatagory in the scenario tag)	
weapon_name	A "Weapons" value (a subcatagory in the scenario tag)	
device_name	A "Device" value (a subcatagory in the scenario tag)	
scenery_name	A "Scenery" value (a subcatagory in the scenario tag)	

Expression	Details	Usage\Example

(begin <expression(s)>)	returns the last expression in a sequence after evaluating the sequence in order.	
(begin_random <expression(s)>)	evaluates the sequence of expressions in random order and returns the last value evaluated.	
(if <boolean> <then> [<else>])	returns one of two values based on the value of a condition.	(if (and (player_action_test_action) true) (cheat_all_powerups))
(cond (<boolean1> <result1>) [(<boolean2> <result2>) [...]])	returns the value associated with the first true condition.	(cond ((=exterior_player_location 1) (migration_a_a)) ((=exterior_player_location 2) (migration_a_b)) ((=exterior_player_location 3) (migration_a_c)) ((=exterior_player_location 4) (migration_a_crev)) ((=exterior_player_location 5) (migration_b_a)) ((=exterior_player_location 6) (migration_b_b)))
(set <variable name> <expression>)	set the value of a global variable.	(set global_string1 "Set")
(and <boolean(s)>)	returns true if all specified expressions are true.	(and (player_action_test_action) true)
(or <boolean(s)>)	returns true if any specified expressions are true.	
(+ <number(s)>)	returns the sum of all specified expressions.	(+ 5 6 7 8 9) returns:35 (+ 5.5 6.6 7 8.4 9) returns:36.5
(- <number> <number>)	returns the difference of two expressions.	(- 10 5) returns:5 (- 1 0.5) returns:0.5
(* <number(s)>)	returns the product of all specified expressions.	(* 5 5) returns:25 (* 5.5 6) returns:33
(/ <number> <number>)	returns the quotient of two expressions.	(/ 10 5) returns:2 (/ 2.5 2) returns:1.25
(min <number(s)>)	returns the minimum of all specified expressions.	(min 1 3 2 4 5 7 6 8 9) returns: 1
(max <number(s)>)	returns the maximum of all specified expressions.	(max 1 3 2 4 5 7 6 8 9) returns:9
(= <expression> <expression>)	returns true if two expressions are equal	(= (hud_get_timer_ticks) 0)
(!= <expression> <expression>)	returns true if two expressions are not equal	(!= (hud_get_timer_ticks) 0)
(> <number> <number>)	returns true if the first number is larger than the second.	(> 10 5) returns: true (> 5 10) returns: false
(< <number> <number>)	returns true if the first number is smaller than the second.	(> 4 8) returns: true (> 8 4) returns: false
(>= <number> <number>)	returns true if the first number is larger than or equal to the second.	(>= 10 10) returns: true (>= 5 10) returns: false
(<= <number> <number>)	returns true if the first number is smaller than or equal to the second.	(>= 4 4) returns: true (>= 8 4) returns: false
(sleep <short> [<script>])	pauses execution of this script (or, optionally, another script) for the specified number of ticks.	(sleep 100 more_weapons)

(sleep_until <boolean> [<short>])	pauses execution of this script until the specified condition is true, checking once per second unless a different number of ticks is specified.	(sleep_until false 5)
(wake <script name>)	wakes a sleeping script in the next update.	(wake more_weapons)
(inspect <expression>)	prints the value of an expression to the screen for debugging purposes.	(inspect (* 5 5) returns: 25 in console window
(unit <object>)	converts an object to a unit.	(unit chief)
(ai_debug_communication_suppress <string(s)>)	suppresses (or stops suppressing) a set of AI communication types.	
(ai_debug_communication_ignore <string(s)>)	ignores (or stops ignoring) a set of AI communication types when printing out communications.	
(ai_debug_communication_focus <string(s)>)	focuses (or stops focusing) a set of unit vocalization types.	
(not <boolean>)	returns the opposite of the expression.	
(print <string>)	prints a string to the console.	(print "50 dollars for this?!") returns: "50 dollars for this?!" to the console
(players)	returns a list of the players	(players)
(volume_teleport_players_not_inside <trigger_volume> <cutscene_flag>)	moves all players outside a specified trigger volume to a specified flag.	
(volume_test_object <trigger_volume> <object>)	returns true if the specified object is within the specified volume.	(volume_test_object trig_volume1 player0)
(volume_test_objects <trigger_volume> <object_list>)	returns true if any of the specified objects are within the specified volume.	(volume_test_objects trig_volume2 (players))
(volume_test_objects_all <trigger_volume> <object_list>)	returns true if any of the specified objects are within the specified volume.	(volume_test_objects_all trig_volume2 (players))
(object_teleport <object> <cutscene_flag>)	moves the specified object to the specified flag.	(object_teleport player0 red_base_flag)
(object_set_facing <object> <cutscene_flag>)	turns the specified object in the direction of the specified flag.	(object_set_facing player0 blue_base_flag)
(object_set_shield <object> <real>)	sets the shield vitality of the specified object (between 0 and 1).	(object_set_shield player0 1.0) ;; Set players shield to full
(object_set_permutation <object> <string> <string>)	sets the desired region (use "" for all regions) to the permutation with the given name, e.g. (object_set_permutation flood "right arm" ~damaged)	(object_set_permutation player0 "right arm" ~damaged)
(object_create <object_name>)	creates an object from the scenario.	(object_create warthog_mp_1)
(object_destroy <object>)	destroys an object.	
(object_create_anew <object_name>)	creates an object, destroying it first if it already exists.	(object_create_anew banshee_mp_1)
(object_create_containing <string>)	creates all objects from the scenario whose names contain the given substring.	(object_create_containing "warthog")
(object_create_anew_containing <string>)	creates anew all objects from the scenario whose names contain the given substring.	(object_create_anew_containing "pelican")
(object_destroy_containing <string>)	destroys all objects from the scenario whose names contain the given substring.	(object_destroy_containing "pelican")
(object_destroy_all)	destroys all non player objects.	(object_destroy_all)

(list_get <object_list> <short>)	returns an item in an object list.	(list_get the_warthogs 3)
(list_count <object_list>)	returns the number of objects in a list	(list_count the_warthogs)
(effect_new <effect> <cutscene_flag>)	starts the specified effect at the specified flag.	(effect_new "effects\coop teleport" teleporting_flag)
(effect_new_on_object_marker <effect> <object> <string>)	starts the specified effect on the specified object at the specified marker.	(effect_new_on_object_marker "effects\burning large" warthog_mp "driver")
(damage_new <damage> <cutscene_flag>)	causes the specified damage at the specified flag.	(damage_new "scenery\emitters\burning_flame\flame" enter_lava_flag)
(damage_object <damage> <object>)	causes the specified damage at the specified object.	(damage_object "weapons\assault rifle\bullet" player0)
(objects_can_see_object <object_list> <object> <real>)	returns true if any of the specified units are looking within the specified number of degrees of the object.	(objects_can_see_object the_warthogs player0 90)
(objects_can_see_flag <object_list> <cutscene_flag> <real>)	returns true if any of the specified units are looking within the specified number of degrees of the flag.	(objects_can_see_flag the_warthogs tunnel_flag 45)
(objects_delete_by_definition <object_definition>)	deletes all objects of type <definition>	
(sound_set_gain <string> <real>)	absolutely do not use this	
(sound_get_gain <string>)	absolutely do not use this either	
(script_recompile)	recompiles scripts.	(script_recompile)
(help <string>)	prints a description of the named function.	(help cheats_load)
(random_range <short> <short>)	returns a random value in the range [lower bound, upper bound)	(random_range 1 10)
(real_random_range <real> <real>)	returns a random value in the range [lower bound, upper bound)	(random_range 0 1)
(numeric_countdown_timer_set <long> <boolean>)	<milliseconds>, <auto_start>	(numeric_countdown_timer_set 15500 false) (numeric_countdown_timer_set 10000 false)
(numeric_countdown_timer_get <short>)	<digit_index>	(numeric_countdown_timer_get 1) (numeric_countdown_timer_get -1)
(numeric_countdown_timer_stop)	Stop the timer	(numeric_countdown_timer_stop)
(numeric_countdown_timer_restart)	Reset the timer	(numeric_countdown_timer_restart)
(breakable_surfaces_enable <boolean>)	enables or disables breakability of all breakable surfaces on level	(breakable_surfaces_enable false)
(recording_play <unit> <cutscene_recording>)	make the specified unit run the specified cutscene recording.	
(recording_play_and_delete <unit> <cutscene_recording>)	make the specified unit run the specified cutscene recording, deletes the unit when the animation finishes.	
(recording_play_and_hover <vehicle> <cutscene_recording>)	make the specified vehicle run the specified cutscene recording, hovers the vehicle when the animation finishes.	
(recording_kill <unit>)	kill the specified unit's cutscene recording.	(recording_kill player0)
(recording_time <unit>)	return the time remaining in the specified unit's cutscene recording.	(recording_time player0)

(object_set_ranged_attack_inhibited <object> <boolean>)	FALSE prevents object from using ranged attack	(object_set_ranged_attack_inhibited player0 true) (object_set_ranged_attack_inhibited player0 false)
(object_set_melee_attack_inhibited <object> <boolean>)	FALSE prevents object from using melee attack	(object_set_melee_attack_inhibited player0 true) (object_set_melee_attack_inhibited player0 false)
(objects_dump_memory)	debugs object memory usage	(objects_dump_memory)
(object_set_collideable <object> <boolean>)	FALSE prevents any object from colliding with the given object	(object_set_collideable player0 true) (object_set_collideable player0 false)
(object_set_scale <object> <real> <short>)	sets the scale for a given object and interpolates over the given number of frames to achieve that scale	(object_set_scale player0 1.5 10)
(objects_attach <object> <string> <object> <string>)	attaches the second object to the first; both strings can be empty	(objects_attach chief "right hand" ar1 "")
(objects_detach <object> <object>)	detaches from the given parent object the given child object	(objects_detach chief ar1)
(garbage_collect_now)	causes all garbage objects except those visible to a player to be collected immediately	(garbage_collect_now)
(object_cannot_take_damage <object_list>)	prevents an object from taking damage	(object_cannot_take_damage (players))
(object_can_take_damage <object_list>)	allows an object to take damage again	(object_can_take_damage (players))
(object_beautify <object> <boolean>)	makes an object pretty for the remainder of the levels' cutscenes.	(object_beautify chief true) (object_beautify chief false)
(objects_predict <object_list>)	loads textures necessary to draw a objects that are about to come on-screen.	(objects_predict the_bipeds)
(object_type_predict <object_definition>)	loads textures necessary to draw an object that's about to come on-screen.	
(object_pvs_activate <object>)	just another (old) name for object_pvs_set_object.	
(object_pvs_set_object <object>)	sets the specified object as the special place that activates everything it sees.	
(object_pvs_set_camera <cutscene_camera_point>)	sets the specified cutscene camera point as the special place that activates everything it sees.	
(object_pvs_clear)	removes the special place that activates everything it sees.	(object_pvs_clear)
(render_lights <boolean>)	enables/disables dynamic lights	(render_lights true) (render_lights false)
(scenery_get_animation_time <scenery>)	returns the number of ticks remaining in a custom animation (or zero, if the animation is over).	
(scenery_animation_start <scenery> <animation_graph> <string>)	starts a custom animation playing on a piece of scenery	
(scenery_animation_start_at_frame <scenery> <animation_graph> <string> <short>)	starts a custom animation playing on a piece of scenery at a specific frame	
(render_effects <boolean>)	Render game effects if TRUE	(render_effects true) (render_effects false)

(unit_can_blink <unit> <boolean>)	allows a unit to blink or not (units never blink when they are dead)	
(unit_open <unit>)	opens the hatches on the given unit	
(unit_close <unit>)	closes the hatches on a given unit	
(unit_kill <unit>)	kills a given unit, no saving throw	
(unit_kill_silent <unit>)	kills a given unit silently (doesn't make them play their normal death animation or sound)	
(unit_get_custom_animation_time <unit>)	returns the number of ticks remaining in a unit's custom animation (or zero, if the animation is over).	
(unit_stop_custom_animation <unit>)	stops the custom animation running on the given unit.	
(unit_custom_animation_at_frame <unit> <animation_graph> <string> <boolean> <short>)	starts a custom animation playing on a unit at a specific frame index(interpolates into animation if next to last parameter is TRUE)	
(custom_animation <unit> <animation_graph> <string> <boolean>)	starts a custom animation playing on a unit (interpolates into animation if last parameter is TRUE)	
(custom_animation_list <object_list> <animation_graph> <string> <boolean>)	starts a custom animation playing on a unit list (interpolates into animation if last parameter is TRUE)	
(unit_is_playing_custom_animation <unit>)	returns TRUE if the given unit is still playing a custom animation	
(unit_aim_without_turning <unit> <boolean>)	allows a unit to aim in place without turning	
(unit_set_emotion <unit> <short>)	sets a unit's facial expression (-1 is none, other values depend on unit)	
(unit_set_enterable_by_player <unit> <boolean>)	can be used to prevent the player from entering a vehicle	(unit_set_enterable_by_player warthog_mp_3 true) (unit_set_enterable_by_player warthog_mp_3 false)
(unit_enter_vehicle <unit> <vehicle> <string>)	puts the specified unit in the specified vehicle (in the named seat)	(unit_enter_vehicle player0 warthog_mp_2 "gunner")
(vehicle_test_seat_list <vehicle> <string> <object_list>)	tests whether the named seat has an object in the object list	(vehicle_test_seat_list ghost_mp_2 "driver" (players))
(vehicle_test_seat <vehicle> <string> <unit>)	tests whether the named seat has a specified unit in it	(vehicle_test_seat banshee_mp_1 "driver" player0)
(unit_set_emotion_animation <unit> <string>)	sets the emotion animation to be used for the given unit	
(unit_exit_vehicle <unit>)	makes a unit exit its vehicle	
(unit_set_maximum_vitality <unit> <real> <real>)	sets a unit's maximum body and shield vitality	
(units_set_maximum_vitality <object_list> <real> <real>)	sets a group of units' maximum body and shield vitality	
(unit_set_current_vitality <unit> <real> <real>)	sets a unit's current body and shield vitality	
(units_set_current_vitality <object_list> <real> <real>)	sets a group of units' current body and shield vitality	

(vehicle_load_magic <unit> <string> <object_list>)	makes a list of units (named or by encounter) magically get into a vehicle, in the substring-specified seats (e.g. CD-passenger... empty string matches all seats)	
(vehicle_unload <unit> <string>)	makes units get out of a vehicle from the substring-specified seats (e.g. CD-passenger... empty string matches all seats)	
(magic_seat_name <string>)	all units controlled by the player will assume the given seat name (valid values are 'asleep', 'alert', 'stand', 'crouch' and 'flee')	
(unit_set_seat <unit> <string>)	this unit will assume the named seat	(unit_set_seat player0 "driver")
(magic_melee_attack)	causes player's unit to start a melee attack	(magic_melee_attack)
(vehicle_riders <unit>)	returns a list of all riders in a vehicle	(vehicle_riders the_tanks)
(vehicle_driver <unit>)	returns the driver of a vehicle	(vehicle_driver the_ghots)
(vehicle_gunner <unit>)	returns the gunner of a vehicle	(vehicle_gunner the_warthogs)
(unit_get_health <unit>)	returns the health [0,1] of the unit, returns -1 if the unit does not exists	(unit_get_health player0)
(unit_get_shield <unit>)	returns the shield [0,1] of the unit, returns -1 if the unit does not exists	(unit_get_shield player0)
(unit_get_total_grenade_count <unit>)	returns the total number of grenades for the given unit, 0 if it does not exist	(unit_get_total_grenade_count player0)
(unit_has_weapon <unit> <object_definition>)	returns TRUE if the <unit> has <object> as a weapon, FALSE otherwise	(unit_has_weapon player0 plasma_cannon)
(unit_has_weapon_readied <unit> <object_definition>)	returns TRUE if the <unit> has <object> as the primary weapon, FALSE otherwise	(unit_has_weapon_readied player0 plasma_cannon)
(unit_doesnt_drop_items <object_list>)	prevents any of the given units from dropping weapons or grenades when they die	(unit_doesnt_drop_items (players))
(unit_impervious <object_list> <boolean>)	prevents any of the given units from being knocked around or playing ping animations	(unit_impervious (players) true) (unit_impervious (players) false)
(unit_suspended <unit> <boolean>)	stops gravity from working on the given unit	(unit_suspended player0 true) (unit_suspended player0 false)
(unit_solo_player_integrated_night_vision_is_active)	returns whether the night-vision mode could be activated via the flashlight button	(unit_solo_player_integrated_night_vision_is_active)
(units_set_desired_flashlight_state <object_list> <boolean>)	sets the units' desired flashlight state	(units_set_desired_flashlight_state (players) true) (units_set_desired_flashlight_state (players) false)
(unit_set_desired_flashlight_state <unit> <boolean>)	sets the unit's desired flashlight state	(unit_set_desired_flashlight_state player0 true) (unit_set_desired_flashlight_state player0 false)
(unit_get_current_flashlight_state <unit>)	gets the unit's current flashlight state	(unit_get_current_flashlight_state player0)

(device_set_never_appears_locked <device> <boolean>)	changes a machine's never_appears_locked flag, but only if paul is a bastard	(device_set_never_appears_locked <device> true) (device_set_never_appears_locked <device> false)
(device_get_power <device>)	gets the current power of a named device	
(device_set_power <device> <real>)	immediately sets the power of a named device to the given value	(device_set_power <device> 1.0)
(device_set_position <device> <real>)	set the desired position of the given device (used for devices without explicit device groups)	(device_set_position <device> 1.0)
(device_get_position <device>)	gets the current position of the given device (used for devices without explicit device groups)	
(device_set_position_immediate <device> <real>)	instantaneously changes the position of the given device (used for devices without explicit device groups	(device_set_position_immediate <device> 1.0)
(device_group_get <device_group>)	returns the desired value of the specified device group.	
(device_group_set <device_group> <real>)	changes the desired value of the specified device group.	
(device_group_set_immediate <device_group> <real>)	instantaneously changes the value of the specified device group.	
(device_one_sided_set <device> <boolean>)	TRUE makes the given device one-sided (only able to be opened from one direction), FALSE makes it two-sided	
(device_operates_automatically_set <device> <boolean>)	TRUE makes the given device open automatically when any biped is nearby, FALSE makes it not	
(device_group_change_only_once_more_set <device_group> <boolean>)	TRUE allows a device to change states only once	
(breakable_surfaces_reset)	restores all breakable surfaces	(breakable_surfaces_reset)
(cheat_all_powerups)	drops all powerups near player	(cheat_all_powerups)
(cheat_all_weapons)	drops all weapons near player	(cheat_all_weapons)
(cheat_spawn_warthog)	drops a warthog near player	(cheat_spawn_warthog)
(cheat_all_vehicles)	drops all vehicles on player	(cheat_all_vehicles)
(cheat_teleport_to_camera)	teleports player to camera location	(cheat_teleport_to_camera)
(cheat_active_camouflage)	gives the player active camouflage	(cheat_active_camouflage)
(cheat_active_camouflage_local_player <short>)	gives the player active camouflage	(cheat_active_camouflage_local_player 1)
(cheats_load)	reloads the cheats.txt file	(cheats_load)
(ai_free <ai>)	removes a group of actors from their encounter and sets them free	(ai_free the_jacks)
(ai_free_units <object_list>)	removes a set of units from their encounter (if any) and sets them free	
(ai_attach <unit> <ai>)	attaches the specified unit to the specified encounter.	
(ai_attach_free <unit> <actor_variant>)	attaches a unit to a newly created free actor of the specified type	
(ai_detach <unit>)	detaches the specified unit from all AI.	

(ai_place <ai>)	places the specified encounter on the map.	
(ai_kill <ai>)	instantly kills the specified encounter and/or squad.	
(ai_kill_silent <ai>)	instantly and silently (no animation or sound played) kills the specified encounter and/or squad.	
(ai_erase <ai>)	erases the specified encounter and/or squad.	
(ai_erase_all)	erases all AI.	
(ai_select <ai>)	selects the specified encounter.	
(ai_deselect)	clears the selected encounter.	(ai_deselect)
(ai_spawn_actor <ai>)	spawns a single actor in the specified encounter and/or squad.	
(ai_set_respawn <ai> <boolean>)	enables or disables respawning in the specified encounter.	
(ai_set_deaf <ai> <boolean>)	enables or disables hearing for actors in the specified encounter.	(ai_set_deaf the_flamers true) (ai_set_deaf the_flamers false)
(ai_set_blind <ai> <boolean>)	enables or disables sight for actors in the specified encounter.	(ai_set_blind the_jacks true) (ai_set_blind the_jacks false)
(ai_magically_see_encounter <ai> <ai>)	makes one encounter magically aware of another.	
(ai_magically_see_players <ai>)	makes an encounter magically aware of nearby players.	
(ai_magically_see_unit <ai> <unit>)	makes an encounter magically aware of the specified unit.	
(ai_timer_start <ai>)	makes a squad's delay timer start counting.	
(ai_timer_expire <ai>)	makes a squad's delay timer expire and releases them to enter combat.	
(ai_attack <ai>)	makes the specified platoon(s) go into the attacking state.	
(ai_defend <ai>)	makes the specified platoon(s) go into the defending state.	
(ai_retreat <ai>)	makes all squads in the specified platoon(s) maneuver to their designated maneuver squads.	
(ai_maneuver <ai>)	makes all squads in the specified platoon(s) maneuver to their designated maneuver squads.	
(ai_maneuver_enable <ai> <boolean>)	enables or disables the maneuver/retreat rule for an encounter or platoon. the rule will still trigger, but none of the actors will be given the order to change squads.	
(ai_migrate <ai> <ai>)	makes all or part of an encounter move to another encounter.	
(ai_migrate_and_speak <ai> <ai> <string>)	makes all or part of an encounter move to another encounter, and say their 'advance' or 'retreat' speech lines.	
(ai_migrate_by_unit <object_list> <ai>)	makes a named vehicle or group of units move to another encounter.	

(ai_allegiance <team> <team>)	creates an allegiance between two teams.	
(ai_allegiance_remove <team> <team>)	destroys an allegiance between two teams.	
(ai_living_count <ai>)	return the number of living actors in the specified encounter and/or squad.	
(ai_living_fraction <ai>)	return the fraction [0-1] of living actors in the specified encounter and/or squad.	
(ai_strength <ai>)	return the current strength (average body vitality from 0-1) of the specified encounter and/or squad.	
(ai_swarm_count <ai>)	return the number of swarm actors in the specified encounter and/or squad.	
(ai_nonswarm_count <ai>)	return the number of non-swarm actors in the specified encounter and/or squad.	
(ai_actors <ai>)	converts an ai reference to an object list.	
(ai_go_to_vehicle <ai> <unit> <string>)	tells a group of actors to get into a vehicle, in the substring-specified seats (e.g. passenger for pelican)... does not interrupt any actors who are already going to vehicles	(ai_go_to_vehicle the_jacks warthog_mp_1 "W-driver")
(ai_go_to_vehicle_override <ai> <unit> <string>)	tells a group of actors to get into a vehicle, in the substring-specified seats (e.g. passenger for pelican)... NB: any actors who are already going to vehicles will stop and go to this one instead!	(ai_go_to_vehicle_override the_jacks warthog_mp_1 "W-driver")
(ai_going_to_vehicle <unit>)	return the number of actors that are still trying to get into the specified vehicle	(ai_going_to_vehicle warthog_mp_1
(ai_exit_vehicle <ai>)	tells a group of actors to get out of any vehicles that they are in	(ai_exit_vehicle the_banshee_driver)
(ai_braindead <ai> <boolean>)	makes a group of actors braindead, or restores them to life (in their initial state)	(ai_braindead the_flamers true) (ai_braindead the_flamers false)
(ai_braindead_by_unit <object_list> <boolean>)	makes a list of objects braindead, or restores them to life. if you pass in a vehicle index, it makes all actors in that vehicle braindead (including any built-in guns)	(ai_braindead_by_unit the_ghosts true) (ai_braindead_by_unit the_ghosts false)
(ai_disregard <object_list> <boolean>)	if TRUE, forces all actors to completely disregard the specified units, otherwise lets them acknowledge the units again	(ai_disregard (players) true) (ai_disregard (players) false)
(ai_prefer_target <object_list> <boolean>)	if TRUE, *ALL* enemies will prefer to attack the specified units. if FALSE, removes the preference.	(ai_prefer_target (players) true) (ai_prefer_target (players) false)
(ai_teleport_to_starting_location <ai>)	teleports a group of actors to the starting locations of their current squad(s)	(ai_teleport_to_starting_location the_jacks)
(ai_teleport_to_starting_location_if_unsupported <ai>)	teleports a group of actors to the starting locations of their current squad(s), only if they are not supported by solid ground (i.e. if they are falling after switching BSPs)	(ai_teleport_to_starting_location_if_unsupported the_jacks)

(ai_renew <ai>)	refreshes the health and grenade count of a group of actors, so they are as good as new	(ai_renew the_jacks)
(ai_try_to_fight_nothing <ai>)	removes the preferential target setting from a group of actors	(ai_try_to_fight_nothing the_jacks)
(ai_try_to_fight <ai> <ai>)	causes a group of actors to preferentially target another group of actors	(ai_try_to_fight the_jacks the_flamers)
(ai_try_to_fight_player <ai>)	causes a group of actors to preferentially target the player	(ai_try_to_fight_player the_flamers)
(ai_command_list <ai> <ai_command_list>)	tells a group of actors to begin executing the specified command list	
(ai_command_list_by_unit <unit> <ai_command_list>)	tells a named unit to begin executing the specified command list	
(ai_command_list_advance <ai>)	tells a group of actors that are running a command list that they may advance further along the list (if they are waiting for a stimulus)	
(ai_command_list_advance_by_unit <unit>)	just like ai_command_list_advance but operates upon a unit instead	
(ai_command_list_status <object_list>)	gets the status of a number of units running command lists: 0 = none, 1 = finished command list, 2 = waiting for stimulus, 3 = running command list	
(ai_is_attacking <ai>)	returns whether a platoon is in the attacking mode (or if an encounter is specified, returns whether any platoon in that encounter is attacking)	(ai_is_attacking the_jacks)
(ai_force_active <ai> <boolean>)	forces an encounter to remain active (i.e. not freeze in place) even if there are no players nearby	(ai_force_active the_jacks true) (ai_force_active the_jacks false)
(ai_force_active_by_unit <unit> <boolean>)	forces a named actor that is NOT in an encounter to remain active (i.e. not freeze in place) even if there are no players nearby	
(ai_set_return_state <ai> <ai_default_state>)	sets the state that a group of actors will return to when they have nothing to do	
(ai_set_current_state <ai> <ai_default_state>)	sets the current state of a group of actors. WARNING: may have unpredictable results on actors that are in combat	
(ai_playfight <ai> <boolean>)	sets an encounter to be playfighting or not	(ai_playfight the_jacks true) (ai_playfight the_jacks true)
(ai_status <ai>)	returns the most severe combat status of a group of actors (0=inactive, 1=noncombat, 2=guarding, 3=search/suspicious, 4=definite enemy(heard or magic awareness), 5=visible enemy, 6=engaging in combat.	(ai_status the_jacks)
(ai_reconnect)	reconnects all AI information to the current structure bsp (use this after you create encounters or command lists in sapien, or place new firing points or command list points)	(ai_reconnect)

(ai_vehicle_encounter <unit> <ai>)	sets a vehicle to 'belong' to a particular encounter/squad. any actors who get into the vehicle will be placed in this squad. NB: vehicles potentially drivable by multiple teams need their own encounter!	
(ai_vehicle_enterable_distance <unit> <real>)	sets a vehicle as being impulsively enterable for actors within a certain distance	(ai_vehicle_enterable_distance warthog_mp_1 20.0)
(ai_vehicle_enterable_team <unit> <team>)	sets a vehicle as being impulsively enterable for actors on a certain team	(ai_vehicle_enterable_team ghost_mp_1 the_flamers)
(ai_vehicle_enterable_actor_type <unit> <actor_type>)	sets a vehicle as being impulsively enterable for actors of a certain type (grunt, elite, marine etc)	(ai_vehicle_enterable_actor_type ghost_mp_1 elite)
(ai_vehicle_enterable_actors <unit> <ai>)	sets a vehicle as being impulsively enterable for a certain encounter/squad of actors	(ai_vehicle_enterable_actors warthog_mp_1 the_jacks)
(ai_vehicle_enterable_disable <unit>)	disables actors from impulsively getting into a vehicle (this is the default state for newly placed vehicles)	
(ai_look_at_object <unit> <object>)	tells an actor to look at an object until further notice	
(ai_stop_looking <unit>)	tells an actor to stop looking at whatever it's looking at	
(ai_automatic_migration_target <ai> <boolean>)	enables or disables a squad as being an automatic migration target	
(ai_follow_target_disable <ai>)	turns off following for an encounter	(ai_follow_target_disable the_jacks)
(ai_follow_target_players <ai>)	sets the follow target for an encounter to be the closest player	(ai_follow_target_players the_jacks)
(ai_follow_target_unit <ai> <unit>)	sets the follow target for an encounter to be a specific unit	
(ai_follow_target_ai <ai> <ai>)	sets the follow target for an encounter to be a group of AI (encounter, squad or platoon)	
(ai_follow_distance <ai> <real>)	sets the distance threshold which will cause squads to migrate when following someone	
(ai_conversation <conversation>)	tries to add an entry to the list of conversations waiting to play. returns FALSE if the required units could not be found to play the conversation, or if the player is too far away and the 'delay' flag is not set.	
(ai_conversation_stop <conversation>)	stops a conversation from playing or trying to play	
(ai_conversation_advance <conversation>)	tells a conversation that it may advance	
(ai_conversation_line <conversation>)	returns which line the conversation is currently playing, or 999 if the conversation is not currently playing	

(ai_conversation_status <conversation>)	returns the status of a conversation (0=none, 1=trying to begin, 2=waiting for guys to get in position, 3=playing, 4=waiting to advance, 5=could not begin, 6=finished successfully, 7=aborted midway	
(ai_link_activation <ai> <ai>)	links the first encounter so that it will be made active whenever it detects that the second encounter is active	
(ai_berserk <ai> <boolean>)	forces a group of actors to start or stop berserking	(ai_berserk the_flamers true) (ai_berserk the_flamers false)
(ai_set_team <ai> <team>)	makes an encounter change to a new team	
(ai_allow_charge <ai> <boolean>)	either enables or disables charging behavior for a group of actors	
(ai_allow_dormant <ai> <boolean>)	either enables or disables automatic dormancy for a group of actors	
(ai_allegiance_broken <team> <team>)	returns whether two teams have an allegiance that is currently broken by traitorous behavior	
(camera_control <boolean>)	toggles script control of the camera.	(camera_control true) (camera_control false)
(camera_set <cutscene_camera_point> <short>)	moves the camera to the specified camera point over the specified number of ticks.	(camera_set somewhere_point 100)
(camera_set_relative <cutscene_camera_point> <short> <object>)	moves the camera to the specified camera point over the specified number of ticks (position is relative to the specified object).	(camera_set_relative somewhere_point 200 warthog_mp_1
(camera_set_animation <animation_graph> <string>)	begins a prerecorded camera animation.	
(camera_set_first_person <unit>)	makes the scripted camera follow a unit.	(camera_set_first_person player0)
(camera_set_dead <unit>)	makes the scripted camera zoom out around a unit as if it were dead.	(camera_set_dead player0)
(camera_time)	returns the number of ticks remaining in the current camera interpolation.	(camera_time)
(debug_camera_load)	loads the saved camera position and facing.	(debug_camera_load)
(debug_camera_save)	saves the camera position and facing.	(debug_camera_save)
(game_speed <real>)	changes the game speed.	(game_speed 0.5)
(game_time)	gets ticks elapsed since the start of the game.	(game_time)
(game_variant <string>)	set the game engine	
(game_difficulty_get)	returns the current difficulty setting, but lies to you and will never return easy, instead returning normal	(game_difficulty_get)
(game_difficulty_get_real)	returns the actual current difficulty setting without lying	(game_difficulty_get_real)
(profile_service_clear_timers)	clears the timers that are present in the profiling service	(profile_service_clear_timers)
(profile_service_dump_timers)	dumps the profiling service timers	(profile_service_dump_timers)

(map_reset)	starts the map from the beginning.	(map_reset)
(map_name <string>)	changes the name of the solo player map.	(map_name "a10")
(multiplayer_map_name <string>)	changes the name of the multiplayer map	(multiplayer_map_name "schwinnzno1_alpha01a")
(game_difficulty_set <game_difficulty>)	changes the difficulty setting for the next map to be loaded.	(game_difficulty_set easy) (game_difficulty_set normal) (game_difficulty_set hard) (game_difficulty_set impossible)
(crash <string>)	crashes (for debugging).	(crash "Something is wrong")
(switch_bsp <short>)	takes off your condom and changes to a different structure bsp	(switch_bsp 0)
(structure_bsp_index)	returns the current structure bsp index	(structure_bsp_index)
(version)	prints the build version.	(version)
(playback)	starts game in film playback mode	(playback)
(quit)	quits the game	(quit)
(texture_cache_flush)	don't make me kick your ass	(texture_cache_flush)
(sound_cache_flush)	i'm a rebel!	(sound_cache_flush)
(sound_cache_dump_to_file)	dump dat shit!	(sound_cache_dump_to_file)
(debug_memory)	dumps memory leaks.	(debug_memory)
(debug_memory_by_file)	dumps memory leaks by source file.	(debug_memory_by_file)
(debug_memory_for_file <string>)	dumps memory leaks from the specified source file.	(debug_memory_for_file "\halopc\haloce\source\tag_files\tag_groups.c")
(debug_tags)	writes all memory being used by tag files into tag_dump.txt	(debug_tags)
(profile_reset)	resets profiling data.	(profile_reset)
(profile_dump <string>)	dumps profile based on a substring.	
(profile_activate <string>)	activates profile sections based on a substring.	
(profile_deactivate <string>)	deactivates profile sections based on a substring.	
(profile_graph_toggle <string>)	enables or disables profile graph display of a particular value.	
(debug_pvs <boolean>)	displays the current pvs.	(debug_pvs true) (debug_pvs false)
(radiosity_start)	starts radiosity computation.	(radiosity_start)
(radiosity_save)	saves radiosity solution.	(radiosity_save)
(radiosity_debug_point)	tests sun occlusion at a point.	(radiosity_debug_point)
(ai <boolean>)	turns all AI on or off.	(ai true) (ai false)
(ai_dialogue_triggers <boolean>)	turns impromptu dialogue on or off.	(ai_dialogue_triggers true) (ai_dialogue_triggers false)
(ai_grenades <boolean>)	turns grenade inventory on or off.	(ai_grenades true) (ai_grenades false)
(ai_lines)	cycles through AI line-spray modes	(ai_lines)
(ai_debug_sound_point_set)	drops the AI debugging sound point at the camera location	(ai_debug_sound_point_set)
(ai_debug_vocalize <string> <string>)	makes the selected AI vocalize	
(ai_debug_teleport_to <ai>)	teleports all players to the specified encounter	(ai_debug_teleport_to the_jacks)

(ai_debug_speak <string>)	makes the currently selected AI speak a vocalization (e.g. ai_speak "pain minor")	(ai_speak "pain minor")
(ai_debug_speak_list <string>)	makes the currently selected AI speak a list of vocalizations (e.g. ai_speak_list "involuntary")	(ai_speak_list "involuntary")
(fade_in <real> <real> <real> <short>)	does a screen fade in from a particular color in the amount of ticks	(fade_in 0.0 0.0 0.0 100)
(fade_out <real> <real> <real> <short>)	does a screen fade out to a particular color in the amount of ticks	(fade_out 1.0 1.0 1.0 100)
(cinematic_start)	initializes game to start a cinematic (interruptive) cutscene	(cinematic_start)
(cinematic_stop)	initializes the game to end a cinematic (interruptive) cutscene	(cinematic_stop)
(cinematic_abort)	aborts a cinematic	(cinematic_abort)
(cinematic_skip_start_internal)		(cinematic_skip_start_internal)
(cinematic_skip_stop_internal)		(cinematic_skip_stop_internal)
(cinematic_show_letterbox <boolean>)	sets or removes the letterbox bars	(cinematic_show_letterbox true) (cinematic_show_letterbox false)
(cinematic_set_title <cutscene_title>)	activates the chapter title	(cinematic_set_title aotcr_daddy_title)
(cinematic_set_title_delayed <cutscene_title> <real>)	activates the chapter title, delayed by <real> seconds	(cinematic_set_title_delayed aotcr_rolling_title 5.0)
(cinematic_suppress_bsp_object_creation <boolean>)	suppresses or enables the automatic creation of objects during cutscenes due to a bsp switch	(cinematic_suppress_bsp_object_creation true) (cinematic_suppress_bsp_object_creation false)
(attract_mode_start)		(attract_mode_start)
(game_won)	causes the player to successfully finish the current level and move to the next	(game_won)
(game_lost)	causes the player to revert to his previous saved game	(game_lost)
(game_safe_to_save)	returns FALSE if it would be a bad idea to save the player's game right now	(game_safe_to_save)
(game_all_quiet)	returns FALSE if there are bad guys around, projectiles in the air, etc.	(game_all_quiet)
(game_safe_to_speak)	returns FALSE if it would be a bad idea to save the player's game right now	(game_safe_to_speak)
(game_is_cooperative)	returns TRUE if the game is cooperative	(game_is_cooperative)
(game_save)	checks to see if it is safe to save game, then saves (gives up after 8 seconds)	(game_save)
(game_save_cancel)	cancels any pending game_save, timeout or not	(game_save_cancel)
(game_save_no_timeout)	checks to see if it is safe to save game, then saves (this version never gives up)	(game_save_no_timeout)
(game_save_totally_unsafe)	disregards player's current situation	(game_save_totally_unsafe)
(game_saving)	checks to see if the game is trying to save the map.	(game_saving)
(game_revert)	reverts to last saved game, if any (for testing, the first bastard that does this to me gets it in the head)	(game_revert)
(game_reverted)	don't use this for anything, you black-hearted bastards.	(game_reverted)

(core_save)	saves debug game state to core\core.bin	(core_save)
(core_save_name <string>)	saves debug game state to core\<path>	(core_save_name "test1\thrusday") returns: core\test1\thursday\
(core_load)	loads debug game state from core\core.bin	(core_load)
(core_load_at_startup)	loads debug game state from core\core.bin as soon as the map is initialized	(core_load_at_startup)
(core_load_name <string>)	loads debug game state from core\<path>	(core_load_name "Fuzzy\thrusday") returns: core\Fuzzy\thursday\
(core_load_name_at_startup <string>)	loads debug game state from core\<path> as soon as the map is initialized	(core_load_name_at_startup "Sanchez\thrusday") returns: core\Sanchez\thursday\
(game_skip_ticks <short>)	skips <short> amount of game ticks. ONLY USE IN CUTSCENES!!!	(game_skip_ticks 5)
(sound_impulse_predict <sound> <boolean>)	loads an impulse sound into the sound cache ready for playback.	(sound_impulse_predict "sound\sfx\impulse\ting\ting" true) (sound_impulse_predict "sound\sfx\impulse\ting\ting" false)
(sound_impulse_start <sound> <object> <real>)	plays an impulse sound from the specified source object (or "none"), with the specified scale.	
(sound_impulse_time <sound>)	returns the time remaining for the specified impulse sound.	
(sound_impulse_stop <sound>)	stops the specified impulse sound.	
(sound_looping_predict <looping_sound>)	your mom.	
(sound_looping_start <looping_sound> <object> <real>)	plays a looping sound from the specified source object (or "none"), with the specified scale.	
(sound_looping_stop <looping_sound>)	stops the specified looping sound.	
(sound_looping_set_scale <looping_sound> <real>)	changes the scale of the sound (which should affect the volume) within the range 0 to 1.	
(sound_looping_set_alternate <looping_sound> <boolean>)	enables or disables the alternate loop/alternate end for a looping sound.	
(debug_sounds_enable <string> <boolean>)	enables or disabled all sound classes matching the substring.	
(debug_sounds_distances <string> <real> <real>)	changes the minimum and maximum distances for all sound classes matching the substring.	
(debug_sounds_wet <string> <real>)	changes the reverb level for all sound classes matching the substring.	
(sound_enable <boolean>)	enables or disables all sound.	(sound_enable true) (sound_enable false)
(sound_set_master_gain <real>)	Set the game's master gain	(sound_set_master_gain 0.5)
(sound_get_master_gain)	Returns the game's master gain	(sound_get_master_gain)
(sound_set_music_gain <real>)	Set the game's music gain	(sound_set_music_gain 3.0)
(sound_get_music_gain)	Returns the game's music gain	(sound_get_music_gain)
(sound_set_effects_gain <real>)	Set the game's effects gain	(sound_set_effects_gain 2.0)
(sound_get_effects_gain)	Returns the game's effects gain	(sound_get_effects_gain)

(sound_class_set_gain <string> <real> <short>)	changes the gain on the specified sound class(es) to the specified game over the specified number of ticks.	
(vehicle_hover <vehicle> <boolean>)	stops the vehicle from running real physics and runs fake hovering physics instead.	(vehicle_hover "vehicles\warthog\warthog" true) (vehicle_hover "vehicles\warthog\warthog" false)
(players_unzoom_all)	resets zoom levels on all players	(players_unzoom_all)
(player_enable_input <boolean>)	toggle player input. the player can still free-look, but nothing else.	(player_enable_input true) (player_enable_input false)
(player_camera_control <boolean>)	enables/disables camera control globally	(player_camera_control true) (player_camera_control false)
(player_action_test_reset)	resets the player action test state so that all tests will return false.	(player_action_test_reset)
(player_action_test_jump)	returns true if any player has jumped since the last call to (player_action_test_reset).	(player_action_test_jump)
(player_action_test_primary_trigger)	returns true if any player has used primary trigger since the last call to (player_action_test_reset).	(player_action_test_primary_trigger)
(player_action_test_grenade_trigger)	returns true if any player has used grenade trigger since the last call to (player_action_test_reset).	(player_action_test_grenade_trigger)
(player_action_test_zoom)	returns true if any player has hit the zoom button since the last call to (player_action_test_reset).	(player_action_test_zoom)
(player_action_test_action)	returns true if any player has hit the action key since the last call to (player_action_test_reset).	(player_action_test_action)
(player_action_test_accept)	returns true if any player has hit accept since the last call to (player_action_test_reset).	(player_action_test_accept)
(player_action_test_back)	returns true if any player has hit the back key since the last call to (player_action_test_reset).	(player_action_test_back)
(player_action_test_look_relative_up)	returns true if any player has looked up since the last call to (player_action_test_reset).	(player_action_test_look_relative_up)
(player_action_test_look_relative_down)	returns true if any player has looked down since the last call to (player_action_test_reset).	(player_action_test_look_relative_down)
(player_action_test_look_relative_left)	returns true if any player has looked left since the last call to (player_action_test_reset).	(player_action_test_look_relative_left)
(player_action_test_look_relative_right)	returns true if any player has looked right since the last call to (player_action_test_reset).	(player_action_test_look_relative_right)
(player_action_test_look_relative_all_directions)	returns true if any player has looked up, down, left, and right since the last call to (player_action_test_reset).	(player_action_test_look_relative_all_directions)

(player_action_test_move_relative_all_directions)	returns true if any player has moved forward, backward, left, and right since the last call to (player_action_test_reset).	(player_action_test_move_relative_all_directions)
(player_add_equipment <unit> <starting_profile> <boolean>)	adds/resets the player's health, shield, and inventory (weapons and grenades) to the named profile. resets if third parameter is true, adds if false.	
(debug_teleport_player <short> <short>)		
(show_hud <boolean>)	shows or hides the hud	(show_hud true) (show_hud false)
(show_hud_help_text <boolean>)	shows or hides the hud help text	(show_hud_help_text true) (show_hud_help_text false)
(enable_hud_help_flash <boolean>)	starts/stops the help text flashing	(enable_hud_help_flash true) (enable_hud_help_flash false)
(hud_help_flash_restart)	resets the timer for the help text flashing	(hud_help_flash_restart)
(activate_nav_point_flag <navpoint> <unit> <cutscene_flag> <real>)	activates a nav point type <string> attached to (local) player <unit> anchored to a flag with a vertical offset <real>. If the player is not local to the machine, this will fail	
(activate_nav_point_object <navpoint> <unit> <object> <real>)	activates a nav point type <string> attached to (local) player <unit> anchored to an object with a vertical offset <real>. If the player is not local to the machine, this will fail	
(activate_team_nav_point_flag <navpoint> <team> <cutscene_flag> <real>)	activates a nav point type <string> attached to a team anchored to a flag with a vertical offset <real>. If the player is not local to the machine, this will fail	
(activate_team_nav_point_object <navpoint> <team> <object> <real>)	activates a nav point type <string> attached to a team anchored to an object with a vertical offset <real>. If the player is not local to the machine, this will fail	
(deactivate_nav_point_flag <unit> <cutscene_flag>)	deactivates a nav point type attached to a player <unit> anchored to a flag	
(deactivate_nav_point_object <unit> <object>)	deactivates a nav point type attached to a player <unit> anchored to an object	
(deactivate_team_nav_point_flag <team> <cutscene_flag>)	deactivates a nav point type attached to a team anchored to a flag	
(deactivate_team_nav_point_object <team> <object>)	deactivates a nav point type attached to a team anchored to an object	
(hud_team_icon_set_pos <long> <long>)	shit	
(hud_team_icon_set_scale <real> <real>)	shit	
(hud_team_background_set_pos <long> <long>)	shit	
(hud_team_background_set_scale <real> <real>)	shit	
(cls)	clears console text from the screen	(cls)

(connect <string> <string>)	Attempt to connect to server - use ip:port password as parameters	(connect "127.0.0.1:2356" "SCEW")
(disconnect)	Disconnect from a server	(disconnect)
(hammer_begin <string> <string> <long> <short> <short>)	hammers the server by connecting and disconnecting repeatedly.	(hammer_begin "127.0.0.1:2356" "SCEW" 200 1 5) returns: hammers the sever for 200 times, and delays the disconnection between 1ms and 5ms
(hammer_stop)	stops hammering the server.	(hammer_stop)
(network_server_dump)	Dumps info on network server.	(network_server_dump)
(network_client_dump)	Dumps info on network client.	(network_client_dump)
(net_graph_clear)	Clears the net_graph.	(net_graph_clear)
(net_graph_show <string> <string>)	Changes the net_graph display (bytes/packets, sent/received)	
(play_update_history <long> <boolean>)	Playback client input history starting from the specified last completed update id.	
(show_player_update_stats)	Shows update history playback stats.	(show_player_update_stats)
(message_metrics_clear)	clears network messaging metrics	(message_metrics_clear)
(message_metrics_dump <string>)	dumps network messaging metrics to given file ("" for default)	(message_metrics_dump "")
(error_overflow_suppression <boolean>)	enables or disables the suppression of error spamming	(error_overflow_suppression true) (error_overflow_suppression false)
(structure_lens_flares_place)	places lens flares in the structure bsp	(structure_lens_flares_place)
(player_effect_set_max_translation <real> <real> <real>)	<x> <y> <z>	
(player_effect_set_max_rotation <real> <real> <real>)	<yaw> <pitch> <roll>	
(player_effect_set_max_vibrate <real> <real>)	<left> <right>	
(player_effect_start <real> <real>)	<max_intensity> <attack time>	
(player_effect_stop <real>)	<decay>	
(hud_show_health <boolean>)	hides/shows the health panel	(hud_show_health true) (hud_show_health false)
(hud_blink_health <boolean>)	starts/stops manual blinking of the health panel	(hud_blink_health true) (hud_blink_health false)
(hud_show_shield <boolean>)	hides/shows the shield panel	(hud_show_shield true) (hud_show_shield false)
(hud_blink_shield <boolean>)	starts/stops manual blinking of the shield panel	(hud_blink_shield true) (hud_blink_shield false)
(hud_show_motion_sensor <boolean>)	hides/shows the motion sensor panel	(hud_show_motion_sensor true) (hud_show_motion_sensor false)
(hud_blink_motion_sensor <boolean>)	starts/stops manual blinking of the motion sensor panel	(hud_blink_motion_sensor true) (hud_blink_motion_sensor false)
(hud_show_crosshair <boolean>)	hides/shows the weapon crosshair	(hud_show_crosshair true) (hud_show_crosshair false)
(hud_clear_messages)	clears all non-state messages on the hud	(hud_clear_messages)
(hud_set_help_text <hud_message>)	displays <message> as the help text	
(hud_set_objective_text <hud_message>)	sets <message> as the current objective	

(hud_set_timer_time <short> <short>)	sets the time for the timer to <short> minutes and <short> seconds, and starts and displays timer	
(hud_set_timer_warning_time <short> <short>)	sets the warning time for the timer to <short> minutes and <short> seconds	
(hud_set_timer_position <short> <short> <hud_corner>)	sets the timer upper left position to (x, y)=>(<short>, <short>)	
(show_hud_timer <boolean>)	displays the hud timer	(show_hud_timer true) (show_hud_timer false)
(pause_hud_timer <boolean>)	pauses or unpauses the hud timer	(pause_hud_timer true) (pause_hud_timer false)
(hud_get_timer_ticks)	returns the ticks left on the hud timer	(hud_get_timer_ticks)
(time_code_show <boolean>)	shows the time code timer	(time_code_show true) (time_code_show false)
(time_code_start <boolean>)	starts/stops the time code timer	(time_code_start true) (time_code_start false)
(time_code_reset)	resets the time code timer	(time_code_reset)
(reload_shader_transparent_chicago)	reload the transparent chicago shader	(reload_shader_transparent_chicago)
(rasterizer_reload_effects)	check for shader changes	(rasterizer_reload_effects)
(set_gamma <long>)	set the gamma	(set_gamma 200)
(rasterizer_fixed_function_ambient <long>)	set the ambient light value for fixed function	(rasterizer_fixed_function_ambient 200)
(rasterizer_decals_flush)	flush all decals	(rasterizer_decals_flush)
(rasterizer_fps_accumulate)	average fps	(rasterizer_fps_accumulate)
(rasterizer_model_ambient_reflection_tint <real> <real> <real> <real>)		
(rasterizer_lights_reset_for_new_map)		
(script_screen_effect_set_value <short> <real>)	sets a screen effect script value	
(cinematic_screen_effect_start <boolean>)	starts screen effect; pass TRUE to clear	
(cinematic_screen_effect_set_convolution <short> <short> <real> <real> <real>)	sets the convolution effect	
(cinematic_screen_effect_set_filter <real> <real> <real> <real> <boolean> <real>)	sets the filter effect	
(cinematic_screen_effect_set_filter_desaturation_tint <real> <real> <real>)	sets the desaturation filter tint color	
(cinematic_screen_effect_set_video <short> <real>)	sets the video effect: <noise intensity[0,1]>, <overbright: 0=none, 1=2x, 2=4x>	
(cinematic_screen_effect_stop)	returns control of the screen effects to the rest of the game	(cinematic_screen_effect_stop)
(cinematic_set_near_clip_distance <real>)		
(delete_save_game_files)	delete all custom profile files	(delete_save_game_files)
(fast_setup_network_server <string> <string> <boolean>)	sets up a network server with the given map name, game variant, and true for remote connections, false for not	
(profile_unlock_solo_levels)	unlocks all the solo player levels for player 1's profile	(profile_unlock_solo_levels)

(player0_look_invert_pitch <boolean>)	invert player0's look	
(player0_look_pitch_is_inverted)	returns TRUE if player0's look pitch is inverted	(player0_look_pitch_is_inverted)
(player0_joystick_set_is_normal)	returns TRUE if player0 is using the normal joystick set	(player0_joystick_set_is_normal)
(ui_widget_show_path <boolean>)	blah blah	(ui_widget_show_path true) (ui_widget_show_path false)
(display_scenario_help <short>)	display in-game help dialog	(display_scenario_help 1)
(sound_enable_eax <boolean>)	Enable or disable EAX extensions	(sound_enable_eax true) (sound_enable_eax false)
(sound_eax_enabled)	Returns true if EAX extensions are enabled	(sound_eax_enabled)
(sound_set_env <short>)	Change environment preset	(sound_set_env 1)
(sound_enable_hardware <boolean> <boolean>)	Enable or disable hardware sound buffers	
(sound_set_supplementary_buffers <short> <boolean>)	Set the amount of supplementary buffers	
(sound_get_supplementary_buffers)	Get the amount of supplementary buffers	(sound_get_supplementary_buffers)
(sound_set_rolloff <real>)	Set the DSound rolloff value	
(sound_set_factor <real>)	Set the DSound factor value	
(input_get_joy_count)	test function to return the number of joysticks enumerated	(input_get_joy_count)
(input_is_joy_active <short>)	test function to determine if an enumerated joystick is activated or not	
(input_activate_joy <short> <short>)	activates an enumerated joystick into a logical joystick slot	
(input_deactivate_joy <short>)	deactivates an enumerated joystick, freeing up the logical joystick slot	
(input_find_joystick <string>)	test function to find a joystick by GUID (string representation)	
(input_show_joystick_info)	test function to show the enumerated joystick information for all joystick	(input_show_joystick_info)
(input_find_default <string>)	test function that looks up a default profile for a deviceid	
(config_one_control <string>)	test function to configure a single control	
(get_yaw_rate <short>)	gets the yaw rate for the given player number	
(get_pitch_rate <short>)	gets the yaw rate for the given player number	
(set_yaw_rate <short> <real>)	sets the yaw rate for the given player number	
(set_pitch_rate <short> <real>)	sets the yaw rate for the given player number	
(get_digital_forward_throttle <short>)	gets the amount of forward throttle applied by digital device stimuli	
(set_digital_forward_throttle <short> <real>)	sets the amount of forward throttle applied by digital device stimuli	
(get_digital_strafe_throttle <short>)	gets the amount of strafe throttle applied by digital device stimuli	
(set_digital_strafe_throttle <short> <real>)	sets the amount of strafe throttle applied by digital device stimuli	

(get_digital_yaw_increment <short>)	gets the increment in yaw applied by digital device stimuli	
(set_digital_yaw_increment <short> <real>)	sets the increment in yaw applied by digital device stimuli	
(get_digital_pitch_increment <short>)	gets the increment in pitch applied by digital device stimuli	
(set_digital_pitch_increment <short> <real>)	sets the increment in pitch applied by digital device stimuli	
(get_mouse_forward_threshold <short>)	gets the threshold beyond which mouse movement is full forward throttle	
(set_mouse_forward_threshold <short> <real>)	sets the threshold beyond which mouse movement is full forward throttle	
(get_mouse_strafe_threshold <short>)	gets the threshold beyond which mouse movement is full strafe throttle	
(set_mouse_strafe_threshold <short> <real>)	sets the threshold beyond which mouse movement is full strafe throttle	
(get_mouse_yaw_scale <short>)	gets the scale for mouse control of yaw	
(set_mouse_yaw_scale <short> <real>)	sets the scale for mouse control of yaw	
(get_mouse_pitch_scale <short>)	gets the scale for mouse control of pitch	
(set_mouse_pitch_scale <short> <real>)	sets the scale for mouse control of pitch	
(get_gamepad_forward_threshold <short>)	gets the threshold beyond which gamepad movement is full forward throttle	
(set_gamepad_forward_threshold <short> <real>)	sets the threshold beyond which gamepad movement is full forward throttle	
(get_gamepad_strafe_threshold <short>)	gets the threshold beyond which gamepad movement is full strafe throttle	
(set_gamepad_strafe_threshold <short> <real>)	sets the threshold beyond which gamepad movement is full strafe throttle	
(get_gamepad_yaw_scale <short>)	gets the scale for gamepad control of yaw	
(set_gamepad_yaw_scale <short> <real>)	sets the scale for gamepad control of yaw	
(get_gamepad_pitch_scale <short>)	gets the scale for gamepad control of pitch	
(set_gamepad_pitch_scale <short> <real>)	sets the scale for gamepad control of pitch	
(bind <string> <string> <string>)	binds an input device/button combination to a game control	
(unbind <string> <string>)	unbinds an input device/button combination	
(print_binds)	prints a list of all input bindings	(print_binds)
(sv_end_game)	End the current game.	(sv_end_game)
(change_team <short>)	change your team (0=red,1=blue,else=auto)	(change_team 0) returns: changes you to red (change_team 1) returns: changes you to blue (change_team 2) returns: auto balance
(sv_mapcycle)	Print the contents of the currently loaded mapcycle file	(sv_mapcycle)

(sv_mapcycle_begin)	Restart or begin playing the currently loaded mapcycle file	(sv_mapcycle_begin)
(sv_mapcycle_add <string> <string>)	Usage: sv_mapcycle_add <mapname> <variantname> Add a new game to the end of the mapcycle file.	
(sv_mapcycle_del <long>)	Usage: sv_mapcycle_del <index> Removes the game at <index>. Will not affect running games.	
(sv_map_next)	<Server Only> Abort the current game and begin the next game in the playlist	(sv_map_next)
(sv_map_reset)	<Server Only> Reset the current game	(sv_map_reset)
(sv_map <string> <string>)	<Server Only> Usage: "sv_map <mapname> <variantname>" Abort current game and playlist and start specified game	
(rcon [rcon password] [command])	Sends a command for server to execute at console. Use \" to send quotes.	
(sv_rcon_password [remote console password])	Sets the server remote console password. If no password is given, displays the current password. Enter "" to disable rcon.	
(sv_say <string>)	<Server Only> Usage: "sv_say <message>" Send a message to users	(sv_say "can't see me") returns: "can't see me" to the chat window
(sv_players)	<Server Only> Print a list of players in the current game	
(sv_kick <string>)	<Server Only> Usage: sv_kick <player # or name> Kicks the specified player from the server	(sv_kick "Micro$oft") returns: kicks Micro$oft
(sv_ban [player # or name] opt:[duration (#)(m,h,d)])	<Server Only> Player is kicked and added to banned.txt. Use sv_players to find the index. Specify optional duration for timed ban. Use 0 to follow sv_ban_penalty rules.	
(sv_banlist)	Print a list of banned players	(sv_banlist)
(sv_unban <long>)	<Server Only> Usage: sv_unban <index> Removes player at index in the banlist. Use sv_banlist to find the index	(sv_unban 1)
(sv_parameters_reload)	<Server Only> Usage: sv_parameters_reload Reloads the parameters.cfg file.	(sv_parameters_reload)
(sv_parameters_dump)	Dumps out the local parameter configuration to parameters.cfg file.	(sv_parameters_dump)
(sv_status)	Shows status of the server	(sv_status)
(sv_name [name])	Sets the name of the server. If no name is given, displays the current name.	(sv_name) returns: current sever name (sv_name "yousuck") returns: "yousuck" as sever name
(sv_password [password])	Sets the server password. If no password is given, displays the current password.	(sv_password) returns: sever password (sv_password "1234") returns: "1234" as sever password

(sv_log_note <string>)	Leave a note in the server log	(sv_log_note "kornman00 was here") returns: "kornman00 was here" in the sever log
(sv_log_file [log file name])	Sets the server log file name. If no name is given, displays the current log file name.	(sv_log_file) returns: sever log file name (sv_log_file "in teh box") returns: "in teh box" as the sever log file name
(sv_log_enabled ["1" to enable, "0" to disable])	Enables or disables server logging. If 0/1 is not given, displays the current logging status.	(sv_log_enabled) returns: displays if logging is enabled or not (sv_log_enabled 1) returns: enables logging (sv_log_enabled 0) returns: disables logging
(sv_log_rotation_threshold [threshold in kilobytes])	Sets the log rotation threshold. When a log file's size (in kilobytes) exceeds this number, it will be rotated. Set to 0 to disable log rotation. If the threshold is not specified, displays the current threshold.	
(sv_log_echo_chat [preference])	Enables or disbles chat echo to the console. Set the preference to 0 to disable chat echo, or 1 to enable chat echo. If the preference is not specified, displays the current preference.	(sv_log_echo_chat) returns: displays if chat echoing is enabled or not (sv_log_echo_chat 1) returns: Enables chat echoing (sv_log_echo_chat 0) returns: Disables chat echoing
(profile_load <string>)	Load any included builtin profiles and create profiles on disk.	(profile_load a hobo) returns: loads the profile "a hobo"
(track_remote_player_position_updates <string>)	Sets the name of the remote player whose position update are to be tracked.	
(remote_player_stats <string>)	Displays the prediction stats of the specified remote player.	(remote_player_stats "player1") *use "player" than the numbers 0 - 15, zero being player 1 and 15 being player 16
(sv_get_player_action_queue_length <string>)	Displays the action queue length for the specified player.	
(thread_sleep <long>)	Sleeps the calling thread for the specified number of ms.	(thread_sleep 20) returns: sleeps for 20ms
(checkpoint_save)	save last solo checkpoint	(checkpoint_save)
(checkpoint_load <string>)	load a saved checkpoint	
(sv_maplist [substring])	Display a list of maps, matching an optional substring.	(sv_maplist) (sv_maplist "bloodgluch")
(sv_gamelist [substring])	Display a list of game types, matching an optional substring.	(sv_gamelist) (sv_gamelist "king")
(sv_friendly_fire ["0" = defaults, "1" = off, "2" = shields, "3" = on])	Use to provide a global override for the gametype friendly fire setting.	
(sv_timelimit ["-1" = default, "0" = infinite, <time in minutes>])	Use to provide a global override for the gametype timelimit setting.	
(sv_ban_penalty [(#)(m,h,d), 0=infinite])	Specify up to 4 ban times for repeat ban/TK offenders.	
(sv_tk_grace [time (#)(s,m)])	Specify the grace period for TK during which you don't get a TK point.	
(sv_tk_cooldown [time (#)(s,m)])	Specify a TK point cooldown period, after which players lose a TK point.	
(sv_banlist_file [alphanumeric banlist file suffix])	Sets and opens the file to be used for the player ban list.	

(sv_maxplayers [1 - 16])	Sets the maximum number of players (between 1 and 16). If no value is given, displays the current value.	(sv_maxplayers 10)
(sv_single_flag_force_reset [boolean])	Force the flag to reset in single flag CTF games when the timer expires, even if held by a player. If not specified, displays the current value.	(sv_single_flag_force_reset) (sv_single_flag_force_reset true) (sv_single_flag_force_reset false)
(sv_motd [motd file name])	Sets the server message of the day file name. If no name is given, displays the current motd file name. Set to "" to turn motd off.	(sv_motd) (sv_motd bloodgulch)
(oid_watch)	Sets/displays translated object table watch parameters.	(oid_watch)
(oid_dump)	dumps the whole translated object table to network.log	(oid_dump)
(oid_status)	displays the status of the translated object table.	(oid_status)

Download example script with source tags