SOF2 Scripting with ICARUS

Contents

Using BehavEd

BehavEd is a drag and drop interface for building scripts. On the left you'll see the list of ICARUS commands available to you. On the right is a column of buttons used for BehavEd functions like loading and saving your scripts, quitting, copying and pasting commands, etc.

BehavEd Command Buttons

Actions
Add – Add selected command to the script window (same as double-clicking on the command)
Delete – Delete the selected line from the script
Edit – Edit the selected line (same as double-clicking on the line)
Clone – Copy the selected line and paste the copy on the next line
Copy – Copy the selected line to the buffer
Paste – Paste the line in the buffer to the line after the currently selected line

File
New – Start a new script
Open – Open an existing script
Append – Append an existing script to the end of the one you're editing.
Save – Save your script under it's current name
Save As… - Name and save your script

Application
Preferences – Brings up the list of directories BehavEd should look fo certain files in.
About… - Some information about BehavEd
Exit – Quit BehavEd

Treeview Options
Show Types – Turns on the showing of the types of all values and strings and table entries.
%g floats – Shows all floats with several decimal places (default shows only decimal places used)

Compile! – Process your script so that it is ready to be run in the game by ICARUS.

The Feedback Window
All errors and messaged related to compiling a script will appear in the bottom white window. If a script compiled successfully you will see a smiley face and the text “OK”. Otherwise, it will report the error.

ICARUS Command List
You can add a command to your script (displayed in the large central window) by either dragging a command from this list to that window or double clicking on it. If you simply highlight a command, information about it will appear just above the Feedback Window.

Key:
{} – Indicates a command that can be expanded to contain other commands.
e – A normal command
[] – A macro, essentially a group of commands.

Commands:
(For detailed descriptions, see each command’s individual description later in this document)
affect move print

Editing Your Commands

Once you’ve added a new command to your script, you have to enter the right parameters. This is done by either double-clicking on the command or highlighting it and hitting the “edit” button.

You will be presented with a series of edit boxes that you can alter the values of.

Some are simple edit boxes where you just type in the values you desire. These will usually have an indication below the box of what type of data is expected, string, float, vector, etc.

Some are drop-boxes where you must pick a value from a list of available ones. These drop-downs will occasionally change what the subsequent edit boxes expect as data. When you’ve chosen a new drop-down value, you must hit the “Re-Evaluate” button to ensure that the extra edit boxes are updated to the right number and type.

Next to edit boxes you will see a “**Helper**” button. By clicking on this, you can access the “**get**” command (see below). Only the get fields that match the type of the expected data will be available. If the expected data type is a float you also have the choice of using the “**random**” function (again, see below). Simply type in the minimum and maximum ranges for the float value and hit the random button. If you decide not to use any of the helper functions, hit “**revert**” to return to a normal edit-box.

When you are done, hit “ok” or “cancel” if you don’t wish to apply the changes you made.

Flow Control Commands

(affect, wait, If/else, loop, random, signal/waitsignal, declare, set)

These are the commands that control the “flow” of the script. In other words, they affect how and when commands are executed.

affect – This command is used to send a block of commands to an entity other than the one running the current script. For example:

```
Print( “Hello, Fred!”)

Affect ( “Fred”, FLUSH )
{
        wait( 1000 )
        print( “Leave me alone!” )
}
```

In the above example, the entity running the script will print “Hello, Fred!”, then tell Fred to wait a second and respond with “Leave me alone!”

The name (“Fred”) that you specify must be the ICARUSname (not classname) of the entity you wish to affect. ICARUSname is a key/value pair specified in radiant. For all other entities, you must specify a name if you intend to affect them with a script at some point.

The second parameter in the “affect” command decides whether you want to totally wipe out what the entity was doing before or just insert the new command into what he was doing. The choices are:

FLUSH – Just like the “flush” command, will wipe out any script commands Fred might have been waiting to process in favor of the new ones in the “affect” block.

INSERT – Wherever Fred is in his script commands he is running, throw in these new commands. So if Fred is sitting at a table and alternating between eating and looking around, this new command will make him print “Leave me alone!” and allow him to continue eating and looking around.

It’s important to remember that an affect simply dumps commands onto another entity. The script that has the affect in it will not wait for the commands inside the affect to finish.

For example:

```
Print( “Hello, Fred!”)

Affect ( “Fred”, FLUSH )
{
        wait( 1000 )
        print( “Leave me alone!” )
}

print( “Damn you to Hell, Fred!” )
```

This will *not* wait until Fred is done talking before printing “Damn you to Hell, Fred!”. What will happen is something like this:

“Hello, Fred!”
“Take that, Fred!”

Then, a second later, Fred will say:
“Leave me alone!”

This is because “affects” complete instantly for the script the affect is in. If you wanted the reply to Fred to wait a second after Fred spoke, this is what you would do.

```
Print( “Hello, Fred!”)
```

```
Affect ( “Fred”, FLUSH )
{
        wait( 1000 )
        print( “Leave me alone!” )
}

wait( 2000 )
print( “Take that, Fred!” )
```

The reason you wait *two* seconds is because you and Fred are now running your scripts simultaneously. You have to wait for the one second before Fred speaks and wait another second before you speak.

This is a very important rule and is easy to forget. Always remember: *affects are not executed in the script you are running, they are shoved onto another entity for them to execute!*

Important: Do NOT use the affect command on the entity running the script. There is a bug somewhere in the ICARUS internals that does not properly run singly-nested affect commands affecting the calling entity. At some point, this may get fixed. But, it’s inside the ICARUS internals somewhere and NOT in the game stuff.

Wait is a simple command used to make a script wait a certain number of milliseconds before execution continues. There are 1000 milliseconds in one second.

For example:

```
print( Hello… )
wait( 1000 )
print( World! )
```

Will make the script print “Hello…”, wait a second, then print “World!”.

Random gives a random value between two numbers, so:

```
random(0,1)
```

Will give a random float between 0 and 1.

Declare allows you to set up a variable in ICARUS. A variable is something that can be set and later referenced in some other part of the script. When you declare a variable you have to let ICARUS know what type of variable it is (e.g. float). So in the example:

```
declare (  FLOAT, "choice" );
```

Choice is the variable name and it can have any numbered value. To actually assign choice some value you will use the Set command.

Set allows you to set some internal variable within ICARUS to a specified value. Usually this is used with NPCs to set their state, animation, etc.. But you can also use it to set up variables which you declared or to set parms on brushes. So, in the example:

```
set ( "choice", random( 0, 2 ) );
```

Will set the variable choice to some random number between 0 and 2. Remember that you have to declare choice (as above) before this will work.

If/else allows you to branch a script based on the value of some if-then logic test. So, if the comparison in the if statement is true, then it will execute whatever is below it. You can put an else after the if to execute a separate block of commands if this comparison is not true.
So, in the example:

```
if ( get( FLOAT, "choice") < 1 )
{
   set ( "SET_MOVE_BY", "128.0 0.0 0.0 1000.0" );
}
else (  )
{
   set ( "SET_MOVE_BY", "0.0 128.0 128.0 1000.0" );
}
```

We move the brush by one of two possible values based on the comparison in the first if statement. If the variable choice is less than one, it will move by 128,0,0 for 1 second. Otherwise, it will move by 0,128,128 for 1 second.

Signal and **Waitsignal** allow you to have one script signal another to continue. So, for example- I could have some trigger or deathscript or painscript set a signal, which some other script would be waiting to receive before continuing. However, please note that once the receiving script gets the signal that signal is cleared! Therefore, you can't have more than one script waiting on the same signal. If you need something like this- send more than one signal from the sending script.

Example: Say I want to have some npcs stay right where they are until certain other npcs are dead. We could accomplish this with a signal and waitsignal. The npcs who are dying would have this as a death script:

Chris's death script:

```
signal ( "chris dead" );
signal ( "chris dead 2" );
```

Matt's death script:

```
signal ( "matt dead" );
signal ( "matt dead 2" );
```

Marine scout's script (the guy who's waiting for the signal from Matt, Chris being dead):

```
set (  "SET_TASK",  "Attack" );
      set (  "SET_ENEMY", "matt" );
      set (  "SET_STANCE",  "Crouch" );
      rem ( "scout stays crouched until matt is dead" );
      rem ( "signaled from col6/mattdead" );

      waitsignal ( "matt dead" );
      waitsignal ( "chris dead" );
       set (  "SET_TASK_SPEED",  "3 - TakeYourTime" );
      set (  "SET_ENEMY", "NULL" );
      task ( "donekill" )
      {
         set (  "SET_TASK",  "Path" );
         set (  "SET_NAVGOAL", "donekill" );
      }
```

```
    dowait ( "donekill" );
    set (  "SET_STANCE",  "Crouch" );
```

So, the scout is signaled from the deathscripts of Matt and Chris before he continues on his script. A similar script could be setup for another npc, this npc would have to use the signals “matt dead2” and “chris dead2”

Loop allows you to repeat a sequence of actions for a specified number of times. If you put a –1 in the number of loop times, this will loop forever.
So, in the example:

```
affect ( "boat01",  FLUSH )
{
   task ( "boat01float" )
   {
       play (  "PLAY_ROFF", "scripts/col8/boatfloat.rof" );
   }
   loop ( -1 )
   {
      dowait ( "boat01float" );
   }
}
```

The boat is roffed over and over again. (see below for details on PLAY_ROFF)

Brush Manipulation Commands

(move, moveDir, moveBy)

These are commands that will allow you to physically move brushes with scripting.

The **move** command will make a brush move from its current position to a second coordinate over a length of time in milliseconds, like so:

move (<1000 2000 1000>, 1000)

will make the brush move from its current position to (1000, 2000, 1000) over 1 second.

moveDir will move the brush from it’s current position with a specified trajectory for the specified amount of time, and in the velocity it must go to travel the specified distance so:

moveDir(<0 90 10>, 50 1000) moves the brush on an angle of 0 degrees pitch, 90 roll and 10 yaw and with a velocity that is needed to travel 50 units in one second.

moveBy will move the brush from it’s current position with a specified trajectory over an amount of time, so:

moveBy<0 90 10>, 1000) moves the brush 90 units in the y, 10 in the z over 1 second.

rotate will rotate the brush from it’s current orientation in the given pitch, yaw and roll for the given milliseconds, so:

```
affect ( "object",  FLUSH )
{

task ( "rotate stuff" )
{
rotate ( < 0 90 0 >, 1000 );
}

dowait ( "rotate stuff" );
}
```

Will rotate a brush with targetname “object” to 90 degrees yaw and over 1 second.

Miscellaneous Commands

(print, rem, run, sound, use, camera, play)

Print takes a string parameter and then prints it to the top of the screen. This is intended for debugging purposes.

Rem doesn’t actually control the flow of the script, it is just a comment. You can put notes to yourself in here or a description of the action to make it easier for others to read the script, etc. It’s generally a good practice to put a rem as the first line of your script so you can tell what the script does just by looking at the first line.

Run allows one script to start running another script. This does not affect the status of the currently running script, in other words: more than one script can be running at one time.
So, in this example:

```
affect ( "Carlos",  FLUSH )
{
   run ( "scripts/example/walkpatrol" );
}
```

Carlos runs the walkpatrol script. Note that this script may or may not include instructions for Carlos.

Sound allows a script to play a sound. At the moment, only sounds which are sent to CHAN_VOICE can be dowaited. CHAN_VOICE attenuates and is based on position.
If you need the player to hear something, either affect the player and play the sound or play the sound on CHAN_ANNOUNCER. So, in this example:

```
task ( "pilot briefing 1" )
   {
      sound (  CHAN_VOICE, "sound/npc/col9/pilot/reconreports.mp3" );
   }
dowait ( "pilot briefing 1" );
```

The sound in reconreports.mp3 will be played and nothing in this script will continue until it’s finished.

Use allows you to use some entity. For this to work you must specify the **targetname.** So, in the example:

```
affect ( "barrel1",  FLUSH )
{
   use ( "barrel1_explosion1" );
   use ( "smoke_barrel1" );
   wait ( 200 );
```

barrel1_explosion1 is the targetname of an effect which is used by this script, then smoke_barrel1 is used and the script waits 200 milliseconds before continuing. You can use anything that would

normally be used in the game by (e.g. by a trigger or player, or npc) but the entity in question must have a targetname.

Camera allows you to control the camera. Here are the available commands:

ENABLE	Puts game in camera mode
DISABLE	Takes game out of camera mode
ZOOM	<zoom amount, speed> Normal zoom amount is 80, 10 is zoomed in, max is 120. Second value is time in ms to take to zoom to the new FOV
MOVE	<vector origin, speed> Move to a absolute vector origin or TAG("targetname", ORIGIN) over time over number of milliseconds
PAN	<angle, direction of movement, duration> Pan to absolute angle from current angle in dir or to a TAG('targetname", ANGLES) (no dir will use shortest) over number of milliseconds
ROLL	<angle, duration> Roll to relative angle offsets of current angle over number of milliseconds
TRACK	<trackname, speed, lerp> Get on track and move at speed, last number is whether or not to lerp to the start pos
FOLLOW	<cameraGroup, speed, lerpt> Follow ents with matching cameraGroup at angleSpeed, last number is whether or not to lerp to the start angle
DISTANCE	<distance, lerp> Keep this distance from cameraGroup (if any), last number is whether or not to lerp to the start angle
FADE	<start, opacity, end, opacity, distance> Fade from [start Red Green Blue], [Opacity] to [end Red Green Blue], [Opacity] (all fields valid ranges are 0 to 1) over [number of milliseconds]
SHAKE	<intensity, duration> Intensity (0-16) and duration, in milliseconds
PATH	<filename> Path to ROF file to be used for camera movement.

Play

Play a given ROF file. For example:

```
task ( "first heli rof" )
{
	play (  "PLAY_ROFF", "scripts/col9/heli1.ROF" );
}
do ( "first heli rof" );
```

Will play the ROF named heli1.ROF

There are two types of ROFF types: PLAY_ROFF (a normal ROF) and PLAY_ROFF_TRANSLATED.

ROFF_TRANSLATED allows you to do one generic ROFF, and the entire ROFF animation will be rotated around to fit up with the current position of the entity. So this way, if you have a generic explosion which flips a truck, the explosion will orientate itself no matter which direction the truck is facing Note that this translation has more math involved, so if you want to do a ROFF which plays over and over again (on the same object), you may encounter math precision errors which can accumulate over time. I.e. if you have a fancy path that ends at the same point where

it starts, having it translated and played many times may result in the end position being slightly off.

Debug Commands

g_ICARUSDebug 4
Is the command given to the console to display all ICARUS directives as they are parsed and pushed onto sequencers.

NPC Scripting with ICARUS and Lich

ICARUS provides an interface to AI to enable designers to script everything from very basic NPC point to point movements and animation calls, to behaviors and team goals. This document assumes you have already read the "Nav Point System".doc document in the documents/design folder.

Basic Concepts

ICARUSname vs. Targetname

We will often refer to ICARUSname and targetname. An ICARUSname is needed on any entity who will be affected or run scripts. A targetname is needed generally for any entity you want to talk about it in ICARUS or use it in ICARUS but don't necessarily want to affect. Example of things which need targetnames are ref_tag and doors. Anything you are to affect, for example a Colombian soldier needs an ICARUSname.
Note that in soldier of fortune 2 "player" is the ICARUSname and targetname of the player.

Goal Scripts vs. Regular Scripts

Another important concept to understand early on is the difference between pure AI, goal driven AI, and scripts. The best way to demonstrate this is with an example: there are three different ways to have an NPC patrol in sof2: using *just* his AI, using a Goal (this might be called an "AI Script" or "Goal Script"), and using a regular script. In the AI way of patrol, the NPC's default behavior is to patrol- which means he will automatically pick a few navpoints near him to patrol. In the Goal Script you (as a designer) would give him the "Guard" goal and a set of navpoints, these navpoints would be the points he chooses to patrol between. The third way to do a patrol is with a regular script. This last method is just like you've seen in the original sof: tell an NPC to move to one navpoint, then another, then another- perhaps all surrounded by a loop.

This is only one example of Goal Scripts vs. Regular Scripts. Basically, just remember that a Goal Script is not a pure script. The closest example of a goal script from past games would be the Star Trek EF idea of a behavior state: you give the guy some

parameters and let him go. In the above goal example, we gave him a Guard goal and some points, then from there the AI took over. This is very different from a regular script, because in a regular script the AI is doing nothing. So, if you are expecting the guy to respond to enemy gunfire for example- they would not (by default) handle this (unless you tell them to, see "RELEASE ON CONTACT" for more info. on this). Also, usually a goal script will automatically have the Sergeant do hand signals and orders before executing, a regular script would not do that for you.

We will first talk about the SET_TASK instruction. Some SET_TASK command is needed pretty much anytime you want the NPC to move based on a script. The command you choose will determine how much Lich is used to help the scripting system.

What Happens After a Script?

In general, after a script is finished the NPC will go back to his AI. However, there are several exceptions to this and things to watch out for:

- Goals are persistent and stay around until you give another goal or do a GOAL_FLUSH (see below for details)
- If you set up some field on an NPC (e.g. ENEMY, STANCE, …) and did not flush it before the end of the script, this field is still set on the NPC when he exists the script. If at some point you want to clear it, you will have to use the appropriate command (e.g SET_ENEMY NULL, see below for details).

TASKS

Move

Instantly moves NPC to a specific x,y,z on the map

Requirements:

After SET_TASK Move do a SET_VECGOAL with the given world coordinate.

Example:

```
set ( "SET_TASK", "MOVE" );
set ( "SET_VECGOAL", "50.0 1000.0 70.0" );
```

MoveToEnt

Lerps to a specific entity location.

Requirements:

After SET_TASK Move do a SET_ENTGOAL with the given targetname.

Example:

```
set ( "SET_TASK", "MoveToEnt" );
set ( "SET_ENTGOAL", "reftag1" );
```

WALK

Walks to specific NavPoint on the map. (you **don't** have to specify animations, Lich picks them for you based on velocity, etc..). If you do not want to use the default walk speed (3 out of a scale of 1 to 10) you may use SET_TASK_SPEED to change the speed of the walk and the corresponding animation to go with this speed (see SET_TASK_SPEED for more information).
NOTE: Walk does NOT use the nav system, line of sight is required for this command.
Requirements:

After SET_TASK Walk do a SET_NAVGOAL with the given name of the Navpoint (set with namenp in the game).

Example:

```
set ( "SET_TASK", "Walk" );
set ( "SET_NAVGOAL", "navpoint1" );
```

RUN

Run to specific NavPoint on the map. (you don't have to specify animations, Lich picks them for you based on velocity, etc..) If you do not want to use the default run speed (7 out of a scale of 1 to 10) you may use SET_TASK_SPEED to change the speed of the run and the corresponding animation to go with this speed (see SET_TASK_SPEED for more information).
NOTE: Run does NOT use the nav system, line of sight is required for this command.

Requirements:
After SET_TASK Run do a SET_NAVGOAL with the given name of the Navpoint (set with namenp in the game).

Example:

```
set ( "SET_TASK", "Run" );
set ( "SET_NAVGOAL", "navpoint1" );
```

PATH

Plans a path to specific NavPoint on the map. (you don't have to specify animations, Lich picks them for you based on velocity, etc..). This command will automatically generate the **quickest** route between the NPC's current location and the navpoint. In otherwords, you could specify this when the NPC is on one side of the map and the navpoint is in a completely different area, and the NPC will walk there automatically.
NOTE: PATH is the only NPC command that uses the nav system to get around objects. You can use this command to tell the NPC to get from any navpoint in the map to any other navpoint in the map as long as at least one path exists.
Requirements:

After SET_TASK Path do a SET_NAVGOAL with the given name of the Navpoint (set with namenp in the game).

You MUST have sufficient navpoints set for this to work properly. In other words, if you want him to walk to navpoint1, there better be a way for him to get from his current position to navpoint1 using the navpoint connections.

Example:

```
set ( "SET_TASK", "Path" );
set ( "SET_NAVGOAL", "navpoint1" );
```

ATTACK

Shoots NPC's gun and plays default shooting animation for the gun that NPC is holding.

Requirements:

NPC must be near an enemy. By default, any other NPC (or the player) who is not on the same team is treated as an enemy. And, executing this command on it's own will attempt to attack the closest enemy. If you wish to set an enemy manually, you may do so with SET_ENEMY (see below).

Example:

```
set ( "SET_TASK", "Attack" );
```

Or:

```
set ( "SET_ATTACK", "ON" );
```

The SET_TASK attack and SET_ATTACK ON do the exact same thing, so you can use either one. If you want to stop the NPC from shooting, you must do a SET_ATTACK OFF.

```
set ( "SET_ATTACK", "OFF" );
```

Task Related Set Commands

SET_ENEMY

Give the ICARUSname of an enemy. This is used in conjunction with the SET_TASK Attack command to have an NPC fire at a specific "player", NPC, or anything else. (Note: you can use this for anything that can have an ICARUSname: brushes, reftags, etc…) By their nature, NPCs are part of a team: so Marines are enemies of Colombians and vice versa- you do not have to set enemy for these basic relationships.

Example:

In the above FOCUS task we saw that you have no control over which enemy you focus on, you just focus on the closest one. With SET_ENEMY you can specify an exact enemy to focus on by doing the following:

```
set ( "SET_ENEMY", "Carl" );
dowait ( "enemy" );
```

```
task ( " focus on carl " )
{
        set ( "SET_TASK", "Focus" );
}
dowait("focus on carl")
task ( "flip off" )
{
        set (  "SET_HAND_SIGNAL",  "Point" );
}
dowait("flip off")
```

So as you can see: once you have enemy set you can focus on him or you could also simply attack Carl in a similar manner:

```
set ( "SET_ENEMY", "Carl" );
dowait ( "enemy" );
set("SET_TASK", "Attack");
```

Note that I do not need to set focus because the Attack will do this for me (it's pretty hard to attack someone if your back is turned to them).

SET_HITLOC

Say you want to have one NPC shoot a specific location on another NPC, You can do that with SET_HITLOC. Just make sure to set this command **before** you do a SET_TASK ATTACK. Example:

```
task ( "attack" )
{
   set (  "SET_ENEMY", "carlos" );
   set (  "SET_HITLOC",  "Left Arm" );
   set (  "SET_TASK",  "Attack" );
}
dowait("attack")
```

Having it in a task means that once the attack is complete you can clear the hit location. This way, with dowait- the designer can have a subsequent command that does a SET_HITLOC NULL. For example:

```
set("SET_HITLOC", "NULL");
```

SET_NAVGOAL,
SET_ENTGOAL,
SET_VECGOAL

These are all ways to specify a point to go to for a SET_TASK Path, Walk or Run. Navgoal takes the name of a navpoint, entgoal takes the ICARUSname of an entity, vecgoal takes a point in the world.

SET_TASK_SPEED

This lets you set the speed at which an NPC will execute a task. By default, walk is set at speed 3 (TakeYourTime), run is set at 7 (HurryUp), and Path is set at 5

(NoRush). However, you can override any of these by doing a SET_TASK_SPEED at any point in the script before the task executes.

Example:

```
task ( "Go to point1" )
{
        set (  "SET_TASK_SPEED",  "7 - HurryUp" );
        set (  "SET_TASK",  "Path" );
        set (  "SET_NAVGOAL", "nav1" );
}
dowait ( "Go to point1" );
```

SET_VIEWDIST

Controls the distance at which an NPC can see things.

SET_RELEASE_ON_CONTACT

With this set, an NPC will release from his script and go to his normal AI combat states on seeing an enemy. Note that if at some point you want the NPC to go BACK to scripting, you will have to trigger this somehow and there is no guarantee that his old script will still be around. The safe bet in this case would be to issue a flush command when you do your affect, and start over. The most common thing I can think you'd want to do is script a guy to go to a specific point, then let him do his Lich based fire fighting with some enemies, then after they are dead have him walk to another point. To do this you would want a RELEASE_ON_CONTACT in the first script. Then use signals and waitsignals (the signals sent in the death scripts of the enemy NPCs) to trigger the new script. Or using trigger counters, and make sure to use the trigger counter in each enemy deathscript, and trigger the new script based off of that.

Note that it doesn't matter where in the script "release on contact is set". But if it is set when the npc sees either a "natural enemy" (e.g. a marine is natural enemy of a Colombian) of an enemy set with SET_ENEMY he will break out of the script. By default it the SET_RELEASE_ON_CONTACT is set to off, so an npc will only break out of a script when he runs out of commands. The RELEASE_ON_CONTACT field stays set on the enemy until you set it back to off.

Note that goals (as opposed to tasks like RUN, WALK and PATH) should by default fight the enemies along the way and move on, you won't have to do RELEASE_ON_CONTACT if you're using a goal.

Other Basic Commands

SET_INGORE_PAIN

Will completely ignore pain- won't even play the animations.

SET_INVINCIBLE

Will not die, but will play pain animations

SET_STATIONARY

Tells the NPC not to move (unless you script him to). This, in conjunction with SET_TASK attack can be put into a script to make any NPC "just stand there and shoot". Of course, you can then remove the SET_STATIONARY by setting if off. For example because of some trigger, perhaps the NPC is supposed to start moving again- in this case you would then SET_STATIONARY OFF and let the NPC go about his business in the rest of the script (or go back to AI).

Example: (stand there and shoot)

```
set ( "SET_STATIONARY", "ON" );
set ( "SET_TASK", "Attack" );
```

Say you want a guy to stand around until triggered. You would set his spawn script up like this:

```
set ( "SET_STATIONARY", "ON" );
```

Then in the script which is triggered you would do this to turn off stationary

```
set ( "SET_STATIONARY", "OFF" );
```

From that point on you can script as normal. For an example, lets say you want the NPC to walk to a navpoint named rallyPoint but you want sure the NPC breaks out of the script and does AI if he sees the player. You would put this in the script which is triggered to "wake up" the NPC:

```
set ( "SET_STATIONARY", "OFF" );
set ( "SET_ENEMY", "player" );
set ( "SET_RELEASE_ON_CONTACT", "ON" );
task("Go To RallyPoint")
{
        set( "TASK", "Path")
        set ( "SET_NAVGOAL", "RallyPoint" );
}
dowait("Go To RallyPoint");
```

If I had NOT done a SET_ENEMY here the NPC would have broken out of his script when he saw ANY enemies. So for example, if I was scripting a Colombian and he saw a Marine and he did not SET_ENEMY as above, then he would attack. The above example will only break out of the script on seeing the player. (And then it will go into default AI which will do the combat for you). Also note that I have set the npc to go to a navpoint named "RallyPoint". Normally he will only "finish" this script once he has gotten to that navpoint, but since

SET_RELEASE_ON_CONTACT is set to ON, he will break out of the script and go to normal AI at any point he sees the player.

SET_NAVIGATION

This can also be set as a spawn field. Turning this off effectively disallows an NPC from using the navpoint system, but is good for snipers or people who will never move. If you DO want them to move later in life, you can turn navigation back on- and this will again allow him to use the navpoint system. USE WITH CAUTION!

Example:

```
set ( "SET_NAVIGATION", "OFF" );
```

SET_HEALTH

Set's an NPC's health to the given value. Doing a SET_ANIM_PAIN and choosing a hit location will allow you to simulate an NPC taking pain. Example:

```
set ( "SET_ANIM_PAIN", "Chest" );
set ( "SET_HEALTH", -1 );
```

Setting health to -1 will effectively kill the NPC.

SET_MOVE

Move to coordinate x,y,z over time t.

SET_MOVE_BY

Moves in direction x,y,z for for time t (at current velocity).

SET_MOVE_DIR

Moves in direction x,y,z for for time t, and distance d.

SET_ORIGIN

Instantly teleports an NPC to the given x,y,z.

SET_ANGLE

You can use this to change an NPC's model orientation. Note: this is a LAST resort. The best way to change facing is using SET_LOOK_POINT, SET_LOOK_TARGET, or also by doing a SET_ENEMY followed by a SET_FOCUS.

SET_DEATH_SCRIPT

You can set this in a script (through an affect on an NPC) or on the spawn field in the Editor. It will run the given script when the NPC dies.

SET_PAIN_SCRIPT

You can set this in a script (through an affect on an NPC) or on the spawn field in the Editor. It will run the given script when the NPC gets pain.

SET_SPAWN_SCRIPT

You can set this in a script (through an affect on an NPC) or on the spawn field in the Editor. It will run the given script when the NPC is spawned in.

SET_USE_SCRIPT

You can set this in a script (through an affect on an NPC) or on the spawn field in the Editor. It will run the given script when the NPC is used.

SET_KO_SCRIPT

The script ot use when an NPC is knocked out.

SET_PLAYER_USE_SCRIPT

The script to run when a player presses use on this NPC.

SET_TEAM

Changes the NPCs team.

Goals

Goals are primarily for team based behavior. Also, goals are more robust than tasks: they are better at handling breaking out of the script to handle enemy fire. For example, if I script a fire team to Guard- they will guard until an enemy comes by, and continue to guard after they have killed him. Goals aren't really scripts, they are ways of telling an AI when he should prefer to be doing.

Before we discuss goals we should talk about the chain of command for team based AI. A chain of command consists of a sergeant (the leader), and several fire teams beneath him, including: a strike fire team, a support fire team, a back up fire team and a recon fire team. Fire teams consist of two people: it's a buddy system and is used for the leapfrog tactics.

When you place a sergeant in your map any other soldiers and scouts are assigned under his chain of command. The squad is assigned based on proximity. So, for example- say you have two groups of marines, each made up of a sergeant, 4 soldiers and 2 scouts), one group is in a building in the middle of the map, the other is deep in the jungle. The group in the building will automatically become a squad with the sergeant as the leader, and the group in the jungle will be a separate squad. The scouts in the building will be assigned to the recon fire team, just as the scouts in the jungle will be assigned to the recon fire team of the jungle squad. The other soldiers will be paired up into fire teams based on whom they are closest to. It is OK if a soldier doesn't have a partner (in the case of only 3 soldiers place on a map), the AI can handle this case.

Understanding this chain of command is an important part of understanding how goals work.

Squad level goals (these must be given to the sergeant!)

HoldArea

HoldArea tells a team to go to an area and hold it. (For more on areas, see the LICH navpoints document.) If they are not within 100 units of the areapoint center, they will advance to this area using their leapfrog formation.
You can set the above "100 units" manually with a SET_GOALDIST. So, for example- if you do a hold area and set goal distance to 200, your team will stop within 200 units of the areapoint center. Use SET_AREAGOAL for the area that the team is to hold. An AREAGOAL is the name of the areapoint, and as of this version of the document- it MUST be within the radius of some existing navpoint.
The GOALMODE is described below in detail. Briefly here, a GOALMODE of "Rambo" tells the team not to send out a recon team to investigate, but to instead proceed immediately to the area. **NOTE: you must affect the sergeant when giving squad level goals.**

HoldArea works by sending a goal to each fireteam, which essentially tells that group of two soldiers to move up to an area using leapfrog. Once they reach this area, they will position themselves around the AREAGOAL and shoot at anything that is an enemy. This goal is used in all the Colombia team maps.

Example (from col6):

```
task ( "start01" )
  {
    set (  "SET_GOAL",  "HoldArea" );
    set (  "SET_AREAGOAL", "start01" );
    set (  "SET_GOALDIST",  "200" );
  }
dowait ( "start01" );
```

Surround

Surround is another squad level goal, which when given to the commander will instruct all his subordinates to surround a specific enemy. You must use SET_ENTGOAL to specify the ICARUSname of the Entity to surround. This is used by the enemy troops on demolition after the first explosion.

FireTeam level Goals (these can be given to any NPC)

Guard

Guard is a useful goal, it tells the NPC in question to patrol an area. Here is an example from Kam1:

```
task ( "e_guy_patrol1" )
{
   set (  "SET_GOAL",  "Guard" );
   set (  "SET_NAVGOAL", "end_patrol_a" );
   set (  "SET_NAVGOAL2", "end_patrol_b" );
}
dowait ( "e_guy_patrol1" );
```

You can set up to three NAVGOAL's. Setting NAVGOAL, NAVGOAL2, and NAVGOAL3 is only needed for the Guard goal. Another way to do a simulated guard would be like this:

```
task ( "e_guy_patrol1" )
{
   set (  "SET_TASK",  "WALK" );
   set (  "SET_NAVGOAL", "end_patrol_a" );
}
dowait ( "e_guy_patrol1" );
task ( "e_guy_patrol2" )
{
   set (  "SET_TASK",  "WALK" );
   set (  "SET_NAVGOAL", "end_patrol_b" );
}
dowait("e_guy_patrol2")
```

Notice that here I only use a set task of WALK and a SET_NAVGOAL, first for navpoint end_patrol_a then for point end_patrol_b.

The difference between the first method (one guard GOAL) and the second method (two separate walk TASKs) is that the guard goal will automatically break out of a script when an enemy comes into view, and allow the guard to take care of the threat. Also, the first method is persistant. Once he completes one pass, he will start up again, so he will visit navpoint a, then navpoint b, then navpoint a, then navpoint b, always making a circuit until told to stop (with a GOAL_FLUSH) or he sees an enemy. In the second method you would have to loop these tasks and also do a SET_RELEASE_ON_CONTACT to accomplish similar behaviors.

Follow

Follow instructs an NPC to follow another NPC (or the player). For example:

```
task ( "e_guy_follow1" )
{
   set (  "SET_GOAL",  "Follow" );
   set (  "SET_ENTGOAL", "player" );
}
```

This is the same goal as Huang in HK and the marine in the RMG demolition map use.

Stand

This is the preferred way to make an NPC stay in one spot until he notices something. He will stay in the same general area (but looks around) until he sees: a body, blood stain, footprint, or an enemy. If he hears gunfire or loud footsteps he will investigate. You don't even need a script to do this command, since it is so common there is a spawnfield called "standgoal" set this to True and the NPC in question will have a standgoal when he spawns.

Goal Related Set Commands

SET_AREAGOAL

Needed for Recon and HoldArea goals, this is an areapoint. Note: currently the areapoint MUST be within a navpoint radius. Soon to be fixed…

SET_AREAGOAL2

Needed for the flank goal mode of HoldArea, this is for the point which to send ½ of the team while the other half proceeds to the AREAGOAL. In the end, they will all meet up at AREAGOAL.

Example:

```
task ( "flank" )
   {
      set (  "SET_GOAL",  "HoldArea" );
      set (  "SET_AREAGOAL", "rallypoint" );
      set (  "SET_AREAGOAL2", "flankpoint" );
      set (  "SET_GOALMODE",  "Flank" );
   }
dowait ( "flank" );
```

SET_NAVGOAL

SET_NAVGOAL2

SET_NAVGOAL3

These are used in the Guard goal for the points at which your NPC will patrol between.

SET_GOALMODE (UNSOPPORTED)

This is used for Hold Area. It determines how the NPCs will proceed to the area: Stealth, Rambo, or Flank. Stealth tells all the team members to proceed in stealth stance and also sends the recon team (1 or more scouts) ahead of

everyone else. Rambo sends the whole team to the spot. Flank is used in conjunction with AREAGOAL2 to split off the team in two branches and have them meet up at the final AREAGOAL. The Strike fire team will be sent to the AREAGOAL and the Support will be sent to the AREAGOAL2. Once the Support fire team reaches AREAGOAL2 they will proceed to AREAGOAL and meet up with the Strike fire team there. This was used briefly, but abandoned for another approach. If you try it, proceed with caution.

SET_GOALFLUSH

This is used to turn off the goal system. For example, if you're waiting on a signal from somewhere telling you to have a new goal or a new task, then do a SET_GOALFLUSH to clear the current goal(s).

SET_GOALDIST

This is used to set the distance at which a sergeant decides his team is close enough to the provided goal. By default, this distance is 100 units. This command lets you override it:

```
task ( "hold" )
{
	set (  "SET_GOAL",  "HoldArea" );
	set (  "SET_AREAGOAL", "Town Square" );
	set (  "SET_GOALDIST", 500 );
}
```

Handling Enemies During Goals

SET_GOAL_RELEASE

When a squad is given a goal, and somewhere along the way one of the fire teams finds an enemy there are basically three ways to handle the situation in ICARUS:

1. Go

   In this solution, the squad will continue regardless of the enemy. The squad will shoot at the threat as it moves, however the men will not revert to normal AI behavior when they see the threat. An example of the usage might be for men crossing a road when the priority is to get across the road as quickly as possible. Use hold area on the point on the other side of the road, and let them leap frog across it. If at some point an enemy walks along, they will shoot at him but not stop the immediate task of crossing the road. Setting for this mode involves doing a SET_GOAL_RELEASE on the sergeant to Go. (This tells him not to care about anything as he goes about the goal). Example:

```
set (  "SET_GOAL_RELEASE",  "GO" );
task ( "hold" )
{
    set (  "SET_GOAL",  "HoldArea" );
    set (  "SET_AREAGOAL", "Town Square" );
```

}

2. Pause
   In this solution, the squad will pause for 10 seconds. While they are "pausing" they will be in their normal AI behavior of getting to cover and shooting at the enemies. The default 10 seconds is changeable with SET_GOAL_PAUSE. An example of the usage here might be to simulate a squad responding to a couple enemies in a building. They might pause for a moment to take them on, before moving on. **This is the default GOAL_RELEASE for ALL team goals.**

   Setting the pause hold area (or any other goal) involves making sure you have GOAL_RELEASE set to PAUSE on the sergeant.
3. Stop
   In this solution, the squad will stop when it sees an enemy. This means that if any member of the team spots an enemy all the squad will respond by going into normal AI combat behavior (taking cover, etc…) Once the enemies are vanquished, the squad will continue. For stop to work, you must set GOAL_RELEASE on the sarge to STOP.

Animations

You can either use the SET_ANIM command or use the appropriate stance/mood commands to set an animation. **If a stance, mood, or handsignal exists for the animation you wish to script, it is always the preferred method**. However, if it's a custom animation you will have to provie the following information in the SET_ANIM:

- Skel : Skelement to play animation on
- Prim: Primitive this animation is within. This is a hierarchy of similar animations For example: Attack, Damage, HMOVE, Inventory
- Health: Health of NPC: Full, Medium, Low, Dead
- Item : Item holding
- Mood: Happy, Bored
- Stance : Crouching, standing,…
- Name of Animation : (e.g. Fire M4)

Read this important note before script animations

SET_ANIM

It is easy to get these parameters using SklView. Open up the average_sleves skeleton in skelview and traverse the available animations, in the grey box below the animation will be the full path take this string and copy it int the SET_ANIM field in ICARUS.

Example (from col6):

```
affect ( "pilot",  FLUSH )
{
   set (  "SET_ANIM_SEQUENCER",  "OFF;
   set (  "SET_ANIM", "Body|Interaction|Full+None+Bored+Stand|Heli Piloting" );
```

}

Remember that SOF2 has a hierarchical animation system. Thus, if you set a body animation, that will effect the entire skeleton (since the body skelement is the root of the entire skeleton). However, once you've set the body animation you could(for example) go back and set a head animation. Just remember that anytime you set a higher level animation such as body, it will override anything currently playing on the rest of the skelements.

SET_ANIM_SEQUENCER

This turns the animation sequencer on or off. The important thing to remember is that any time you have the animation sequencer off you need to set up ALL the animations you want played for any given situation. When you turn the sequencer back on, you can stop setting the animations manually.

Turning Animations off

If you want to play an animation once only you can put it in a task, do the task and then set the animation sequencer back on, or override it with a new animation. If you want to turn off an anim, use the SET_ANIM_OFF command. For example:

SET_ANIM_OFF BODY

Will turn off the currently playing animation on the body.

Common Animations

For convenience we've included several commonly done animations within the animation sequencer. This is always the **preferred** way to set an animation. For example, if all you want is to crouch (and not some special crouch) then use SET_STANCE crouch.

SET_STANCE:

Specify crouch or stand here to have your NPC crouch or stand. Although you could do this as described above, this macro is the preferred way to crouch an NPC.

SET_MOOD

Specify stealth or bored to control the stance of an NPC. This is the preferred way to change the stance an NPC.

SET_HANDSIGNAL

Specify the hand signal you want an NPC to play. This is the preferred way to play hand signals.

SET_ANIM_PAIN

Used for SET_HEALTH to override which pain animation is called and simulate a pain event.

Facing and Looking

***Important note on facing, looking and animations:**

If you intend to fully control an NPC with animation and/or looking commands you will probably want to use SET_FORCE_SCRIPT ON for the duration of your fully controlled script. This completely turns off all AI functions. Without it, it's sometimes possible for the AI to stomp joint commands or animations. But note this will also wipe out all default AI behavior (which is sometimes undesirable) so you will need to turn force script off after the script.

SET_LOOK_TARGET

Give the ICARUSname of an entity to this command and the NPC will turn to face that entity. Note that the eyes have 40 degrees of freedom, so if the NPC does not have to rotate his torso, but instead can focus his eyes on the target, he will do so. Example:

```
set (  "SET_LOOK_TARGET", "look2" );
```

Set Look Point or look target to NULL to stop looking. Example:

```
set (  "SET_LOOK_TARGET", "NULL" );
```

SET_LOOK_POINT

Same as above, but looks at a point in the world. Set Look Point or look target to NULL to stop looking.

SET_LOOK_ANGLE

Same as above, but will looks at the relative angle provided in the command.

SET_LOOK_PARENT

Turn this on or off depending on if you want the parent of the given joint to activate once the NPC has maxed out his join constraints.

SET_LOOK_ROOT

Turn this on or off depending on if you want the root of the model to turn during the look.

SET_TALK_TO

This is a special look command that will activate all appropriate joints and look commands to simulate an NPC talking to another NPC. Provide the targetname to the TALK_TO. You do NOT need to specify LOOK_ROOT, LOOK_PARENT or any of the other settings required for a LOOK_TARGET, the TALK_TO is all you need for this case:

```
set (  "SET_TALK_TO", “Chris” );
```

SET_SEARCH

Set search is a way to automatically have the NPC look from side to side. We provide you with parameters to control the range of looking. Example (from col9):

```
set (  "SET_SEARCH", < 35 45 0 > );
wait ( 5000 );
```

This tells the NPC to search at a max of 35 degrees up and down, 45 degrees side to side and with a minimum of 0 degrees.

SET_SEARCH_OFF

Turns off searching.

Appendix A : SET COMMAND Listings

SET_PARM1	Set entity parm1
SET_PARM2	Set entity parm2
SET_DISCONNECT	send disconnect
SET_MAINMENU,	go to main menu
SET_CONSOLE_COMMAND	Issue this command to the console
SET_CVAR	Change this CVAR_NAME to the following VALUE
SET_STRING_PACKAGE	Name of the string package for tokens
SET_PRINT_TOKEN	Name of token to print out as subtitle
SET_MUSIC_PERCENTAGE	Set the music to be this percetage of what it currently is (e.g. .5 will turn it down 1/2 of current volume)
SET_HUD_MAIN	name of Hud to set
SET_HUD_WEAPONS	display weapon hud
SET_PLAYERHEALTH_OLD	call this if you want to set the player health to g_playerHealthOld
SET_PLAYERARMOR_OLD	call this if you want to set the player armor to g_playerArmorOld
SET_SAVE_PLAYERHEALTH	call this if you want to save the player health to g_playerHealthOld
SET_SAVE_PLAYERARMOR,	call this if you want to savethe player armor to g_playerArmorOld
SET_WORLD_DEATH_PAUSE	call this to freeze the world, intended for death scripts (ideal after fading out)
SET_SPAWNSCRIPT	Script to run when spawned. For any SET_XXXSCRIPT you can set to NULL to clear the set script.
SET_USE_SCRIPT	Script to run when used inside game (not by player)
SET_PLAYER_USE_SCRIPT	Script to run when player uses
SET_DEATH_SCRIPT	Script to run when dead

SET_PAIN_SCRIPT	Script to run when in pain
SET_SPAWN_SCRIPT	Script to run when spawned
SET_VIEW_SCRIPT	Script to run when NPC sees enemy
SET_ANY_VIEW_SCRIPT	Script to run when NPC sees **any** NPC or player
SET_PLAYER_VIEW_SCRIPT	Script to run when NPC sees player
SET_HEAR_SCRIPT	Script to run when NPC hears any enemy or player
SET_KO_SCRIPT	Script to run when NPC is knocked out
SET_CONTACT_SCRIPT	Script to run when NPC sees enemy, hears enemy or has evidence of an enemy's presence.
SET_WORLD_HEAR_SCRIPT	Script to run first time player sound is above 0.7 on the PADD
SET_PLAYER_CAM_DEATH,	Turn on/off the player death camera (roff for dropping player to ground)(defaults ON)
SET_NAME	Set/change npc targetname
SET_CAMERA_GROUP	Set/change your cameraGroup for camera FOLLOW commands
SET_FACE_TARGET	Set/change your face target (NOT supported for npc's)
SET_ENEMY	Set/change your enemy
SET_HITLOC	Set/change your hit location (for shooting other NPC's and players)
SET_FAKEGUN	Set/change targetname of fake gun (for using SET_HEALTH to kill NPCs)
SET_PLAYER_VIEW	Attach camera to brush model child (on/off)
SET_VIEW_PARENT	Set targetname of brush model for SET_PLAYER_VIEW command
SET_VIEW_CHILD	Set targetname of brush model child for SET_PLAYER_VIEW command
SET_INVIEW_WPN	Toggle inview weapon model (on/off)
SET_ADD_WPN	Add a weapon to the player's inventory
SET_REMOVE_WPN	Remove a weapon from the player's inventory
SET_NOCLIP	Turn noclip on/off
SET_IGNOREPAIN	NPC will ignore being in pain
SET_SOUNDBLOCK	NPC will not play any sounds
SET_INVINCIBLE	NPC will ignore being in pain
SET_STATIONARY	Turn npc stationary mode on or off

SET_FORCE_SCRIPT	Turn force script mode on(1) or off(0)
SET_HIDE	Turn npc hide mode on or off
SET_ENABLE	Enable/disable affected device (not for NPCs!!!)
SET_EMPLACED	Force this entity onto emplaced weapon with targetname, NULL turns off
SET_RELEASE_ONCONTACT	NPC will be released from script when he sees or is shot by some enemy
SET_GOAL_RELEASE	GO: ignore enemies when in goal, PAUSE: default combat AI for a little while, STOP: default combat AI until enemies gone, default is PAUSE, set pause time with
SET_GOAL_PAUSE	NPCs will stop for this many ms and fight bad guys
SET_NAVIGATION	Turn navpoint usage on(1) or off(0)
SET_TASK_FLUSH	Set to ON to flush all pending script tasks
SET_ROFF_TRANSLATED	Set to ON and when the camera is roffed it will play it translated
SET_PLAYERALWAYS_HURT	Turn ON ability for player to always hurt this npc, regardless of invicible setting
SET_SUIT	Turn on/off the biohazard suit fullscreen effect
SET_MORELIGHT	Turn ON to give this npc a minlight effect
SET_ORIGIN	Set origin explicitly or with TAG
SET_ANGLES	Set angles explicitly or with TAG
SET_MOVE_BY	Move by vector x,y,z for time t
SET_MOVE	Move in direction x,y,z for d distance and t time
SET_WAIT	Change an entity's wait field
SET_VIEWDIST	Change an NPC's viewing distance
SET_GOALDIST	Change a acceptable goal reached distance (set on commanders)
SET_RESET_ALL_VIEWDIST	Set all NPC's view distances to this value
SET_ACCURACY	Accuracy of NPC
SET_DAMAGE	Damage of NPC
SET_HEALTH	Change health
SET_HEALTHNOANIM	Change health but don't play animation for it

SET_SAVEHEALTH	Saves the health out to prepare for level load, cvar name: health_icarusname
SET_LOADHEALTH	Loads the health from a previous level, cvar name: health_icarusname
SET_MAXSHOTS	Shots the entity can take before starting to ignore pain, use -1 to not use maxshots
SET_ANIM_SEQUENCER	Turns Animation Sequencer on/off
SET_ANIM	Plays an animation this many times (use string provided from SklView)
SET_ANIM_PAIN	When killing/hurting a guy in a script, you can set his death/pain animation here
SET_ANIM_OFF	Turn animations on this skel off (back to default)
SET_ANIM_FLUSH	Flush all animations (returns all to stationary)
SET_ANIM_LOOP	Loop currently cached animation
SET_ANIM_EYES	Set to OFF to stop blinking (use only in special cases- see Ben)
SET_MOOD	Changing mood changes the default animations of the NPC. E.G. soldier mood vs. stealth mood.
SET_STANCE	Have an NPC sit, stand, crouch. This combined with mood afraid gives unique afraid animations.
SET_HAND_SIGNAL	Play a hand signal once
SET_TASK	Set/change your current task
SET_GOAL	Set/change your team goal (for TeamCommander NPCs)

SET_GOALFLUSH	Set to on(1) to flush goals. Set to off(0) to again accept goals.
SET_NAVGOAL	Set/change your navgoal
SET_NAVGOAL2	Set/change your 2nd navgoal (mostly for designer assisted Patrols)
SET_NAVGOAL3	Set/change your 3rd navgoal (mostly for designer assisted Patrols)
SET_AREAGOAL	Set/change your area goal
SET_AREAGOAL2	Set/change your secondary area goal (mostly for flank area goal)
SET_VECGOAL	Set an arbitrary point on the map (x,y,z) to go to.
SET_ENTGOAL	Set to go to an entity the map.
SET_ENTGOAL2	Backup ent goal, good for elevators and small places.
SET_ENTGOAL3	Backup ent goal
SET_ENTGOAL4	Backup ent goal
SET_INVENTORY_ACTION	Set the type of action to do on the inventory task
SET_TASK_SPEED SET_TASK (e.g. walk)	Set the speed to use in a specified
SET_TASK_PAUSE	Set the pause time between navpoints
SET_ATTACK	Set to on(1) to attack (same as SET_TASK ATTACK), off(0) to stop attacking
SET_TEAM	Set this NPC or Player to the given team.

SET_BLOCK_RETRY	Set the number of blocks needed before a replan on MoveToEnt- 0 for never replan
SET_LOOK_TARGET	Set/change your current look target (look at this entity)
SET_LOOK_POINT	Set/change your current look target to be this point in space
SET_LOOK_ANGLE	Set/change your current look target by this degree (e.g. look over 10 degrees)
SET_LOOK_TYPE	Use this to set the type of looktarget, lookpoint or lookangle you want.
SET_LOOK_OFF	Use this to turn off a looking command for the given type
SET_LOOK_PARENT	Turn on/off parent activation of joint commands (once set, this stays on entity until set again. Default is ON)
SET_LOOK_ROOT	Turn on/off root activation of joint commands (once set, this stays on entity until set again. Default is ON)
SET_AIM	Aim Gun at Current Focus
SET_SEARCH	Set/change your search angles
SET_SEARCH_OFF	
SET_TALK_TO	Set/change your current talking to target (turn towards this entity to talk to it)
SET_SCOTT_MCNUTT	Switch Scott McNutt Off or On (use with caution). Actually, this has a vial appear in a man's hand and then he throws it on the floor in anger.
SET_SPECIAL_USE	Set to targetname to use (must be inside Affect of some NPC)

SET_VIOLENCE_USE	Set to targetname to use (this is a use that will only happen if the violence lock is not on)
SET_CAMERA_NOHUD	Set to true to not allow enabling of camera to letterbox screen
SET_TOUCH	Touch this entity
SET_GOALMODE	Set/change your team goal mode (for TeamCommander NPCs). NOT supported.