



Ya3dag

Scripting language
The source of intelligence

Based on V2.19 release of November 27, 2022

Edition history

8.06.2004 RR: First edition.
10.06.2004 RR: DispEingebung
13.06.2004 RR: Actor-Variable PlayerSkills@...
6.11.2004 RR: Actor-Variable PlayerJobs@...
5.08.2008 RR: Start translation to English.
16.08.2008 RR: Done with translation to English.
19.08.2012 RR: Updated based on V1.40 release of August 19, 2012.
17.08.2014 RR: Added tabulators
03.03.2018 RR: Updated based on V2.00 release of March 4, 2018.
07.05.2018 RR: Updated based on V2.01 release of May 7, 2018.
20.07.2019 RR: Updated based on V2.10 release of July 21, 2019.
23.11.2019 RR: Updated based on V2.12 release of November 24, 2019.
31.10.2020 RR: Updated based on V2.14 release of November 1, 2020.
09.04.2021 RR: Updated based on V2.16 release of April 11, 2021.
01.12.2021 RR: Updated based on V2.17 release of December 1, 2021.
01.06.2022 RR: Updated based on V2.18 release of June 1, 2022.
27.11.2022 RR: Updated based on V2.19 release of November 27, 2022.

Table of contents

- **Introduction**
- **Types of scripts**
 - Actor scripts
 - Level scripts
 - Function scripts
 - Player Scripts
- **Elements of the script language.**
 - Comments
 - Sections
 - Script commands
 - Arithmetic expressions
 - * Operators
 - * Operands
 - Numbers
 - Strings
 - Variables
 - Functions
- **Predefined variables**
- **PlayerSkills**
- **PlayerJobs**
- **Text modifiers**
 - Character modifiers
 - Tabulators
- **Script Commands**
 - Overview script commands
 - Assignment to variables
 - If commands
 - Loop commands
 - Sleep command
 - Detailed description of script commands

Introduction

At startup of a level, scripts are given to misc_actor (or other objects) as an argument. They determine the reaction of such an object to events. Scripts make an object smart.

Writing scripts is something for advanced people. You need knowledge of programming languages. In addition, you need experience in dealing with Quake2 objects.

Do you know how a targetname is used? OK, read on in this documentation. If not, look at the Lazarus documentation or learn more about this at an other place.

To get access to game scripts and documentation files from the project, unzip the game data in [Ya3dag\BaseQ2\Q2T_BaseQ2.pkz](#) and [Ya3dag\RRGame\Q2T_RRGame.pkz](#). Rename the .pkz file extension to .zip and unzip it into [Ya3dag\BaseQ2](#) respectively [Ya3dag\RRGame](#).

Types of scripts

Scripts are located in the [gamedata](#) subdirectory. There are files with the extension .txt. To work with these files, use the NotePad editor (or something similar).

- **Actor scripts**

Use **AScr** suffix for this type of scripts (like [AScr_Cal_GhostCastle.txt](#)). Used for misc_actor objects.

- **Level scripts**

Use **LScr** suffix for this type of scripts (like [LScriptWGXmas.txt](#)). This type of script is executed at startup of a level from the worldspawn object.

[LScriptEveryLevel.txt](#) is executed startup of each level and is intended as base initialization of variables (such as player skills). The only usable section is [Startup].

Thereafter the script file assigned to the [name](#) setting of the worldspawn object (by the level editor) is executed.

Additional to the [Startup] section the ,clock` and ,timer` can be used. Use this type of script to give some initial items to the player or to assign and monitor level quests.

- **Function scripts**

Use **FScr** suffix for this type of scripts (like [FScr_GBe_Hitlist.txt](#)). Used for func_script objects.

- **Player Scripts**

Use **PScr** suffix for this type of scripts (like [PScr_XXX_Default.txt](#)). Used by player_script objects.

Elements of the script language

- Comments

Comments start with either the ";" Characters or with the characters "//".

- Sections

The name of a section is enclosed by square brackets and starts at begin of a line. The code in a section is executed at occurrence of an associated event. All script commands in this section are executed until begin of the next section or the end of the file.

[EventArg1](#) and [EventArg2](#) are two parameters that are specified when a section starts executing.

Section names and related events:

Section	EventArg1	EventArg2	remark
[Startup]			First execution of a script.
[ActorWayEnd]	targetname		End of a waypoint movement.
[ActorUsed]	targetname	classname	For Actor scripts, Actor was used from entity „classname“.
[ActorUsed]	targetname	PlayerX	For Level scripts, trigger name 'worldspawn' was triggered by player X.
[ActorUsed]	targetname	PlayerX	For Level scripts, trigger name 'worldspawn' was used by an actor flagged as bot (with edict nr. X).
[ClockTick]			Clock.
[DialogCancel]			A dialog was closed.
[ActorPain]	classname		Object/Actor was injured.
[ActorDead]	targetname	classname	Actor died.
[EnemyOn]			Object has an enemy. * AI_STAND_GROUND is removed * AI2_SLEEPING is removed * Execute script command „Weaponon
[EnemyOff]			Enemy is gone. * Execute script command „Weaponoff“
[PlayerTouch]	PlayerX	PlayerY	The player has touched an Actor. X is the number of the player (1 ...). In multiplayer games the level script also gets „PlayerY“ if player X touches player Y.
[PlayerUse]	PlayerX		The player in near an actor and pressed the ‚use‘ key. X is the number of the player (1 ...).
[BotTouch]	PlayerX	"Player"	An actor flagged as bot has touched the player X.
[BotTouch]	EdictX	"Bot"	An actor flagged as bot has touched an other bot actor with edict nr. X.
[StealItem]	Item		The player has an item stolen from an Actor. Item is the classname (ammo_rockets, item_quad, ...) of the stolen object.
[TargetActor]	targetname	target	A target_actor with name „target name“ has been reached.
[xxx]	target name		Script command „trigger“ from an other script.
[xxx]			A target_actor with targetname xxx has been reached.
[xxx]			Answer of a dialog.

- **Script commands**

To a certain action. Only one script command per line.
There is a chapter on script commands at the end of this documentation.

- **Jump targets**

The goal of a `goto` script command.
Jump targets have a colon at the end and must be at the beginning of the line.

- **Arithmetic expressions**

Whenever necessary, spaces separate part of one instruction from another and allow the parser to recognize where an element is in an instruction, such as `int`, ends and the next element begins (e.g. `int age`).

Operators

Operators	Associativity	Remark
- + ! ~	Right to left	unary operators
* / % ..	Left to right	Multiplicative operators
+ -	Left to right	Additive operators
<< >>	Left to right	Shift operators
< <= > >=	Left to right	Relational operators
== !=	Left to right	Relational operators
&	Left to right	Bitwise operator
^	Left to right	Bitwise operator
	Left to right	Bitwise operator
&&	Left to right	Logical operator
	Left to right	Logical operator, Lowest precedence

Operators are listed in descending order of precedence. If several operators appear on the same line or in a group, they have equal precedence.

.. Random operator, number between 1. and 2. operand.

If both operands are not numeric, it's assumed that the operands are text:

+ Concatenate text strings
 == != < > <= >= Text comparison

Operands

Numbers

The usual „C“-style floating point and integer numbers are useable.
Hexadecimal (`0x` suffix), binary (`0b` suffix), and octal (`0` suffix) notation are supported for integers.

Examples:

```
60          // Decimal based integer number, has no leading zero
0x3c       // Hexadecimal based integer number, has a leading 0x
0b00111100 // Binary based integer number, has a leading 0b
074       // Octal based integer number, has a leading 0
-10       // Negative integer number
0.75      // Floating point number
1.45E10   // Floating point number
```

Strings

Strings are enclosed in double quotes. If the next line is also a string, this strings are concatenated (with an additional new line character between).

Use the character sequence `\n` for a new line (used for text output).

A `$` character followed by the name of a variable is replaced by the value of the variable.

Examples:

```
"Hello World"          // single line string

"Hello"                // double line string
"World"                // with new line character between

"Hello\nWorld"         // Same as above

"My Name is $This.Name." // Reference the name of an entity
```

Variables

A variable is a name given to a storage area that our script can manipulate. Each variable has a specific type, which determines the size and layout of the variable's memory.

The name of a variable can be composed of letters, digits, and the underscore character (an **identifier**). It must begin with either a letter or an underscore. A name can have a maximum length of 71 characters.

There is no difference between upper and lower case letters because the script language is case insensitive.

The following types are known:

```
string holds a string with up to 511 Characters
int     holds a 32 bit signed integer value
float   holds a 64 bit floating point value
```

Variables are declared by a line beginning with the keyword `int`, `float`, or `string`, followed by names separated by commas.

Optionally a `const` keyword can be added. In this case the variable can only be read. An initial value can also be assigned, otherwise the variable has a value of 0.

Examples:

```
int xxx          // A single variable
int const xxx    // A read only variable
int xxx = 1      // Assign a initial value
int a, b, c, d   // Multiple variable declations
int a = 1, b=2, c, d=4 // All together
```

A single **identifier** used in a variable declaration results in a **local variable**. They can only be used in the script in which they were declared.

Global variables are declared by linking two identifiers with a period. They can be used from any script. In this manual, these types of variables are sometimes referred to as "**grouped variables**".

Some **predefined variables** are also grouped together. These are described later in a separate chapter.

Examples:

```
int Global.State                               // A single global variable
int const Global.State_Init = 0               // A read only variable
int const Global.State_Busy = 1
int const Global.State_Done = 2
Global.State = Global.State_Init              // Assign a value
if Global.State == Global.State_Init          // Test for a specific value
```

The first identifier of a [global variable](#) is also known as the [group name](#). The second identifier is then used for the members of the group.

The following [group names](#) are used for special purposes:

```
ThisLevel           is a shortcut for this level.
                    This is unique for each specific level.
ThisScript          is a shortcut for this script file.
                    All scripts with the same name that are used by
                    different entities can access it.
ThisLevelScript     is a shortcut for all scripts with this name
                    in this level.
```

Functions

Functions have the format: [Function-Name](#)([Argument](#))

[itemIsInGroup](#)([ItemGroup](#) [ItemTest](#))

Test an item to be the member of a specific item group.
[ItemGroup](#): Classname of an item group.
[ItemTest](#): Classname of an item. This is tested be a member of the item group.

Return: 1 'ItemTest' is member of 'ItemGroup'
0 'ItemTest' is NOT member of 'ItemGroup'
-1 Error, 'ItemGroup' is no known item
-2 Error, 'ItemGroup' is no grouping item
-3 Error, 'ItemTest' is no known item

[itemIsDye](#)([Argument](#))

[Argument](#) is an item class name. Test argument to be a dye.
Returns the dye code if there is a match else -1 is returned.

[itemWoolByIdx](#)([Argument](#))

[Argument](#) is a number in the range 0 .. 15, a dye code.
Returns the item name of an colored wool block.

[lround](#)([Value](#))

Round a float value to nearest integer and return this value.

[MobsNearbyCount](#)([MobType](#) [DistanceXY](#) [DistanceZ](#))

Count the mobs in the near of the calling entity.
[MobType](#): Can be 'Hostile', 'Passive', 'Ambient', 'Water' or 'All'.
[DistanceXY](#): Count mobs within this xy distance.
[DistanceZ](#): Count mobs within this z distance.
Returns the number of mobs in the near.

[random](#)()

Returns a floating point number in the range 0.0 .. 1.0.

[RandomRangeInt](#)([min](#) [max](#))

Both argument are integers.
Returns an integer in the range 'min' .. 'max'.

[StrHasSubString](#)([Argument1](#) [Argument2](#))

Test [Argument1](#) to have the substring [Argument2](#).
Returns 1 (true) or 0 (false) depending on the test result.

Predefined variables

The following predefined variables are grouped together using some predefined **group names** (see previously under **global variable**). In the **R/W** column it is noted whether the variable can be **Read** and/or **Written**.

Group This

The variables in this group relate to the edict (actor, mob, entity, ...) to which this script is bound.

There is also the possibility of indirect access to the variables of an edict. If **EventArg1**, **EventArg2** (section execution parameters) or the name of a **local variable** is used as the group name, the variables of an object can be accessed via their value. In this case the content of the variable must be the text **'Edict'** or **'Player'** followed by a number. In addition, the number must be in the range from 1 to the maximum number of edicts in the game. Events like **'ActorUsed'** or **'BotTouch'** use such values for the section parameters.

Name	Type	R/W	Description
Health	int	R/W	Health of entity
MaxHealth	int	R	Maximum value of health
Name	string	R	Name of entity
OriginX	float	R	Current X position of entity
OriginY	float	R	Current Y position of entity
OriginZ	float	R	Current Z position of entity
OnTheWay	int	R	Entity is walking (moving) to a waypoint Value 0 none, > 0 number of waypoints to go
EventArg1	string	R	Event argument 1
EventArg2	string	R	Event argument 2
StartupArg	string	R	Startup argument given to script at start
ScriptName	string	R	Name of the script file
ScriptLineNr	int	R	Source line nr. (of this statement) in the script file
targetname	string	R	Contents of the targetname field of this object
target	string	R	Contents of the target field of this object
HaveFreezed	int	R	Object (Actor) is freezed
HaveDucked	int	R	Object (Actor) is ducked
HaveParalysed	int	R	Object (Actor) is paralysed
HaveSleeping	int	R	Object (Actor) is sleeping
HaveInvisible	int	R	Object (Actor) is invisible
HaveInfected	int	R	Object (Actor) is infected
HaveGoodGuy	int	R	Object (Actor) is a good guy
HaveBadGuy	int	R	Object (Actor) is a bad guy
HaveEnemy	int	R	Object (Actor) has an enemy
HaveFollowPlayer	int	R	Object (Actor) follows the player
HaveFollowAny	int	R	Object (Actor) follows a player or an other actor
IsPlayer	int	R	Object isplayer else Actor, Bot, ...
VoxLightSun	int	R	Brightness of the sun at the location of the object (actor). Range is 0 .. 255. 0 is no sun, 255 is maximum sunlight. The value depends on the time of day and the shadow at the location of the object.

Group Player

The variables in this group refer to the player (single player game) or to the nearest player (multiplayer game).

Name	Type	R/W	Description
Health	int	R/W	Health of the player
MaxHealth	int	R	Maximum health value of the player.
Name	string	R	Name of the player
Mana	int	R	Mana of the player
Money	int	R	Money of the player
ItemSelected	string	R	Classname of selected item (the player hold this item in the hand)
Distance	int	R	Distance to (nearest) player
InGame	int	R	1 if selected player is in the game, otherwise 0
Infected	int	R	1 if selected player is infected, otherwise 0
DialogOpen	int	R	Dialog open: 0 = no, 1 = yes (this Actor), -1 = yes (other Actor)

Group Game

The variables in this group refer to the player (single player game) or to the nearest player (multiplayer game).

Name	Type	R/W	Description
PlayerMax	int	R/W	Maximum number of players (multiplayer games)
PlayerCurr	int	R	Number of players in the game (multiplayer games)
BotCurr	int	R	Number of Bot's in the game (multiplayer games)
FrameNum	int	R	Frame number (counts 1 for each 1/10 seconds since start of game)
Time	float	R	Elapsed time since the start of the game in seconds
MapName	string	R	Filename of current loaded map (without file extension)
NextMap	string	R/W	Next map loaded on level change
HourAP	float	R	Virtual hour within the day (0 .. 23) with after point digits
HourNr	int	R	Virtual hour within the day (0 .. 23)
DayNr	int	R	Virtual day, counted since start of the game (1 ..)
Skill	int	R	Skill level, 0: easy, 1: normal, 2: hard, 3:hard+
HaveMultiplayer	int	R	1 if we are in a multiplayer game
LastResult	int	R	Result of last executed script command

PlayerSkills

'PlayerSkills', the skills of the player. These are some predefined global variables.

Range of each variable is 0 .. 100.

The dialog **skill** shows all members of the variables in the group PlayerSkills with their value.

PlayerSkills are usually preset in the level script 'LScriptEveryLevel.txt'.

Queries in the script as follows:

- ```
 if PlayerSkills.Intuition > 30
 ...
 endif
```

Set new value as follows:

```
PlayerSkills.Intuition = PlayerSkills.Intuition + 10 ; count up
if PlayerSkills.Intuition > 100 ; over limit
 PlayerSkills.Intuition = 100 ; clip to limit
endif
```

or with the skill command

```
skill "Magic" 7.0 ; increase magic
```

The following skills are defined (until now):

- **Intelligence**

The intelligence will be increased by solving puzzles and is also required to solve puzzles.

- **Perseverance**

Perseverance is reduced during fighting, while running, while climbing or swimming. It will also increased by doing this actions.

- **Strength**

Strength is increased by fighting and is needed for carrying goods and for fighting.

All items have a weight (only magic and money have none). The amount of things the player can carry depends from this skill.

Missing code: ==> use skill for hand fighting, sword or kick jumps.

Missing code: ==> If weight exceeds the maximum, slow speed.

- **HitTheTarget**

Accuracy in shooting.

- **Negotiations**

Negotiating skills (or communication skills) is needed when talking with others to find out certain things and the purchase of goods of all kinds. This skill is used in scripts only.

- **Intuition**

Important for decisive support in the game. Increase by solving secrets.

An object target\_secret increases this skill by 1 (secret found).

An object target\_secret can test against a minimum intuition value (health) to show you a thought bubble.

- **Magic**

Skill in the use of magic. Is increased by the use of magic.

The order of dialogs with the teacher in the school of magic depends on

this skill. Will be used in scripts (sufficient magic ability).

- **Curing**

This ability is necessary for the restoration of vital energy for themselves or others. Some remedies must only be taken. For healing magic enough Mana is needed.

Missing code: ==> to use skill or increase skill.

- **Protection**

This skill reduces the effect of hits in battle. For protection there is armor, shields and spells.

Missing code: ==> to use skill or increase skill.

- **CombatSkill**

Increases in the ghost level in the fight-arena.

- **Jobs**

Will be increased by 1 when a job is done.

In dialogs, the following classification is used:

| Points   | Ability                       |
|----------|-------------------------------|
| 0 - 20   | Novice                        |
| 20 - 40  | Apprentice                    |
| 40 - 60  | Journeyman                    |
| 60 - 80  | Preferred companion           |
| 80 - 100 | Representatives of the master |
| 100      | Masters                       |

## **PlayerJobs**

PlayerJobs stands for jobs/quests that are handed over to the player. The job dialog displays the text of all members of the PlayerJobs variables.

Query the state of a job: :

```
JobState "JobName"
if Game.LastResult
...
```

JobState returns the value

- 0 "JobName" was not given to the player (the job is not pending).
- 1 "JobName" was given to the player, but is not yet done.
- 2 "JobName" is done (completed).

Give a job to the player:

```
JobState "JobName" "JobText"
```

"JobText" contains a description of the job and this text is displayed in the job dialog. On line in the dialog can have up to 25 characters. Note that the character sequence \n can be used as line separator here.

mark job as done.:

```
JobState "JobName" "Done"
```

Examples:

```
JobState "Fireworks" "Light the fireworks in\nthe garden."
```

```
JobState "Fireworks" "Light the fireworks in"
"the garden."
```

```
JobState "Fireworks" "Done"
```

## Text modifiers

Text modifiers can upgrade the text inside message boxes, dialogs, books, rolls or game/level intros (**2D text**). Also text placed with the **misc\_MapText** entity inside the level (**3D text**) can be modified. Text for the console can be color modified only.

- Character modifiers

| Modifier | Modification                                          | 3D text |
|----------|-------------------------------------------------------|---------|
| ^1       | ■■■ Color red                                         | yes     |
| ^2       | ■■■ Color green                                       | yes     |
| ^3       | ■■■ Color yellow                                      | yes     |
| ^4       | ■■■ Color blue                                        | yes     |
| ^5       | ■■■ Color orange                                      | yes     |
| ^6       | ■■■ Color magenta                                     | yes     |
| ^7       | Color white                                           | yes     |
| ^8       | ■■■ Color black                                       | yes     |
| ^9       | ■■■ Color dark red                                    | yes     |
| ^0       | ■■■ Color gray                                        | yes     |
| ^r       | Reset all modifications                               | yes     |
| ^b       | Bold on/off (is implemented as color inversion)       | yes     |
| ^s       | Shadow on/off                                         | yes     |
| ^x       | Size of characters increased by one character height  | yes     |
| ^y       | Size of characters increased by half character height | yes     |
| ^u       | Underline on/off                                      | no      |
| ^f       | Flashing on/off                                       | yes     |
| ^i       | Italic on/off                                         | no      |

Example: this is a ^1red^r ^uunderlined^r text.

The end of a text line also resets all modifications.

- Tabulators

With tabulators you can nice up dialogs, create simple lists or align header and footer information. Tabulators are usable for message boxes, dialogs, books, rolls or game/level intros (2D text).

Tabulator stops are specified by a width in characters. Thereby a character width equals 8 units in Ya3days unified text coordinate system (base is a 640 \* 480 screen size).

Preset are 10 tabulator stops with a width of 8 each. This preset is restored

at begin of each message box, dialog, book, roll or each `,text`` statement of a game/level intro.

You can change these widths or align text left, right or centered on a tab stop.

Tabulator stops are specified by writing

```
^T width1 width2 width3 ...
```

You can specify up to 10 widths. Add a `r` character to a width to get a right align tabulator or a `c` character for a center align tabulator. A tabulator specification ends with a `^` character, an end of line character `\n` or the end of the text line. If no width is specified, the tabulators are reset to the default.

To advance to the next tabulator stop write `^t`.

Examples:

\* From file `Rolle_ExhibitionPhysics.txt`

```
^T 27c^r^5^u^xPhysics lab.
```

Center all text on the roll.

\* From file `BookFirstHelp.txt`

```
^T 14r
Level:^t ^2$Level
My name:^t ^2$Name
Difficulty:^t ^2$Skill
```

Right align the first column, the text after the tabulator stays left aligned .

\* From `BookUsage1.txt`

```
^T 14^1W^r/^1Arrow up^r^tWalk forward
^1S^r/^1Arrow down^r^tWalk backward
^1A^r^tStep left
```

A list. The first column is left aligned and names keys, the second column is the explanation.

# Script Commands

## Overview script commands

**Execute command during script loading.**

**This commands produce no code and can be placed outside any section (before the first section).**

const  
string  
int  
float

## **Run time script commands**

actor\_target  
teleport\_actor\_target  
DispIntuition  
DispGameState  
message  
dialogheader  
dialoganser  
dialogend  
dialogcancel  
stop  
go  
jump  
duck  
goodguy  
EnemyTest  
trigger  
FollowPlayer  
FollowLover  
FollowMe  
freeze  
sleeping  
invisible  
infected  
weaponsave  
weaponoff  
weaponon  
powerarmor  
itemgive  
itemtest  
itemdrop  
itemtake  
itemExchange  
itemUseOrSearch  
sound  
loopsound  
radio  
spawnflags\_set  
wait  
lookat  
pose  
print  
centerprint  
killme  
scriptoff  
timer  
clock  
command  
debug  
waypoint  
JobState

```

Effect
CreateActor
CreateEntity
skill
HitlistEnter
HitlistMessage
ListFill
ListGet
dmgteam
PlayerSelect
Player
PhysicObjectsMoved
sleep
speaksetup
speak
InventoryGive
InventoryRemove
InventoryTest
VoxBlockTrigger
VoxBlockSet
VoxBlockTest
VoxBlockParam

```

## Assignment to variables

Assignments to variables are done with the `=` character. The left side is the name of a variable, the right side is an expression. If the variable does not exist by the time the instruction is executed, it is created as a variable of the type `string`.

Examples:

```

a = 0 // the job is not done
r = random() // random number
p = Game.Time + 1 // reset pause
n = (Game.HourNr >= 19) || (Game.HourNr <= 5) // night time
i = i + 1 // one more
t = "Hello World" // assign text
PlayerSkills.Intuition = (random() * 20) + 5 // assign to grouped variable

```

## If commands

```

if <expression> Begin if command.
 ... Is executed if <expression> evaluates to true.
elseif <expression> Optional, use as often as you need.
 ... Is executed if all previous if/elseif
 ... have evaluated to false and this one to true.
else Optional, can only occur once.
 ... Is executed if all previous if/elseif
 ... have evaluated to false.
endif End if command.

```

Example:

```

R = random()
if r < 0.33
 dialogheader "Attention!"
elseif r < 0.66
 dialogheader "Stay away!"
else
 dialogheader "Hi."
endif

```

## Loop commands

Commands for loops are always used in pairs.

### loop

```
loop
 ...
endloop
```

Endless loop.

### do loop

```
do
 ...
until <expression>
```

If expression evaluates to true, the loop is terminated.  
The commands in the loop are executed at least once.

### while loop

```
while <expression>
 ...
endwhile
```

The loop will not enter or continue if <expression> is/gets false.  
If <expression> is already false the first time, no commands within  
the loop will be executed.

### break

break can only be used within loops.  
The (inner) loop will break.

### continue

continue can only be used within loops.  
The execution of commands continues on begin of the loop.  
While loops also test <expression> again.

Example:

```
v = 0
do
 v = v + 1
 if (v == 3)
 continue
 endif
 if (v > 5)
 break
 endif
 print "do " v
until v > 7
```



## **Sleep command**

```
sleep <expression>
```

<expression> is the time in seconds, where the execution of the script is paused. When the time expires, execution continues after the sleep command.

During the sleep, other events are still processed. If there is a new sleep command executed in event processing, script execution continues after this sleep command.

If <expression>  $\leq$  to 0 so, execution continues after the sleep command without pausing. A previously active sleep command canceled now.

## Detailed description of script commands

```
/******
script command: <type> [const] <variable> [= <value>] { , <variable> [= <value>] }
```

Define a global or local variable.

<type> Variable type, can be 'int', 'float' or 'string'.  
\* string holds a string with up to 511 Characters  
\* int holds a 32 bit signed integer value  
\* float holds a 64 bit floating point value

const The optional const specifier can be placed before or after the variable type.  
If used, the variable cannot be written by an assign statement.

<variable> The name of a variable can be composed of letters, digits, and the underscore character (an identifier).  
A name can have a maximum length of 71 characters.  
One identifier declares an local script variable.  
Two identifiers concatenated with a point declares a global variable (or a so called 'grouped variable').

<value> is any text, can be an expression

NOTE: \* The value assigned is only done on first creation of a variable  
\* Multiple variable definitions are separated by commas.

on entry, \*pText points to begin of the arguments

```
/******
script command: actor_target [run] [AutoWaypoint] <Goal>
```

run

If run is present, the actor will run (not walk)

AutoWaypoint

If AutoWaypoint is present, the actor will use waypoints to reach the goal, if it is more than 512 units away.

<Goal> can be

PlayerX

Actor walks to the named player.  
X must be in the range 1 .. game.maxclients

EdictX

Actor walks to the named edict.  
X must be in the range 1 .. globals.num\_edicts - 1

Infected

Actor walks to the infected player or bot (if any)

'targetname' of actor\_target

Actor walks to the named actor\_target

If this actor\_target is not found, the Actor will stand.  
If there are multiple instances, one of them is picked.  
If the actor is standing on one of the name actor\_target's, this one is skipped as possible goal.

'targetname' of other entity

Actor walks to the named entity

NOTE: The last actor\_target is saved intern.

It is used by the 'go' command.

```
/******
script command: teleport_actor_target <'targetname' of actor_target>

Actors origin is changed to the origin of the named actor_target

If this actor_target is not found, the Actor will stand.
If there are multiple instances, one of them is picked.

NOTE: The last actor_target is saved intern.
It is used by the 'go' command.
```

```
/******
script command: DispIntuition <Text> [TimeToStay]

Displays this message on the Overlay.

<Text> Text to output. If first character is ^, it's a reference
 to an dialog text file.

[TimeToStay] is the text, the message will stay (in seconds) on the
 overlay, if missing, it defaults to 5 seconds.
```

NOTE: Max 7 lines fit in the display.  
The text lines are centered.  
Maximum length 1st Line: 20 characters  
Maximum length 2nd Line: 24 characters  
Maximum length 3rd Line: 26 characters  
Maximum length 4th Line: 26 characters  
Maximum length 5th Line: 26 characters  
Maximum length 6th Line: 24 characters  
Maximum length 7th Line: 20 characters

```
/******
script command: DispGameState <Text> [TimeToStay] [nChars] [nLines] [BackGroundPicture]

Displays the game status on the Overlay.

<Text> Text to output.

[nChars] is the number of characters which fits in the
 overlay, if missing, it defaults to 30 characters.

[nLines] is the number of lines which fits in the
 overlay, if missing, it defaults to 30 lines.

[TimeToStay] is the text, the message will stay (in seconds) on the
 overlay, if missing, it defaults to 5 seconds.

[BackGroundPicture] name of picture used as background.
 overlay, if missing, it defaults to 'Dlg/Dback'.
```

NOTE: Game status is displayed for all players in the game.

```
/******
script command: message <Messagetext> [TimeToStay] [Range] [MessageEndEvent]

Displays this message on the Overlay.

<Messagetext> Text to output. If first character is ^, it's a reference
 to an dialog text file.

[TimeToStay] is the time, the message will stay (in seconds) on the
 overlay, if missing, it defaults to 5 seconds.

[Range] If the distance to the player is more than Range,
 the message is not outputted. Range defaults to near.
 use
 melee (nearer than 80)
 near (nearer than 500 and visible)
 mid (nearer than 1000 and visible)
 far (any distance and visible)
```

always (message is outputted independent of distance and visibility)  
all (like always, in multiplayer games is outputted to all  
players. 'MessageEndEvent' is not used here)

[MessageEndEvent] optional. If message is removed from screen, this section  
is executed in the command script.

NOTE: If the message is outputted 'Game.LastResult' has the value of 1  
If the player is to far or was not visible 'Game.LastResult' has the value of 0  
If the message is not outputted, because any other message or dialog  
is on the screen in the moment, 'Game.LastResult' has the value of -1

```
/******
script command: dialogheader <Messagetext>
```

<Messagetext> Text to output. If first character is ^, it's a reference  
to a dialog text file.

Begin of an Dialog. The actor says the <Messagetext>.

```
/******
script command: dialoganser <Sectionname> <Messagetext>
```

One of the possible answers of the player.  
If this answer is selected, the section <Sectionname> is executed.

<Messagetext> Text to output. If first character is ^, it's a reference  
to an dialog text file.

```
/******
script command: dialogend [Cursor] [NoBackground] [TimeToStay] [Range]
```

End of an dialog definition.

[Cursor] This is optional. Write the text 'Cursor' to force  
a mouse cursor if this dialog is open.  
--> Use this for dialogs whits need the mouse to  
work reasonable.

[NoBackground] This is optional. Write the text 'NoBackground' to  
have no background image for the dialog.  
--> Use to give the dialogs a specific layout.  
NOTE: Drawing the dialog the setting 'Dialog\_Img\_Frame'  
and 'Dialog\_Img\_Back' from the file  
'GameConfiguration.txt' is not used.

[TimeToStay] is the time, the dialog will stay (in seconds) on the  
overlay, if missing, it defaults to 20 seconds.

[Range] If the distance to the player is more than Range,  
the dialog is not outputted. Range defaults to near.  
use  
melee (nearer than 80)  
near (nearer than 500 and visible)  
mid (nearer than 1000 and visible)  
far (any distance and visible)  
always (message is outputted independent of distance and visibility)

NOTE: If the dialog is outputted 'Game.LastResult' has the value of 1  
If the player is to far or was not visible 'Game.LastResult' has the value of 0  
If the dialog is not outputted, because any other message or dialog  
is on the screen in the moment, 'Game.LastResult' has the value of -1

```
/******
script command: dialogcancel
```

Cancels any open Dialog and message

NOTE: The section [DialogCancel] is not executed!

```
/******
script command: stop [alternate stand pose]
```

The Actor will stand.

\* alternate stand pose  
This is an alternate pose to the default stand pose.  
NOTE: The model animations must support this pose.  
Known values are:  
\* none not alternate stand pose (is the default pose)  
\* stand standing, q3 models can turn their heads  
\* sit sitting, q3 models can turn their heads

```
/******
script command: go
```

If there was an saved target\_actor goal, the Actor will continue to walk to this goal.

```
/******
script command: jump [speed] [height]
```

Actor jumps in direction of ideal yaw (It's current viewing direction)

[speed] optional jump speed, defaults to 200  
[height] optional jump height, defaults to 200

```
/******
script command: duck on|off
```

Actor duck on/off

```
/******
script command: goodguy on|off
```

Actor goodguy on/off

```
/******
script command: EnemyTest
 EnemyTest EvadeModel { xxx}
 EnemyTest HuntModel { xxx}
```

Test for enemy or evade from monsters/actors.

\* No arguments  
If the actor has no enemy, test for one in the near.  
\* EvadeModel { xxx}  
Evade from monsters/actors having a specific model.  
'xxx' is a file path to a model string.  
There can be multiple model strings.  
Example: EnemyTest EvadeModel "players/cat/" "players/wolf/"  
Evade from cat and wolf models.  
\* HuntModel { xxx}  
NOTE: If the actor already has an enemy, this action is skipped.  
If the actorHunt monsters/actors having a specific model.  
'xxx' is a file path to a model string.  
There can be multiple model strings.  
Example: EnemyTest HuntModel "players/cat/" "players/wolf/"  
Hunt cat and wolf models.

Can also be used from bad guys walking around. The waypoint move code disables looking for enemies if the actor is on the way.

NOTE: 'Game.LastResult':  
-1: Actor is freezed or paralyzed, actor has no enemy.  
0: Actor has no enemy  
1: Actor has an enemy  
2: Set evade from an enemy or hunt of an enemy

```
/******
script command: trigger <targetname> [SectionName] [EventArg1] [EventArg1]
```

triggers all enties with <targetname>.

[SectionName] if the triggered entity is a misc\_actor, it's section

SectionName is executed, if SectionName is not give,  
the section [ActorUsed] is ececuted.

[EventArg1] Optional argument if event is send to actor with script.

[EventArg2] Optional argument if event is send to actor with script.

```
/******
script command: FollowPlayer on|off [RangeStand [RangeRun]]
```

If on, the Actor will follow the player and will help him  
to fight his enemies.

If off, this feature is turned off.

RangeStand: If actor is is closer than this, it stops.  
Optional, default is 128.

RangeRun: If actor is further away he start to run to the player.  
Optional, default is 256.

```
/******
script command:
```

FollowLover Check [RangeSearch] [RangeStand] [RangeRun]

Check for lover.

Try to find a lover. If there is on in the near  
move to this entity.

RangeSearch: Distance threshold, check for lovers near me.  
Optional, default is 256.

RangeStand: If actor is is closer than this, it stops.  
Optional depends from the entity size.

RangeRun: If actor is further away he start to run to the player.  
Optional, default is 256.

'Game.LastResult':

<= 0 Have no lover  
0 I am in love mode but there is no other  
entity in the near which is in love too.  
-1 I am not in love  
-2 Follow anyone, but not this is no lover  
1 Very near to my lover.  
Entity is standing and looking to lover.  
> 1 Have a lover, value is distance to lover.

FollowLover SpawnBaby

Spawn a baby mob.

ON spawn of a baby the section [BabySpawned] of both parents is executed.  
Use this event to reset the love mode of the parents.

'Game.LastResult':

<= 0 Have no lover  
0 Was not able to create a baby  
1 A baby mob was created  
-1 I am not in love  
-2 Follow anyone, but not this is no lover  
-3 Have no lover  
-4 Lover is to far away

```
/******
script command:
```

followme regroup <'targetname' of misc\_actor> [<order>] [<DistArg1>] [<DistArg2>]

From now on, the named Actors will follow this actor.

followme stop

follow me will stop, the other actors are freed

followme pose <pose string>

pose string for the followMe's, see pose command  
NOTE: The others must have an actorscript to do the poses

followme look atme  
follow me will look at the misc\_actor executing this command

followme look fromme  
follow me will look away from the misc\_actor executing this command

followme look asi  
follow me will look in the same direction as the misc\_actor executing  
this command

The follower look in the give direction.  
The ideal\_yaw is set.

|         |            |                                      |
|---------|------------|--------------------------------------|
| <order> | InLine     | In Line behind the leader (default)  |
|         | DoubleLine | 2 Lines behind the leader            |
|         | Parallel   | Parallel behind the leader           |
|         | Circle     | in a Circle behind/around the leader |
|         | Keil       | Keil behind the leader               |

/\*\*\*\*\*  
script command: variable <variable> [ = <value>]

define ActorVariables

<variable> is one of the ActorVariables  
<value> is any text, can be an expression

NOTE: \* The value assigned is only done on first creation of a variable  
\* with value assigns, no other variable definition may follow.  
\* without value defininitin, there may be more variables in a line.  
In this case, the variable is presetted with an empty string.

/\*\*\*\*\*  
script command: freeze on|off

freeze on or off.

NOTE: A frozen actor stands still like a stone statue.

/\*\*\*\*\*  
script command: sleeping on|off|NoSnore

Sleeping on or off:  
on: Sleeping on with snoring  
NoSnore: Sleeping on without snoring  
off: Sleeping off

NOTE: a sleeping actor makes snore sounds

/\*\*\*\*\*  
script command: invisible on|off

invisible on or off.

NOTE: A invisible actor is not seen.

/\*\*\*\*\*  
script command: infected on|off|clearall|count

Infected on, off, clearall or count.

NOTE: on, off: Actor shows infected effect on/off  
clearall: all infected players/bots infection off  
count: 'Game.LastResult' holts the number of infected

/\*\*\*\*\*  
script command: weaponsave

save the weapon of the actor

```

/*****
script command: weaponoff

no weapon for this actor, the model removes it's weapon
/*****
script command: weaponon

restore weapon of this actor (from weaponsave)
/*****
script command: powerarmor <type> <amount>

powerarmor for the actor

<type> is SHIELD or SCREEN, others switch off any powerarmor
<amount> how long armor holds
/*****
script command: itemgive <name of item> <amount>

give item to player

<name of item> is the classname of an item (ammo_rockets, item_quad, ...)
<amount> if <amount> is given in the argument list, the number
 of items is given to the player.
/*****
script command: itemtest <name of item>

test item of player

<name of item> is the classname of an item (ammo_rockets, item_quad, ...)
Money get players money (is no item)
Mana get players money (is no item)

NOTE: the amount can be picked up with 'Game.LastResult'
 'Game.LastResult' has the value of -1 if item is not existing
/*****
script command: itemdrop <item> [<amount/change>] { <item> [<amount/change>] }

Actor drops items

<item> Is the classname of an item (ammo_rockets, item_quad, ...)
 Use the string 'MyWeapon' to drop the weapons of the actor
 (if it has any).
<amount/change> Is a value (or expression).
 If >= 1.0 this number of items are dropped
 If > 0.0 and < 1.0 this is the change to drop one
 item (0.0 drops no item, 1.0 for sure drops an item).

The argument list can have multiple item <item>/<amount/change> pairs.
The last <amount/change> is optional (it defaults to 1.0).

Dropped items are removed from the game after 29 seconds.

Examples: itemdrop item_quad
 itemdrop item_quad 3
 itemdrop ammo_rockets 1 ammo_rockets 0.6 item_quad 0.3

NOTE: 'Game.LastResult' has the value of -1 if item is not existing
 'Game.LastResult' or a value of >= 1 for the number of items dropped

```



```

/*****
script command: itemtake <name of item> <amount>

If Player have <name of item>, reduce it by <amount>

<name of item> is the classname of an item (ammo_rockets, item_quad, ...)
Money take players money (is no item)
Mana take players money (is no item)

NOTE: 'Game.LastResult' has the value of -1 if item is not existing
 else 'Game.LastResult' holds the item count after reduction.
 The new amount of the item is clipped to 0.
*****/
script command: itemExchange

Test Player to have the startup items named in the second till last
startup items. If the player has all this in it's inventory, remove
them all and give the player the first startup item.

NOTE: 'Game.LastResult' has the value of 1 if the exchange was done
 else 'Game.LastResult' is 0.
*****/
script command: itemUseOrSearch <range> <items>

Search for items being in the range and being visible from the
actor.

If the actor touches the item, the item is picked up and
used.

If the actor don't touches the item, it returns it's edict number
(which could be use for a actor_target command).

<range> items must be inside this distance from the player.

<items> * A list of specific items like
 weapon_railgun ammo_slugs item_health
 * The text 'StartupItems'
 In this case all entries from the startup items are
 searched too.
 * ItemsWeapon, ItemsAmmo, ItemsArmor, ItemsKey, ItemsPowerup,
 ItemsSomething, ItemsHealth, ItemsAll
 Any of this search for items of this type.

Health items are only searched, if the actor has not it's max health.

return in 'Game.LastResult'

 0 Nothing to do
 > 0 EdictNr, Actor must move to this item.
*****/
script command: sound <name of sound> [<targetname>] [<attenuation>]

The actor plays the named sound.

Examples: sound "player/gaspl.wav"
 sound "items/pkup.wav" 3.0
 sound "gladiator/gldidle1.wav" 1.0

<targetname> optional entity which plays the sound
 if not given, the caller plays the sound.
 Use 'player' if the nearest player should play the sound.
 Use the character - if you have an attenuation but don't
 want to use the targetname feature.

```

```
<attenuation> range 0.0 to 4.0, default is ATTN_IDLE
0.0 ATTN_NONE full volume the entire level
1.0 ATTN_NORM full volume the entire level
2.0 ATTN_IDLE full volume the entire level
3.0 ATTN_STATIC diminish very rapidly with distance
```

```
/*****
script command: loopsound <name of sound> [<attenuation>]
```

The actor plays the named sound in a loop.

Examples: loopsound "ambient/Kneipel.wav"

Is <name of sound> Off, than any looped sound is switched off

```
<attenuation> range 0.0 to 4.0, default is ATTN_IDLE
0.0 ATTN_NONE full volume the entire level
1.0 ATTN_NORM full volume the entire level
2.0 ATTN_IDLE full volume the entire level
3.0 ATTN_STATIC diminish very rapidly with distance
```

```
/*****
script command: radio <name of sound>
```

The given sound is heres in the complete level by all clients

Examples: radio "player/gaspl.wav"  
radio "items/pkup.wav"  
radio "gladiator/gldidle1.wav"

```
/*****
script command: spawnflags_set <targetname> <expression>
```

Set Bits in spawnflags of entity with <targetname>.

```
/*****
script command: wait <seconds to wait>
```

The actor waits the time (in seconds).  
The actor goes to the stand pose. The actors pausetime is set to  
the given value.

NOTE: wait must be after 'go' or 'target\_actor', because these commands  
reset any wait time.

```
/*****
script command: lookat <target>
```

The actor sets it's direction towards the target.

```
player look in direction of player
target any existing target
0 .. 360 at this direction
```

NOTE: the direction of the actor is reset after an 'go' or  
'target\_actor'.  
Best usage is after an 'stop' command.

```
/*****
script command: pose <pose string>
```

If the actor is standing, it will makes the poses given as Argument.  
The characters in the argument are the poses the actor will make.

```
F flipoff
S salute
T taunt
W wave
P point
J jump
' ' stand (the character blank!)
| this character sets an repeat, if the pose string ends,
 the poses continue after this character.
```

NOTE:

- \* Every new go, stop or target\_actor will reset the poses.  
It's for use after a stop
- \* Enclose the argument in quotes, if the stand pose is used (the blank).

```

/*****
script command: print arguments

prints out the arguments on the console
/*****
script command: centerprint arguments

prints out the arguments to the center of the screen.
/*****
script command: killme

remove this actor from the game
/*****
script command: scriptoff

removes the script from the calling actor.
/*****
script command: timer <seconds until timer fired> [SectionName] [Argument1] [Argument2]

Fire execution of section SectionName, This is a one-shot timer.
If SectionName is not given, the SectionName 'Timer' is used

NOTE: if <seconds until timer fired> is < 0, the timer is switched off
/*****
script command: clock <seconds clock delta>

Fire execution of section "ClockTick" in deltas of <seconds clock delta>.

NOTE: if <seconds clock delta> <= 0, the clock is switched off
/*****
script command: command "command to system"

executes one of the 'console commands'.

Example: command "menu_loadgame"
/*****
script command: debug on|off|DumpLocVars

on Switches debug prints on
off Switches debug prints off
DumpLocVars Dump the actor script local variables

'Game.LastResult': < 0: unknown argument
 0: debug prints are off
 1: debug prints are on
/*****
script command: waypoint [run] targetname

waypoint managment

waypoint Off
remove all waypoints

waypoint [run] targetname1 targetname2 targetname3 ...
Move to one of the waypoints (up to 32) (1 is randomly chosen).
If run is present, the actor will run (not walk).

If targetname is the string actor moves to Waypoint
" MyHome" shortest distance to start position of actor

```

```

"StartupPositon256" near start position of actor with max distance 256
"StartupPositon512" near start position of actor with max distance 512
"StartupPositon1024" near start position of actor with max distance 1024
"StartupPositon2048" near start position of actor with max distance 2048
"CurrentPositon256" near current position of actor with max distance 256
"CurrentPositon512" near current position of actor with max distance 512
"CurrentPositon1024" near current position of actor with max distance 1024
"CurrentPositon2048" near current position of actor with max distance 2048
"RandomPositionXXX" A Position somewhere in around the current position.
 Max. distance will be XXX.
"EvadePlayerXXX" A Position somewhere around the current position.
 Do not get closer than XXX to the next player.
 This works without waypoints. XXX must be greater zero.

```

/\*\*\*\*\*

```

script command: JobState <name of job> [<job text>]

test / change job state

<name of job> is the name of the job

<job text> is displayed in the job screen
 if <job text> is "Done", the job will not
 be displayed and marked as done.

```

```

NOTE: 'Game.LastResult' has the value of
 0 job is not existing
 1 job is existing and not done
 2 job is done

```

/\*\*\*\*\*

```

script command: Effect <name of effect>

Make effect at actors location.

<name of effect> is the name of the effect:
StarsRed Red stars
StarsGreen Green stars
StarsBlue Blue start
StarsYellow Yellow stars
StarsWhite White stars
Hearts Emits some Hearts
SmokeGray Emits some gray smoke particles
SmokeBlack Emits some black smoke particles
Login Login effect
Logout Logout effect
Explosion1 <Damage> [<Radius>] The actor explodes (type 1 explosion)
 The actor is not hurt
 <Damage> 0 .. 999, damage to the neighborhood
 <Radius> 32 .. 512, optional explosion radius. Default is <Damage> + 40.
Explosion2 <Damage> [<Radius>] The actor explodes (type 2 explosion)
 The actor is not hurt
 <Damage> 0 .. 999, damage to the neighborhood
 <Radius> 32 .. 512, optional explosion radius. Default is <Damage> + 40.
ShowOff show symbol above actor off
ShowExclamation show exclamation mark above actor
ShowQuestion show question mark above actor
ShowCoins show coins above actor
ShowHeart show hear above actor
Light <Range> <Red> <Green> <Blue> Light Around Actor
 <Range> 0 .. 3, use 0 to switch off the light
 <Red> 0 .. 3
 <Green> 0 .. 3
 <Blue> 0 .. 3
ShellOn Actor has a shell, note that the light
 settings are used for the shell color
ShellOff Actor shell off
Rf2EffectOff Render function 2 effect off
Rf2EffectFlames Render function 2 effect, actor burns

RfsFlagBits <value> Render shader flag bits (4 bit)
 Set the shader special effect 'flag bits' of
 this entity. An image shader can test this

```

RfsColorIdx <value> value.  
 Render shader color index (4 bit)  
 Set the shader special effect 'color index' of  
 this entity. An image shader can test this  
 value.

SizeFactor <value> Set size factor of entity.  
 Range is 0.1 .. 10.0.  
 1.0 is default size.  
 0.5 is half size.  
 2.0 is double size.

/\*\*\*\*\*

script command: CreateActor [Bot] <where> <Model> <Name> <spawnflags> <weapon>  
 <health> <ActorScript> <targetname> <target>

create actor

[Bot] Bot is optional and is used for creating bots.  
 [MobAmbient] MobAmbient is optional and is used to flag an ambient mob.  
 [MobPassive] MobPassive is optional and is used to flag a passive mob.

<where> "MyLocation" for loaction of the script owner  
 DeathmatchSpawnPoint  
 name of waypoint

<spawnflags> can be ored together  
 1 "Ambush (Monster)"  
 2 "Trigger Spawn (Monster)"  
 4 "Sight (Monster)"  
 8 "Good Guy"  
 16 "No Gib"  
 32 "Use Homing Rockets"  
 64 "Be Monster"  
 128 "Ignore Fire"  
 4096 "No Visual Weapon"  
 8192 "Follow Player"

<weapon> can be one of this  
 0 no Weapon  
 1 close-range attack (no Weapon)  
 2 close-range attack (with STD Weapon)  
 3 Blaster  
 4 Shotgun  
 5 Supershotgun  
 6 Machinegun  
 7 Chaingun  
 8 GrenadeLauncher  
 9 Rockets  
 10 Hyperblaster  
 11 Railgun  
 12 BFG  
 13 Throws flames  
 14 Throws green poison  
 15 Lightning blue  
 16 Fireball  
 17 Lightning red  
 18 Snowball  
 19 Crossbow  
 20 Crossbow with fire arrows  
 30 Lightsaber blue  
 31 Lightsaber green  
 32 Lightsaber red  
 33 Combat knife  
 34 Assassin dagger  
 35 Rusty sword  
 36 Lohengrins sword  
 37 Katana  
 38 Ancestral sword  
 39 Simple sword (Lego style)

/\*\*\*\*\*

script command: CreateEntity <where> <Classname> <spawnflags> <health>

Spawn an entity by its classname.

<where> "MyLocation" for location of the script owner  
name of waypoint

<Classname> Classname of the entity we want to spawn.

<spawnflags> Depends from the spawned entity.  
A numeric value.

<health> Health value for the spawned entity.  
A numeric value.  
Typically 0 will set a default health value.

```
/******
script command: skill <name of skill> [<amount to add>]
```

test / change job state

<name of skill> is the name of the skill

<amount to add> value to add to skill

Example: skill "Magic" 15.0 ; add skill

NOTE: 'Game.LastResult' has the value of skill after add (0 .. 100)  
or -1 if skill not known

```
/******
script command: HitlistEnter <name of Hitlist> ascend|descend <name of player> <value>
```

test / change job state

<name of Hitlist> is the name of the hitlist

ascend|descend sorting of hitlist  
ascend: sorted by maximum value (like most points)  
descend: sorted by minimum value (like best time)  
NOTE: must match HitlistMessage for same hitlist

<name of player> is the name of the player

<value> is the value to enter in the histlist for this player

NOTE: 'Game.LastResult' has the value of  
0 done  
1 entry is on top of the list

```
/******
script command: HitlistMessage <name of Hitlist> ascend|descend <format>
<Messagegetext> [TimeToStay] [Range] [MessageEndEvent]
```

Displays this message on the Overlay.

<name of Hitlist> is the name of the hitlist

ascend|descend give one of this for sorting direction of hitlist  
NOTE: must match HitlistEnter for same hitlist

<Messagegetext> This text is displayed as header.

<format> Formatting for numbers  
time mm:ss minutes and seconds  
- no formatting

[TimeToStay] is the time, the message will stay (in seconds) on the  
overlay, if missing, it defaults to 5 seconds.

[Range] If the distance to the player is more than Range,  
the message is not outputted. Range defaults to near.

```

use
 melee (nearer than 80)
 near (nearer than 500 and visible)
 mid (nearer than 1000 and visible)
 far (any distance and visible)
 always (message is outputted independent of distance and visibility)

```

[MessageEndEvent] optional. If message is removed from screen, this section is executed in the command script.

NOTE: If the message is outputted 'Game.LastResult' has the value of 1  
 If the player is to far or was not visible 'Game.LastResult' has the value of 0  
 If the message is not outputted, because any other message or dialog is on the screen in the moment, 'Game.LastResult' has the value of -1

```

/*****

```

```

script command: ListFill <what to fill>

```

```

fill list with information

```

```

<what to fill>:

```

```

Reset Resets the list
Add Add a line to the list
AddItem NameOfItem Add a line to the list with info about an item
 GUI name|GUI description|Price|Classname|
 Icon name|Quantity
AddRecipe1 RecipeType NameOfItem Add a line to the list with info about
a simple recipe. The recipe must match the recipe type,
must have one input of the named item and one output.
RecipeType: Type of recipe (crafting, cooking, ...)
NameOfItem: Class name of an item
 Class name of output item
AddMyInventory For each item slot in the inventory of this actor
 add a line to the list with info about the item
 GUI name|GUI description|Price|Classname|
 Icon name|Count
AddPlayerInventory For each item slot of the players inventory
 add a line to the list with info about the item
 GUI name|GUI description|Price|Classname|
 Icon name|Count
TravelOverland Info of reachable levels (Single Player Levels)
 Name of Level|Price for ticket|
 Can reach level|Leveltype|
 Short level description|Author|spare|
 level description
PlayersAndBots Info of players and bots in the game
 Name|IsPlayer|EdictNr|Health|Infected

```

```

NOTE: * 'Game.LastResult' has number of entries in the list
 < 0 has some error
 * before adding entries, the list has to be reseted
 * There is only one list in the game wich can be used.
 So ensure it't build up from new if used in a dialog.
 * NameOfItem is the classname of an item (like 'item_bottle1').

```

```

/*****

```

```

script command: ListGet <variable> <index> <column>

```

```

get listentry from last ListGet

```

```

<variable> Name of local variable, result is placed here.
<index> number of list entry, 0 ..
<column> the .. column, 0 ..

```

```

NOTE: 0 OK
 < 0 has some error

```

```

/*****

```

```

script command: dmgteam teamname

```

```

Sets up a dmgteam.
Actors with the same dmgteam will help each other in case of trouble.
NOTE: use only at startup of actor.

```

```

/*****
script command: PlayerSelect Selection

Selects a player for player related variables/assigns/actions.

Selection: off Auto selection, selects the nearest player (the default).
 PlayerX X is the Player Nr. (1 .. Game.PlayerMax) to select.

NOTE: * only reasonable for multiplayer games.
 * PlayerX is given as argument to PlayerTouch events.
*****/
script command: Player xxxxx

Player related commands

Infected on Infection for this player on,
 'Game.LastResult' has # infected players
Infected off Infection for this player off
 'Game.LastResult' has # infected players
Infected clearall Infection for all players off,
 'Game.LastResult' has # infected players
Infected count 'Game.LastResult' has # infected players
Invisible on Make this player invisible, 'Game.LastResult' is true if player is
 invisible
Invisible off Make this player visible, 'Game.LastResult' is true if player is
 invisible
Invisible test Test this player for being invisible,
 'Game.LastResult' is true if player
 is invisible
Sleep hours Sleep 'hours'.
 Player makes a snore sound and the game time is incremented
 by 'hours'.
 'hours' has a range from 0.0 to 24.0.
FadeScreen R G B alpha fadein fadeout holdtime Fades the screen to a color.
 R G B color components of fade color, 0-1
 alpha opacity of fade. 0=no effect, 1=solid color
 fadein time in seconds from trigger until full alpha
 fadeout time in seconds after fadein+holdtime from full alpha
 to clear screen
 holdtime time to hold the effect at full alpha value.
 -1 = permanent

InvCraft Update RecipeType GridSize Update the inventory crafting output
 RecipeType Type of recipe (crafting, cooking, ...)
 GridSize Side length of the crafting grid.
InvCraft Get RecipeType GridSize Amount Get crafting output
 RecipeType Type of recipe (crafting, cooking, ...)
 GridSize Side length of the crafting grid.
 Possible values are 1, 2 or 3.
 Amount Get this number of items
InvSelItemDamage Points Subtract damage points from players selected
 weapon/tool.
InvSelItemRemove Amount Remove 'amount' items from players selected item.
MData Update RecipeType Update meta data recipe output
 RecipeType Type of recipe (crafting, cooking, ...)
MData Get RecipeType Amount Get meta data recipe output
 RecipeType Type of recipe (crafting, cooking, ...)
 Amount Get this number of items

NOTE: * multi player game
 Works with selected player or nearest (if none is selected).
 Also see 'PlayerSelect' script command.
 * single player game
 Works with the one and only player.
*****/
script command: PhysicObjectsMoved <targetname> [DistMoved DistPitch DistYaw DistRoll]

<targetname> must be the targetname of a physic_trigger_reset entity.
Count all physic objects which have moved away from there start
position.

```





```

/*****
script command: speak Text [<volume>] [<attenuation>]

The actor speaks the text

 Examples: speak "out of my way"

<volume> range 0.0 to 1.0, default is 1.0

<attenuation> range 0.0 to 4.0, default is ATTN_IDLE
 0.0 ATTN_NONE full volume the entire level
 1.0 ATTN_NORM
 2.0 ATTN_IDLE
 3.0 ATTN_STATIC diminish very rapidly with distance
*****/
script command: InventoryGive <name of item> <amount>
 InventoryGive StartupItems

give item to actor

<name of item> is the classname of an item (ammo_rockets, item_quad, ...)

<amount> if <amount> is given in the argument list, the number
 of items is given to the actor.
 <amount> defaults to 1.

StartupItems Using the text 'StartupItems' in place of <name of item>,
 all items give at startup to the actor are transfert
 to the inventory.

NOTE: the amount can be picked up with 'Game.LastResult'
 'Game.LastResult' has the value of -1 if item is not existing
*****/
script command: InventoryRemove <name of item> <amount>

remove item to actor

<name of item> is the classname of an item (ammo_rockets, item_quad, ...)

<amount> if <amount> is given in the argument list, the number
 of items is removed from the actor.
 <amount> defaults to 1.

NOTE: the amount can be picked up with 'Game.LastResult'
 'Game.LastResult' has the value of -1 if item is not existing
*****/
script command: InventoryTest <name of item>

test item of actor

<name of item> is the classname of an item (ammo_rockets, item_quad, ...)
 With 'Game.LastResult', the amount of items in the actors
 inventory can be picked up.
 'Game.LastResult' has the value of -1 if item is not existing.

CountUsedSlots 'Game.LastResult' returns the number of item slots which
 have any items.
*****/
script command: VoxBlockTrigger <BlockOrigin> <BlockState>

Trigger a block.
This change a block state to on/off (or close/open). Use this for
doors, gates, chests, ...

<BlockOrigin> A number.
 1: Player triggers a block in the near, use the selected block.
 2: Trigger the selected block.
 3: Recalculate the origin of this actor as Block address.

```

This is useful if a script is associated to a block.  
4: Use selected meta data block.

<BlockState> A number.  
-1: Toggle on/off (or close/open).  
0: Set to off (or close).  
1: Set to on (or open).  
xx: Other positive values may depend form the type of the block.

/\*\*\*\*\*

script command: VoxBlockSet <BlockOrigin> <BlockName> <param2>

Set a block on a specific block address.

<BlockOrigin> A number.  
1: Player triggers a block in the near, use the selected block.  
2: Trigger the selected block.  
3: Recalculate the origin of this actor as Block address.  
This is useful if a script is associated to a block.  
4: Use selected meta data block.

<BlockName> Name of the block to set.  
Example: "Cobblestone"  
If "-" is used as block name, the block name is don't care.  
This is used to modify the 'param2' of the block.

<param2> A number.  
Use this as 'param2' of the block.

/\*\*\*\*\*

script command: VoxBlockTest <BlockOrigin> <x> <y> <z> <WhatToTest>  
[<Argument1>] [<Argument2>]

Some block test things.

<BlockOrigin> A number.  
1: Player triggers a block in the near, use the selected block.  
2: Trigger the selected block.  
3: Recalculate the origin of this actor as Block address.  
This is useful if a script is associated to a block.  
4: Use selected meta data block.

<x> <y> <z> Offset to block <BlockOrigin> in blocks.  
This are three numbers added to the block origin.  
The result is used as 'test position'.

<WhatToTest> Specify what to test.  
This is a string.  
\* IsAir  
Is the 'test position' an air block.  
'Game.LastResult' is true if there is an air block.  
\* GropGrow  
Try to increase the degree of ripeness a crop  
(carrot, potato, wheat ...).  
\* CampfireCook  
'Argument1' is the name of item which is cooked.  
Try to add this to a campfire.  
The item 'Argument2' is dropped after cooking.

<Argument1> Additional argument.  
Usage depends from <WhatToTest>.  
Example: "Cobblestone"

<Argument2> Additional argument.  
Usage depends from <WhatToTest>.  
Example: "Cobblestone"

/\*\*\*\*\*

script command: VoxBlockParam <BlockOrigin> <x> <y> <z> <WhatToDo> <param>

Block parameter modification.

<BlockOrigin> A number.  
1: Player triggers a block in the near, use the selected block.  
2: Trigger the selected block.  
3: Recalculate the origin of this actor as Block address.  
    This is useful if a script is associated to a block.  
4: Use selected meta data block.

<x> <y> <z> Offset to block <BlockOrigin> in blocks.  
This are three numbers added to the block origin.  
The result is used as 'block position'.

<WhatToDo> Specify what to do.  
This is a string.  
\* All  
    Access all param2 bits.  
\* Color  
    Access paramtype2 'color' parameter value.  
\* StateBit  
    Access state bit for blocks with drawtype 'normal'  
    or 'nodebox' and set DrawSubType 32 (Switch between 1. and 2. model).  
    A 'param' value of 0 or 1 sets the new state, an other value  
    toggles the state bit.

<param> Parameter value.  
This is an integer number.  
< 0: Pick value, no modify of parameter.  
>= 0: Change parameter value

Game.LastResult: >= 0 The picked or modified parameter value  
< 0 Error