



Ya3dag

Scripting language
The source of intelligence

Based on V2.01 release of May 7, 2018

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.

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
- **PlayerSkills@...**
- **PlayerJobs@...**
- **Text modifiers**
 - Character modifiers
 - Tabulators
- **Script Commands**
 - Overview script commands
 - Assignment to variables
 - If commands
 - If 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 another 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 the ";" character.

- 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.

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). Clock.
[ClockTick]			
[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**

The syntax parser of this script language is not the cleverest. Therefore commands, operators and operands must be separated by **blanks**.

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.

Examples:

60
0.75

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). `$`-Variables are allowed in a string.

Examples:

```
"Hello World"           ; single line string

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

"Hello\nWorld"          ; Same as above
```

\$-Variables

<code>\$A .. \$Z</code>	Global variable (Has the same value in all scripts)
<code>\$a .. \$z</code>	Locale variable (each script has it's own set of local variables)
<code>\$health</code>	health of actor (default 100)
<code>\$maxhealth</code>	maximum value health
<code>\$myname</code>	name of the Actor
<code>\$playername</code>	name of the player figure
<code>\$playerhealth</code>	health of the player
<code>\$playermaxhealth</code>	maximum value of the player's health
<code>\$playerItemSelected</code>	classname of selected item (the player hold this item in the hand)
<code>\$playermana</code>	managers of the player
<code>\$playermoney</code>	money player
<code>\$playerdistance</code>	Distance to (nearest) player
<code>\$playerMax</code>	Maximum number of players (multiplayer games)
<code>\$playerCurr</code>	number of players in the game (multiplayer games)
<code>\$playerInGame</code>	1 if selected player is in the game, otherwise 0 (multiplayer games).
<code>\$playerInfected</code>	1 if selected player is infected, or 0 (multiplayer games).
<code>\$botCurr</code>	number of Bot's in the game (multiplayer games).
<code>\$random</code>	a number between 0.0 and 1.0
<code>\$time</code>	period since start of the game in seconds
<code>\$DayNr</code>	Virtual day, counted since start of the game (1 ..)
<code>\$HourNr</code>	Virtual hour within the day (0 .. 23)
<code>\$OnTheWay</code>	0: no, > 0: number of waypoints to go
<code>\$GameSkill</code>	Skill level, 0: easy, 1: normal, 2: hard, 3:hard+
<code>\$lastresult</code>	Result of some script commands
<code>\$dialogopen</code>	Dialog open: 0 = no, 1 = yes, this Actor, -1 = yes, other Actor
<code>\$argument</code>	Event argument 1
<code>\$EventArg1</code>	Argument to execution of a section.
<code>\$EventArg2</code>	Event argument 1
<code>\$StartupArg</code>	Argument to execution of a section.
<code>\$targetname</code>	Startup argument
<code>\$target</code>	On declaration of the script file, this is separated by a comma from the filename of the script.
<code>\$HaveFreezed</code>	Contents of the targetname field of this object
<code>\$HaveDucked</code>	Contents of the target field of this object
<code>\$HaveParalysed</code>	Object (Actor) is freezed
	Object (Actor) is ducked
	Object (Actor) is paralysed

\$HaveSleeping	Object (Actor) is sleeping
\$HaveInvisible	Object (Actor) is invisible
\$HaveInfected	Object (Actor) is Infected
\$HaveEnemy	Object (Actor) have an enemy
\$HaveGoodGuy	Object (Actor) is a good guy
\$HaveBadGuy	Object (Actor) is a bad guy
\$HaveFollowPlayer	Object (Actor) follows the player
\$HaveFollowAny	Object (Actor) follows a player or an other actor
\$IsPlayer	Object isplayer else Actor, Bot, ...
\$HaveMultiplayer	We are in a multiplayer game
\$VoxLightSun	For voxel environment only. Sun light value at place of the actor (0 is dark, 255 is very bright).

Actor-variables

Structure: Group@Name

Group can be any text (letters + digits), whereby:

ThisLevel	is a shortcut for this level.
ThisScript	is a shortcut for this script file.
ThisLevelScript	is a shortcut for all scripts with this name in this level.

IMPORTANT: Before usage, all actor-variables must be defined within a script with the **variable** statement.

Predefined actor-variables:

- * For the skills of the player (range 0 .. 100):
 - PlayerSkills@Intuition
 - PlayerSkills@Perseverance
 - PlayerSkills@HitTheTarget
 -

Functions

Functions have the format: **Function-Name**(**Argument**)

ItemWoolByIdx(**Argument**)
Argument is a number in the range 0 .. 15.
 Returns the item name of an colored wool block.

itemIsDye(**Argument**)
 Test **argument** to be a dye.
 Returns the dye code if there is a match else -1 is returned.

PlayerSkills@...

PlayerSkills are the skills of the player.

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 $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 Ya3dags 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 with 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 `BookUsagel.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

actor_target
teleport_actor_target
DispIntuition
DispGameState
message
dialogheader
dialoganser
dialogend
dialogcancel
stop
go
jump
duck
goodguy
EnemyTest
trigger
followplayer
followme
assign
variable
freeze
sleeping
invisible
infected
weaponsave
weaponoff
weaponon
powerarmor
itemgive
itemtest
itemdrop
itemtake
itemExchange
itemUseOrSearch
sound
loopsound
radio
spawnflags_set
wait
lookat
pose
print
killme
scriptoff
timer
clock
command
debug
waypoint
JobState
Effect
CreateActor
skill
HitlistEnter
HitlistMessage

```
ListFill
ListGet
dmgteam
PlayerSelect
Player
PhysicObjectsMoved
sleep
speaksetup
speak
InventoryGive
InventoryRemove
InventoryTest
```

Assignment to variables

Assignments are done with the script command

```
Assign <variable> [=] <value>
```

or with the short form

```
<variable> = <value>
```

Note that the character = is optional for the script command assign.

Examples:

```
assign a 0 ; the job is not done
assign r $random ; random number
assign p $time + 1 ; reset pause
assign n (($HourNr >= 19) || ($HourNr <= 5)) ; night time
assign i $i + 1 ; one more
assign t "Hello World" ; assign text
v = $v + 1 ; one more
p = $time + 10 ; time next restart
```

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:

```
assign 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 "do1 " $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: 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 \$lastresult has the value of 1
If the player is to far or was not visible \$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, \$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 [TimeToStay] [Range]
```

End of an dialog definition.

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 \$lastresult has the value of 1
If the player is to far or was not visible \$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, \$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
```

The Actor will stand.

```
/******  
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

Test for enemy

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: $lastresult is 1 if the actor has an enemy else
      $lastresult is 0;

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

triggers all entities 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 executed.

[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 [off]

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

/*****
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: assign <variable> [=] <value>
short form: <variable> = <value>

assign a value to a local variable

<variable> ActorVariable
A .. Z global variable
a .. z local variable
health health of this actor
playerhealth health of the player
playermana Mana of the player
playermoney Money of the player

= The assign character is optional if used with the
assign prefix.

<value> is any text, can be an expression

/*****

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 definition, 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 freezed actor is standing still like a statue made of stone.

/*****

script command: sleeping on|off

sleeping on or off.

NOTE: a sleeping actor makes snore sounds

/*****

script command: invisible on|off

freeze 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: \$lastresult holds 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 $lastresult
      $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: $lastresult has the value of -1 if item is not existing

```

```

    $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: $lastresult has the value of -1 if item is not existing
      else $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: $lastresult has the value of 1 if the exchange was done
      else $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 $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: player/gaspl.wav
         items/pkup.wav
         gladiator/gldidle1.wav

<targetname> optional entity which plays the sound
             if not given, the caller plays 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: 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: player/gasp1.wav
 items/pkup.wav
 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: 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 0/1

0 switches debug prints off
1 switches debug prints 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.
```

/*****

```
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: $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
StarsGreen
StarsBlue
StarsYellow
StarsWhite
Login
Logout
Explosion1 <Damage>          The actor explodes (type 1 explosion)
                           The actor is not hurt
  <Damage> 0 .. 999, damage to the neighborhood
Explosion2 <Damage>          The actor explodes (type 2 explosion)
                           The actor is not hurt
  <Damage> 0 .. 999, damage to the neighborhood
ShowOff                     show symbol above actor off
ShowExclamation             show exclamation mark above actor
ShowQuestion                show question mark 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.

```
/******  
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: 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: \$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: \$lastresult has the value of
0 done
1 entry is on top of the list

/*****

script command: HitlistMessage <name of Hitlist> ascend|descend <format>
<MessageText> [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

<MessageText> 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 \$lastresult has the value of 1

If the player is to far or was not visible \$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, \$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
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
VoxMobSpawn	Test for mobs to spawn. Name Model spawn position

NOTE: * \$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> a .. z, 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 .. \$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, \$lastresult has # infected players
Infected off infection for this player off, \$lastresult has # infected players
Infected clearall infection for all players off, \$lastresult has # infected players
Infected count \$lastresult has # infected players

NOTE: * works with the selected player
* only reasonable for multiplayer games.

```
/******  
script command: PhysicsObjectsMoved <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.

DistMoved Object moved minimum this position.
 Use 0 to don't test moved. Default is 48.0.

DistPitch Object turned minimum this angle (in degrees).
 Use 0 to don't test this angel. Default is 40.0.

DistYaw Object turned minimum this angle (in degrees).
 Use 0 to don't test this angel. Default is 0.0.

DistRoll Object turned minimum this angle (in degrees).
 Use 0 to don't test this angel. Default is 40.0.

NOTE: The number of moved objects are picked up with \$lastresult

 \$lastresult has the value of -1 if <targetname> was no
 physic_trigger_reset entity or if <targetname> does not exist.

```
/******  
script command: speaksetup language RelRate RelPitch RelRange roughness  
                                  flutter clarity echo_delay echo_amp
```

Setup speak of this actor.
This setup's are used for following speak commands.

<language> Language to speak. See the
 espeak-data/docs/languages.html
 for languages.
 Example: en for english, de for german.

<RelRate> speed of speak
 Sprechgeschwindigkeit
 range -100 to 100, default is 0

<RelPitch> base sound frequency
 Tonhöhe
 range -100 to 100, default is 0

<RelRange> base sound frequency variation
 Variation der Tonhöhe
 range -100 to 100, default is 0

<roughness> roughness
 Rauhigkeit der Stimme

range -1, 0 to 7, default is -1

<flutter> flutter
Flutter der Stimme
range -1, 0 to 20, default is -1

<clarity> clarity
Deutlichkeit der Stimme
range -1, 0 to 5, default is -1

<echo_delay> echo delay im ms (1/1000 seconds)
Echo der Stimme in ms (1/1000 Sekunden)
range -1, 0 to 250, default is -1

<echo_amp> Echo Amplitude
Echo Amplitude
range -1, 0 to 100, default is -1

NOTE: * This command use the eSpeak software, a speech synthesizer for English and other languages.
See <http://espeak.sourceforge.net>

* Until distribution V1.01 of Ya3dag, the espeak-data subdirectory was missing. This is needed to hear something from the speak software.

* There are also console commands to play around with speak.
SpeakList to enumerat all voices.
SpeakVoice to setup a voide.
Speak speak a text.

/*****

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 \$lastresult
\$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 \$lastresult
\$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 \$lastresult, the amount of items in the actors inventory can be picked up.
\$lastresult has the value of -1 if item is not existing.

CountUsedSlots \$lastresult returns the number of item slots which have any items.