GameBots 2004 Network protocol

Chapter 1: Introduction

GameBots 2004 (GB or GameBots) are a modification (mod) for the game Unreal Tournament 2004 (UT04). GameBots are written in UnrealScript (UT04 scripting language). GB provide network text protocol (TCP/IP) for connecting to UT04 and controlling in-game avatars (bots). With GB user can control bots with text commands and at the same time, he receives information about the game environment in the text format.

GB main purpose is to make available rich environment of UT04 for virtual agent development by allowing easy connection to UT04 through its text protocol. For more information about GB implementation see GB documentation.

This document provides information about GB features and GB text protocol. If you want just a list of all GB messages and commands skip to chapter 7.

Chapter 2: Text protocol

GB commands and messages share common text format. Example of the format is `"MSGTYPE {attribute1name attribute1value} {attribute2name attribute2value}"` (without the quotes). Messages and commands in GB are always parsed like this:

- First characters up to, but not including the first space are the message type (The length of message type can vary, but usually it is three or four characters)
- Then enclosed by "{}" follow attribute pairs. One attribute pair consists of attribute name - which is defined by all characters after "{" until the first space (not including) - and of attribute value - which is defined by the rest of the characters after the first space (excluding it) until "}". The attribute value can include spaces.
- Each attribute pair should be separated by a space after character "}"

So the correct parsing of example message `"MSG {Location 100,150,0} {Rotation 32000,0,0} {String Help me!}"` would be:

Message type = "MSG"
Attribute1Type = "Location"
Attribute1Value = "100,150,0"
Attribute2Type = "Rotation"
Attribute2Value = "32000,0,0"
Attribute3Type = "String"
Attribute3Value = "Help me!"

Chapter 3: Data Types Note

Unreal Tournament features its own "UT units" for measuring the location and rotation. UT units have no real correspondence to the real world, but as an idea, a character in the game has a collision cylinder (a cylinder tightly bounding the graphical model that defines how close something has to be before it collides with the character) 17 units in radius and 39 units tall.

Location of the bot consists of three values X,Y and Z. X and Y are "flat" and Z sets the height of the object in a game. These values are sent to GB separated by commas (example MSG {Location 100,100,100}).

Rotation of the bot consists of pitch, yaw and roll. Yaw is side to side, pitch up and down, and Roll the equivalent of doing a cartwheel. Again it is passed to GB separated by commas (example MSG {Rotation 32000,0,0}). A full rotation using UT's measurements is 65535. To convert the values you are sent to radians, divide by 65535 and multiply by 2 * pi.

Chapter 4: GameBots connections

First of all you need to create GameBots server. That in fact means, you need to create UT 2004 server, that will use the GameBots extension. There are two ways how to do this. First way is to start a dedicated server through console command. Example (this will set GameBots BotDeathMatch on the map DM-TrainingDay):

```
YOUR_UT04_FOLDER\System\ucc server DM-TrainingDay?game=BotAPI.BotDeathMatch
```

Second way is to start the server thtough in-game graphical interface. You need to select Host Game in the game menu, there pick some GameBots gametype (BotDeathMatch for example). Then select a map, add some mutators if you want and after you are finished click listen. The server will be started and you will be automaticly connected to the game as a player.

So now, when the server is on, how can you connect to it? GameBots use text protocol and TCP/IP. So any client capable of connecting to some port through TCP is capable to communicate with GameBots. Bad thing is that you will soon get overhelmed by GB synchronous messages, so it is a good idea to write some interface for communication with GB (for example Pogamut2).

Now what ports the GameBots use. The GameBots supports two different types of connections - BotConnection (usually port 3000) and ControlServer connection (usually port 3001). The ports can be changed in .ini file for example, or by attributes in console command (see chapter 8). The purpose of BotConnection is spawning and controlling the bot in the game, also you will get the information about the bot surrounding by this connection (through synchronous and asynchronous messages). The purpose of ControlConnection is controlling the game server, changing map, kicking players etc.

Chapter 5: Initiating the communication

After the connecting to GB server you will receive the HELLO message (HELLO BOT for BotConnection or HELLO CONTROL SERVER for ControlConnection). The server is now in waiting state. You can send READY command and server responds with game NFO message and list of all navigation points in the map. After the READY command if you are connected as a BotConnection you usually send INIT command, which spawns the bot and you can use the rest of commands for controlling the bot. After INIT command you will receive CONFCH message with actual bot configuration.

You can skip sending of READY command for both the bot and control connection, but you wont receive NFO message. List of navigation points can be requested later by GETNAVS command for both Bot and ControlConnection, but it will lag the connection for a while (maps usually include a lot of navigation points).

Old GameBots Note: If you are familiar with old GB and want to switch back to old initiation you can change in ini file variable bNewProtocol from True to False for both bot and control connection. You will receive then NFO message after the connection to GB instead of HELLO message.

Password Note: GameBots now support password by SETPASS command of ControlServer. If there is a password set, the initiation then looks as follows:

1) A ---> GB sends HELLO message

2) B <--- user sends READY

3) C ... if the server is protected by pass:

3.a) C1 ---> GB sends PASSWORD {BlockedByIP="ip adress with port of the blocker (195.113.12.3:3001)"}

3.b) C2 <--- user sends password (PASSWORD {Password pass})

... if correct

3.c) C3 ---> GB sends PASSWDOK and info batch messages, user can continue normally

.... if the password is wrong

3.d) C4 ---> PASSWDWRONG and connection ends

Chapter 6: Types of Messages

GB features commands and messages. Messages are divided into three categories Synchronous messages, asynchronous messages and batch info messages.

1 Synchronous messages

Synchronous messages will come to your client in a batch at a configurable interval. They include things like a visual update of what the bot sees and a status report of the bot itself. At the start of a batch, the server transmits a "BEG" message marked with a time stamp. All messages received until an "END" message with the same time stamp are part of the synchronous batch. They are all sent at the same instant of game time and thus refer to a single discrete state of the game.

2 Asynchronous messages

Asynchronous messages come as events happen in the game. (Although they will never appear between a "BEG" and its associated "END"). They represent things that may happen at any point in the game at random, less frequent intervals such as taking damage, a message broadcast by another player, or running into a wall. You can always be sure that event triggering an asynchronous message occurred in game time between the synchronous batches before and after it, but there is no guarantee that an asynchronous message refers to the same discrete state of the game that any other message does.

3 Batch info messages

Batch info messages are usually asynchronous and are sent as an response of a request (done by some command). Number of info messages can vary, so they are enclosed by beginning and ending message (beginning message starts always with S, all info messages with I and ending message with E). They provide additional information about a map or a game (list of all navigation points, all players, all available maps, etc.).

4 Commands

There are two types of commands - the Control server commands and the Bot commands. Control server commands are used to control the game mechanics, getting additional info about current game and setting bots and players in the game (kicking them, etc.). Bot commands are used to control the bot - his movement, his rotation, … and to set his internal variables. Both of these types of commands are formated like the server messages - a command name, followed by zero or more arguments with values, each surrounded by "{}" and separated by spaces. For example the message to initialize your bot with a name of MYBOT on team 1 would look like this (sans quotes): `"INIT {Team 1} {Name MYBOT}"`.

Parsing at the server is case insensitive. It should not matter what case you send commands, argument names,

and their values in. Arguments may also be supplied in any order. The above example could have passed the name before the team and the command would have been the same. There are however some commands that have multiple options for how to specify a desired value. A good example is the RUNTO command, which can take the Id of an object or player, or a Location in the world. You can send either or both, but the server will only use the first one it parses (order for each command type is listed below).

Note that most commands have persistent effects. Movement and rotation, once started, will continue until you reach your destination. Start shooting and you will keep shooting. There is NO advantage to sending commands repeatedly. It is quite likely that some kind of filter will be put in to discourage spamming the server.

Chapter 7: List of all GameBots messages and commands

In this chapter all available GB messages and commands will be listed. We will divide them into categories - first two main categories - Bot messages and commands and Control server messages and commands, then to subcategories - bot synchronous messages, bot asynchronous messages, bot batch info messages, bot action commands, bot configure commands and control server messages and control server commands.

1 Bot messages and commands

1.1 Synchronous messages

ATR

results of automatically casted rays. New rays to be continuously launched can be added by ADDRAY command and removed by REMOVERAY command.

Id

a unique id for this ray, assigned when adding ray. See ADDRAY

From

source point of the ray

To

target point of the ray

FastTrace

boolean if function fasttrace (faster, but less information) was used

Result

boolean result if ray hit something or not, if true we have hit something

HitNormal

vector with normal of the plane we have hit (only if NOT using FastTrace)

HitLocation

vector with location, where we have hit something (only if NOT using FastTrace)

TraceActors

boolean if we traced also actors with this ray (actors – moving things in a game – bots, players, monsters …) (only if NOT using FastTrace)

HitId

string with an Id of the thing we have hit (only if NOT using FastTrace)

BEG

Begin of a synchronous batch

Time

time stamp from the game

END

end of a synchronous batch

Time

time stamp from the game

FLG

a flag (Only for CTF games).

Id

a unique id for this flag, assigned by the game

Location

an absolute location of the flag

Holder

the identity of player/bot holding the flag (only sent if flag is being carried).

Team

the team whose flag this is

Reachable

true if the bot can run here directly, false otherwise

State

whether the flag is "Held" "Dropped" or "Home"

GAM

information about the game

PlayerScores

player score will have a list of values - one for each player in the game. Each value will be a list with two values. The first is the id of the player and the second that player's score. (e.g. `"GAM {PlayerScore {player1 2} {player2 5}...}..."`)

TeamScores

like PlayerScore, but for teams. Team is identified by the team index (same number used to describe team for PLR and SLF messages. Not sent in normal deathmatch

DomPoints

like the previous two, this is a multivalued message. This will have one item for each domination point in a Domination game. First value will be Id of the DOM point, the second will be the index of the team that owns the domination point

HaveFlag

sent in CTF games if the bot is carrying an enemy's flag. Value is the team number of whose flag you have.

EnemyHasFlag

sent in CTF games if the bots' team's flag has been stolen. Value is meaningless

INV

an object on the ground that can be picked up

Id

a unique id for this inventory item, assigned by the game.

Location

an absolute location

Reachable

true if the bot can run here directly, false otherwise

Class

a string representing type of object

MOV

a "mover". These can be doors, elevators, or any other chunk of architecture that can move. They generally need to be either run into, or activated by shooting or pressing a button. We are working on ways to provide bots with more of the information they need to deal with movers appropriately.

Id

a unique id for this mover, assigned by the game

Location

an absolute location

Reachable

true if the bot can run here mover, false otherwise

DamageTrig

true if the mover needs to be shot to activated.

Class

class of the mover

IsMoving

boolean, true if the mover is actually moving

Velocity

vector with the velocity of the mover

NAV

a path node in the game. Pathnodes are invisible (at least to humans) objects placed around a level to define paths for the built in bots to follow. They provide a totally connected graph that spans almost all of the level. Note the Mutator called "Path Markers" that, when added to a game makes the path nodes visible to human players as a debugging aid.

Id

a unique id for this pathnode, assigned by the game

Location

an absolute location

Visible

true if the navpoint is in the bots' vision radius, false otherwise

Reachable

true if the bot can run here directly, false otherwise

Item

Holds respawned item at this NavigationPoint if is any (otherwise "None").

Flag

What type is this NavigationPoint. The types are: PathNode, PlayerStart, InventorySpot and AIMarker. If the type is AIMarker, more attributes appear in NAV message - see below.

Rotation

If the type is AIMarker. The rotation the bot should be facing, when doing the action specified by AIMarker.

RoamingSpot

boolean. Some ambush point, where is good chance to intercept approaching opponents.

SnipingSpot

boolean. Point good for sniping.

PreferedWeapon

Class of the weapon that should be prefered when using this point for AIMarker specified action.

DOM

identical attributes to NAV above except for Controller (see above). Represents a domination point in BotDoubleDomination game.

Controller

which team controls this point

PLR

Another character (bot or human) in the game. Only reports those players that are visible. (within field of view and not occluded).

Id

a unique id for this player, assigned by the game

Rotation

which direction the player is facing in absolute terms

Location

an absolute location for the player

Velocity

absolute velocity in UT units

Name

name of the player visible in a game

Team

what team the player is on.

Reachable

true if the bot can run to this other player directly, false otherwise. Possible reasons for false: pit or obstacle between the two characters

Weapon

what class of weapon the character is holding

Firing

0 means is not firing, 1 - firing, 2 - alt firing

SLF

information about your bots' state.

Id

a unique id, assigned by the game

Rotation

which direction the player is facing in absolute terms

Location

an absolute location

Velocity

absolute velocity in UT units

Name

players human readable name

Team

what team the player is on. 255 is no team. 0-3 are red, blue, green, gold in that order

Health

how much health the bot has left. Starts at 100, ranges from 0 to 199

Weapon

weapon the player is holding. Look on the list of all Unreal Tournament 2004 weapons to know what strings to look for

Shooting
if the bot is shooting, it is True, False otherwise

CurrentAmmo
how much ammo the bot has left for current weapon

Armor
how much armor the bot is wearing. Starts at 0, can range up to 199.

AltFiring
1 if the bot shoots and use alternate fire, 0 otherwise

1.2 Asynchronous messages

AIN
added inventory. Bot got new inventory item.

Id
a unique id for this inventory item, assigned by the game. Unique, but based on a string describing the item type

Class
a string representing type of the object

BMP
bumped another actor.

Id
unique id of actor (actors include other players and other physical objects that can block your path

Location
location of thing you rammed

CONFCH
this message is sent when variables of the bot are changed – by CONF command of control server, or of this bot.

Id
id of the bot

ManualSpawn
boolean, if set to true, bot wont respawn automatically after death, but RESPAWN command will have to be called

AutoTrace
boolean, enables or disables bot auto ray trace (If ATR message will be sent or not)

Name
string, current name of the bot

Invulnerable
boolean, if true the bot cant be killed. This can be changed just when cheating is enabled on the server (bAllowCheats = True)

VisionTime
float, ranges from 0.1 to 2 seconds. This will change the period between two synchronous batches

ShowDebug
boolean, if true some additional debug information will be logged to server window

ShowFocalPoint
boolean, if set to true an actor will appear in the game on the location the bot is actually looking at

DrawTraceLines
boolean, if set to true, the rays of automatic ray tracing (ATR messages) will be drawn in the game.

Has some issues, on some UT2004 copies this does not work. We are trying to fix this

SynchronousOff

boolean. It informs if sending of all GB synchronous messages is enables/disables.

CWP

bot changed weapons. Possibly as a result of a command sent by you, maybe just because it ran out of ammo in its old gun. (bots auto switch when empty, just like human players).

Id

unique id of new weapon, based on the weapon's name

Class

a string representing type of weapon

DAM

took damage

Damage

amount of damage taken

DamageType

a string describing what kind of damage

Instigator

if we see attacker, we will send his Id here

DIE

this bot died

Killer

unique ID of player that killed the bot if any (may have walked off a ledge)

DamageType

a string describing what kind of damage killed the bot

FAL

bot just hit a ledge. If walking, will not fall. If running, you are already falling by the time you get this.

Fell

true if you fell. False if you stopped at edge

Location

absolute location of bot

FIN

no attributes. Sent when game is over or the bot is kicked or the map is changed (sent right after MAPCHANGE message).

FTR

response of the FASTTRACE command. Note that trace commands are computationally expensive.

Id

an id matching the one sent by client. Allows bot to match answer with right query

From

source point of the ray

To

target point of the ray

Result

boolean result of FastTrace, true if the ray has hit something

HIT

hurt another player. Hit them with a shot.

Id

unique ID of player hit

Damage

amount of damage done

DamageType

a string describing what kind of damage

HELLO BOT

sent right after you make connection to GameBots. Nothing happen after that, the server will be waiting for READY or INIT command.

HRN

hear noise. Maybe another player walking or shooting, maybe a bullet hitting the floor, or just a nearby lift going up or down.

Source

unique ID of actor making the noise

Rotation

the rotation from where the sound comes

HRP

hear pickup. You hear someone pick up an object from the ground.

Name

name of the item

SourceClass

class of the item

Rotation

the rotation from where the sound comes

JOIN

sent when player joins the server.

Id

an id of the joining player

Name

the name of the joining player

KIL

some other player died

Id

unique ID of player

Killer

unique ID of player that killed them if any (may have walked off a ledge)

DamageType

a string describing what kind of damage killed them

LEFT

sent when player leaves the server.

Id

an id of the leaving player

Name

the name of the joining player

LIN

Lost inventory message.

Id

an id of an object we have lost from our inventory chain

MAPCHANGE

sent when the map is changed (bot will lost the connection).

MapName

text name of the map

NFO

helpful info about the game provided right after you respond to HELLO message by READY command. Your should have this information BEFORE sending "INIT" back to the server.

Gametype

what you are playing (BotDeathMatch, BotTeamGame, …)

Level

name of map in play

TimeLimit

maximum time game will last

FragLimit

number of kills needed to win game (BotDeathMatch only)

GoalTeamScore

number of points a team needs to win game (BotTeamGame, BotCTFGame, BotDoubleDomination)

MaxTeams

max number of teams. Valid team range will be 0 to (MaxTeams – 1) (BotTeamGame, BotCTFGame, BotDoubleDomination)

MaxTeamSize

max number of players per side (BotTeamGame, BotCTFGame, BotDoubleDomination)

GamePaused

boolean, true if the game is currently paused

BotsPaused

boolean, true if just the bots are paused (if GamePaused true, everyone will be paused even if this is set to true or false)

PAUSED

sent when the or the bots are paused.

PRJ

incoming projectile likely to hit you. May give you a chance to dodge.

Time

estimated time till impact

Direction

rotation value that the projectile is coming from. Best chance to dodge is to probably head off at a rotation normal to this one (add ~ 16000 to the yaw value)

Origin

the location the projectile is flying from

DamageRadius

if the projectile has splash damage, how big it is – in ut units

Class

the class of the projectile (so you know what is flying against you)

PTH

a series of pathnodes in response to a getpath call from client

Id

an ID matching the one sent by client. Allows bot to match answer with right query.

Multiple pathnodes: A variable number of attr items will be returned, one for each path node that needs to be taken. They will be listed in the order in which they should be traveled to. Each one is of form "{**number** NavPointId NavPointLocation}", with the number of the node (starting with 0) followed by a space, then a id of the node, then a location of the node. Example: `"PTH {Id Path1} {0 PathNode31 124,2134,0} {1 PathNode22 124,1000,0} {2 PathNode4 124,78,0}"` Maximum length is 16.

RCH

a boolean result of a checkreach call.

Id

an ID matching the one sent by client. Allows bot to match answer with right query

Reachable

true if the bot can run here directly, false otherwise

From

exact location of bot at time of check

RECEND

sent as a response to STOPREC command.

RECSTART

sent as a response to REC command.

RESUMED

when the game and the bots are unpaused.

SEE

see player. A message generated by the engine periodically (on the order of 1 or 2 times a second) when another player is visible by you. Possibly useful if you have the delay between synchronous updates very long. In that case, this can prevent someone from walking by unseen. May be deprecated.

Id

a unique id for this player, assigned by the game

Rotation

which direction the player is facing in absolute terms

Location

an absolute location for the player

Velocity

absolute velocity in UT units

Name

name of the player in the game

Team

what team the player is on.

Reachable

true if the bot can run to this other player directly, false otherwise. Possible reasons for false: pit or obstacle between the two characters

Weapon

what weapon the character is holding

Firing

0 means is not firing, 1 - firing, 2 - alt firing

SPW

you get this every time the bot gets respawned. When the match is not started yet and you connect to the server, there is delay in sending this message, it will be sent when the match will start, although the bot will be spawned in the game for a few seconds at that time.

THROWN

send if THROW command ends successfully (bot will drop a weapon).

TRC

response of the TRACE command. Note that trace commands are computationally expensive.

Id

an ID matching the one sent by client. Allows bot to match answer with right query

From

source point of the ray

To

target point of the ray

Result

boolean result of Trace, true if the ray has hit something

HitNormal

vector of normal of a plane we have hit

HitLocation

vector of an exact point, where the ray has hit something

HitId

string Id of an object we have hit – can be level geometry etc

VCH

some part of the bot body changed the zone (volume changed).

Id

unique id of zone entered

PainCausing

true or false if we get some damage when we stay at this zone

VMS

received message from global chat channel.

Name

name of the sender.

String

a human readable message sent by another player in the game on the global channel

VMT

received message from team chat channel.

Name

name of the sender.

String

a human readable message sent by a team mate in the game on the private team channel

WAL

collided with a wall. Note it is common to get a bunch of these when you try to run through a wall (or are pushed into one by gunfire or something).

Id

unique id of wall hit

Normal

normal of the angle bot collided at

Location

absolute location of bot at time of impact

ZCB

bot changed zones. Entire bot now in new zone.

Id

unique id of zone entered

1.3 Batch info messages

1.3.1 Navpoint info

info about all existing navpoints in current map. You will get a bunch of INAV messages, one for each NavPoint. This batch will start with SNAV message and end with ENAV message. It will be send after READY command, or by GETNAVS command.

SNAV

start of navpoint message block.

INAV

info about one navpoint.

Id

a unique id for this pathnode, assigned by the game

Location

absolute location of the NavPoint

Item

if this is an inventory spot, where some inventory gets respawned, here will be the Id of respawned inventory, will be "None" otherwise.

Flag

What type is this NavigationPoint. The types are: PathNode, PlayerStart, InventorySpot and AIMarker. If the type is AIMarker, more attributes appear in NAV message - see below.

Rotation

If the type is AIMarker. The rotation the bot should be facing, when doing the action specified by AIMarker.

RoamingSpot

boolean. Some ambush point, where is good chance to intercept approaching opponents.

SnipingSpot

boolean. Point good for sniping.

PreferedWeapon

Class of the weapon that should be prefered when using this point for AIMarker specified action.

Neigh<number>

(Neigh0, Neight1, etc…) here are information about one NavPoint reachable from current NavPoint.

One NavPoint can have more neighbours. Syntax of Neigh attribute is a bit changed. Example: `{Neigh0 {Id DMFlux. PathNode23} {Flags 3} {CollisionR 100} {CollisionH 100}}`

Id
Id of the neighbouring NavPoint

Flags
flags about the path, what is required to be able to go from Nav to neighbour is stored as an integer, see UnrealWiki for more

CollisionR
how wide can we be to pass this path

CollisionH
how big can we be to pass this path

ENAV
end of navpoint message block.

1.3.2 Inventory info

info about all respawned pickup items in current map (will not include dropped weapons.). You will get a bunch of IINV messages, one message for each pickup. This batch will start with SINV message and end with EINV message. It will be sent as an response to READY command or GETINVS command.

SINV
start of inventory message block.

IINV
info about one inventory item.

Id
a unique id for this item, assigned by the game

Location
absolute location

Class
a string representing type of object

EINV
end of inventory message block.

1.4 Bot configure commands

ADDINV
adds inventory of supported class to the bot. Bot has to be alive. The change is not permanent. This can be used just when cheating is enabled on the server. (bAllowCheats = True). Any Pickup class can be supported in the class variable. See the available classes in UnrealEd.

Class
string of the name of the class. `Example: ADDINV {Class xWeapons.FlakCannonPickup}`

ADDRAY
will add custom ray for auto tracing.

Id
Unique Id of the ray. If you send Id = Default, all rays will be erased and default set of rays will be loaded (straight ahead (1,0,0) with 250 length, 45 degrees left (1,−1,0) with 200 length, 45 degrees right (1,1,0) with 200 length). This set of rays is also loaded by default. If you want to change existing ray, just support his Id in ADDRAY command along with new parameters.

Direction

Direction of the ray. Bot is looking to x axis, that means if I want ray straight ahead I specify some vector on positive x axis (vectors in unreal are specified by (x,y,z) so it would look like this (1,0,0) or this (123,0,0) – numbers doesn't matter, its about direction). If I want ray behind it would be (–1,0,0). 90 degrees right (0,1,0) etc.

Length

float in ut units. Specifies the length of the ray

FastTrace

boolean, true if we want to use FastTrace function (faster) instead of Trace function

TraceActors

boolean, true if we want to trace also actors – bots, monsters, players. False if we want to trace just level geometry

CONF

this command configures features of the bot

AutoTrace

enables AutoTrace feature

ManualSpawn

sets if the bot will be respawned after death manually by RESPAWN command or automatically

Name

you can change name of the bot in the game

Invulnerable

will set godmode for bot on (bot can't be killed) This can be changed just when cheating is enabled on the server. (bAllowCheats = True)

VisionTime

between 0.1 to 2 seconds, it sets how long should be GB idle before running next checkvision test

ShowDebug

boolean, if true some additional debug information will be logged to server window

ShowFocalPoint

boolean, if set to true an actor will appear in the game on the location the bot is actually looking at

DrawTraceLines

boolean, if set to true, the rays of automatic ray tracing (ATR messages) will be drawn in the game. Has some issues, on some UT2004 copies this does not work. We are trying to fix this

SynchronousOff

boolean. It enables/disables sending of all GB synchronous messages.

CHATTR

will change specified attribute of the bot. Now just attribute Health can be changed.

Health

sets the bot health, can range from 1 to 199. The bot has to be alive. This change is not permanent (after respawn, bot will start with 100 health again)

CHECKREACH

check to see if you can move directly to a destination without hitting an obstruction, falling in a pit, etc...

Target

the unique id of a player/object/nav point/whatever. Must be visible

Location

location you want to go to. Normal location rules. Only used if no Target is sent

Id

message id made up by you and echoed in response so you can match up response with query

FTRACE

will send a ray from specified location to specified destination, responds with FTR message. FTRACE uses FastTrace function, which is faster then Trace function, but still rather slow.

Id

message id made up by you and echoed in response so you can match up response with query

From

origin point of the trace ray. If you wont support From attribute, current bot location will be taken as From

To

target point of trace ray

GETINVS

in response to this command, server will send you the IINV info batch message.

GETNAVS

in response to this command, server will send again the INAV info batch message.

GETPATH

get a path to a specified location. An ordered list of path nodes will be returned to you.

Location

location you want to go to. Normal location rules

Id

message ID made up by you and echoed in response so you can match up response with query

GIVEINV

gives inventory from one bot to another. Bot can give just owned items. (in his inventory chain). He cant give weapon he is wielding. This command is not fully tested yet and may have issues.

Target

the bot who is receiving the item

ItemId

id of the item the bot is giving to another bot

INIT

message you send to spawn a bot in the game world. You must send this message before you have a character to play in the game.

Name

desired name. If in use already or argument not provided, one will be provided for you

Team

preferred team. If illegal or no team provided, one will be provided for you. Normally a team game has team 0 and team 1. In BotDeathMatch, team is meaningless

ManualSpawn

sets if the bot will be respawned after death manually by RESPAWN command or automatically

AutoTrace

enables auto tracing (it can be enabled later by CONF command)

Location

specify start location, if unspecified, then random

Rotation

specify start rotation, if unspecified, then random

Skin

sets the bot current skin, for more information look at SETSKIN command

DesiredSkill

Can range from 0 to 7 - it is float, but you should use just integers (1,2,...). This sets the bot accuracy. 1 lowest, 7 highest. Shouldn't have any other effect.

DesiredAccuracy

float from 0 to 1. This should tweak bot accuracy a little. So far it had very little effect on bot accuracy. When you want to change accuracy significantly use DesiredSkill attribute.

ShouldLeadTarget

boolean. When firing slow projectiles (missiles...), if the engine will try to count the impact point for the bot or not.

PASSWORD

Send password to the server. For more information see ControlServer command SETPASS.

Password

string. Holds the password.

PAUSE

command, will pause/unpause the game.

PauseBots

if true only bots will be paused – players and spectators will move freely

PauseAll

everyone in the game will be paused if set to true. To unpause send PAUSE command with PauseAll set to false

QUIT

will shutdown the connection and kill and remove the bot from a game, the same can be achieved just by closing the connection.

READY

command you should send after server HELLO message. The server will send you game NFO message and INAV info batch message.

REC

server will start recording demo of current game. Command is confirmed by RECSTART message.

FileName

name of the saved demo file

REMOVERAY

will remove a ray from auto ray trace specified by Id

Id

of the ray to be removed. If Id = "All" all rays will be removed

RESPAWN

use this to kill bot and force him to respawn, you can specify start location and rotation.

StartLocation

where bot respawns. If you want to respawn bot at random, don't specify StartLocation

StartRotation

initial rotation of the bot

STOPREC

will stop recording a demo. Is confirmed by RECEND message.

TRACE

will send a ray from specified location to specified destination, responds with TRC message. TRACE uses Trace function, which can be rather slow.

Id

message id made up by you and echoed in response so you can match up response with query

From

origin point of the trace ray. If you wont support From attribute, current bot location will be taken as From

To

target point of trace ray

TraceActors

boolean, when true it means that all actors will be traced – for example players, bots, monsters etc. in a game. With TraceActors false we trace just level geometry

1.5 Bot action commands

CHANGEWEAPON

switch your weapon

Id

unique Id of weapon to switch to. If you just send "Best" as Id, the server will pick your best weapon that still has ammo for you. Obtain Unique Id's from AIN events

CMOVE

the bot will continuously move straight ahead according to his actual rotation.

Speed

sets the speed of the movement. Ranges from 0.1 to 2. This number multiplies default speed of the bot

JUMP

causes bot to jump.

MESSAGE

send a message to the world or just your team. This will likely have some restrictions placed on it soon.

Text

string to send

Global

if True it is sent to everyone. Otherwise (or if not specified), just your team

MOVE

the bot will continuously move first to Location1 and then to Location2, movement between locations will be continuous without any delay.

Speed

sets the speed of the movement. Ranges from 0.1 to 2. This number multiplies default speed of the bot

Location1

vector of the first location

Location2

vector of the second location

ROTATE

turn a specified amount.

Amount

amount in UT units to rotate. May be negative to rotate counter clockwise

Axis

if provided as Vertical, rotation will be done to Pitch. Any other value, or not provided, and rotation will be to Yaw

RUNTO

turn towards and move directly to your destination. May specify destination via either Target or Location argument, will be parsed in that order. (i.e. if Target provided, Location will be ignored). If you select an impossible place to head to, you will start off directly towards it until you hit a wall, fall off a cliff, or otherwise discover the offending obstacle.

Speed

sets the speed of the movement. Ranges from 0.1 to 2. This number multiplies default speed of the bot

Target

the unique id of a player/object/nav point/whatever. The object must be visible to you when the command is received or your bot will do nothing. Note that something that was just visible may not be when the command is received, therefore it is recommended you supply a Location instead of a Target.

Location

location you want to go to. Must be provided as comma delimited ("40,50,45")

SETCROUCH

will crouch/uncrouch the bot.

Crouch

if true the bot will crouch, if false the bot will uncrouch, after respawn this will be reset - bot will uncrouch

SETSKIN

set the current bot skin – the appearance of the bot can be changed by this. The bot will be respawned after this command with the new skin.

Skin

string, which specifies the bot appearance. You need to supply package and skin (mesh) name. Example: SETSKIN {Skin HumanMaleA.MercMaleA}. Find all packages and skins through unrealEd (Actor browser, search in UT2004/Animations folder). Some native packages to look in: HumanMaleA, HumanFemaleA, Bot, Aliens.

SETWALK

set whether you are walking or running (default is run).

Walk

Send "True" to go into walk mode – you move at about 1/3 normal speed, make less noise, and wont fall off ledges. Send "False" to disable walking

SHOOT

start firing your weapon.

Location

location you want to shoot at. Normal rules for a location specification

Target

the unique id of your target. If you specify a target, and it is visible to you, the server will provide aim correction and target leading for you. If not you just shoot at the location specified

Alt

if you send True to this you will alt fire instead of normal fire. For normal fire you don't need to send this argument at all

STOP

stop all movement/rotation.

STOPSHOOT

stop firing your weapon

STRAFE

like RUNTO, but you move towards a destination while facing another object. You must specify some object in a game by id (can be navpoint, some inventory or another player, etc.). This changed from UT 2000, where you could specify just location.

Speed

sets the speed of the movement. Ranges from 0.1 to 2. This number multiplies default speed of the bot

Location

location you want to go to. Must be provided as comma delimited ("40,50,45")

Target

the unique id of a player/object/nav point/whatever that you want to face while moving

Focus

location of the spot you want to look at while moving to location. This will be used, if you wont support target attribute

THROW

without any parameters. Will drop your current weapon (if it is possible) and will change to best weapon available. If done successfully, message THROWN will come right away.

TURNTO

specify a point, rotation value or object to turn towards.

Target

the unique id of a player/object/nav point/whatever that you want to face. Must be visible

Rotation

rotation you want to spin to. Must be provided as comma delimited ("0,50000,0") and should be in absolute terms and in UT units (2pi = 65535 units). Used only if no target provided.

Location

location you want to face. Normal rules for location. Only used if no Target or Rotation

2 ControlServer commands and messages

ControlServer runs at port 3001 (if enabled - bAllowControlServer = True). It provides commands and messages for controlling the game server, managing bots and maps and setting up the game.

2.1 ControlServer commands

ADDBOT

Will add original epic bot to a game. May have issues with team balancing.

Name

optional name of the bot

StartLocation

optional start location of the bot

StartRotation

optional start rotation of the bot

Skill

float, optional skill of the bot - from 1 to 7 (best)

Team

integer number of desired team (usually 0 or 1)

ADDINV

adds inventory of supported class to the bot specified by Id. Bot has to be alive and on the server. The change is not permanent. Every non abstract Pickup class can be supported in Class attribute, see UnrealEd for complete list.

Id

Id of the bot, we want to add inventory to

Class

string of the name of the class. Example: {Class xWeapons.FlakCannonPickup}

CHANGEMAP

will change map to MapName - map must exist on server (wont be tested), will send MAPCHANGE message.

MapName

name of the new map

CHATTR

will change specified attribute of the bot. Now just attribute Health can be changed.

Id

Id of the target bot

Health

sets the bot health, can range from 1 to 199. The bot has to be alive. This change is not permanent

CONF

this command configures features of the bot, who is specified by Id.

Id

Id of the bot to be configured

AutoTrace

boolean, enables or disables bot auto ray trace. (If ATR message will be sent or not)

ManualSpawn

boolean, if set to true, bot wont respawn automatically after death, but RESPAWN command will have to be called

Name

string, will change the name of the bot in the game

Invulnerable

boolean, if true the bot cant be killed. This can be changed just when cheating is enabled on the server (bAllowCheats = True)

VisionTime

between 0.1 to 2 seconds, it sets how long should be GB idle before running next checkvision test

ShowDebug

boolean, if true some additional debug information will be logged to server window

ShowFocalPoint

boolean, if set to true an actor will appear in the game on the location the bot is actually looking at

DrawTraceLines

boolean, if set to true, the rays of automatic ray tracing (ATR messages) will be drawn in the game. Has some issues, on some UT2004 copies this does not work. We are trying to fix this

SynchronousOff

boolean. It enables/disables sending of all GB synchronous messages

CONSOLE

You can run all console command by this.

Command

string. The exact console command (as you would type to console).

ENDPLRS

Will stop exporting of IPLR messages synchronously.

GETMAPS

Will request map list from the server. Server will respond with IMAP batch map info message.

GETNAVS

Will send list of all navpoints in the map with reachability graph. Standard IINV batch info message will be used.

GETPLRS

Will send list of all players currently playing on the server. Server will respond with IPLR batch players info message.

KICK

will kick RemoteBot from the server.

Id

of the bot to be kicked

PASSWORD

Send password to the server. For more information see ControlServer command SETPASS.

Password

string. Holds the password.

PAUSE

will pause/unpause the game.

PauseBots

if true only bots will be paused - players and spectators will move freely

PauseAll

everyone in the game will be paused if set to true

PING

command for connection detection. Server will return "PONG".

QUIT

will close the connection.

READY

command. Response to HELLO message. The server will send standard game NFO message.

REC

server will start recording demo from current game, this command is not fully tested yet. Command is confirmed by RECSTART message.

FileName

name of the saved demo file

RESPAWN

will force bot to respawn, that means – it will kill the bot and spawn him on new location – random or specified.

Id

of the bot to respawn

Location

optional vector, specifies respawn location

Rotation

optional vector, specifies respawn rotation

SETGAMESPEED

will set speed of the game.

Speed

can range from 0.1 to 50. 1 is normal game speed. The reasonable speeding up is around 10. The game engine stops catching up at higher values.

SETLOCK

Will disable new connections to botserver and or control server - depends on parameters. If last ControlServer instance is leaving. ControlServer lock will be canceled.

BotServer

boolean. If BotConnections should be locked.

ControlServer

boolean. If ControlConnections should be locked.

SETPASS

Sets the password for Bot and control connections. If the password is set the initiation of GB communication is changed to this:

1) A ---> GB sends HELLO message

2) B <--- user sends READY

3) C ... if the server is protected by pass:

3.a) C1 ---> GB sends PASSWORD {BlockedByIP="ip adress with port of the blocker (195.113.12.3:3001)"}

3.b) C2 <--- user sends password (PASSWORD {Password pass}) ... if correct

3.c) C3 ---> GB sends PASSWDOK and info batch messages, user can continue normally .... if the password is wrong

3.d) C4 ---> PASSWDWRONG and connection ends

Note: If the user knows that server is passworded he can use PASSWORD {Password pass} command instead of READY, and will be checked and sent info batch messages immediately

Password

string. Sets the password.

STARTPLRS

Will start to export IPLR messages regularly (like synchronous batch). Can be used for continuous visualization of players moving around the map. There are three categories (see below). The default

values for all category is true, that means that without attributes all the categories will be exported.

Humans
boolean. All human players will be exported.

GBBots
boolean. All GameBots bots will be exported.

UnrealBots
boolean. All UnrealBots will be exported.

STOPREC
will stop recording a demo. Is confirmed by RECEND message.

2.2 ControlServer messages

ALIVE
Normally sent once per second, if periodic exporting of players is enabled in ControlServer, it will be sent as often as is the update time of player export.

Time
Reflecting current game time.

CONFCH
this message is sent when variables of the bot are changed – by CONF command of control server, or of this bot. And also when bot joins the game. Then every Control server connected will receive this message after JOIN message.

Id
id of the bot

ManualSpawn
boolean, if set to true, bot wont respawn automatically after death, but RESPAWN command will have to be called

AutoTrace
boolean, enables or disables bot auto ray trace (If ATR message will be sent or not).

Name
string, current name of the bot

Invulnerable
boolean, if true the bot cant be killed. This can be changed just when cheating is enabled on the server (bAllowCheats = True)

VisionTime
float, ranges from 0.1 to 2 seconds. This will change the period between two synchronous batches

ShowDebug
boolean, if true some additional debug information will be logged to server window

ShowFocalPoint
boolean, if set to true an actor will appear in the game on the location the bot is actually looking at

DrawTraceLines
boolean, if set to true, the rays of automatic ray tracing (ATR messages) will be drawn in the game. Has some issues, on some UT2004 copies this does not work. We are trying to fix this

SynchronousOff
boolean. It informs if sending of all GB synchronous messages is enables/disables.

FIN
Sent when the map is changed (sent right after MAPCHANGE message).

HELLO CONTROL SERVER

message, sent immediately after socket establish.

IMAP

batch map info message - Will begin with SMAP message and end with EMAP message. Provides information about maplist on the server. IMAP message has got one attribute - Name of the map.

Name

name of one map in map list on the server.

IPLR

batch players info message - Will begin with SPLR message and end with EPLR message. Provides information about players currently playing on the server. IPLR message has got these attributes:

Id

Unreal Id of the player

Name

name of the player

Location

vector if the player/bot is currently living

Rotation

vector if the player/bot is currently living

AutoTrace

true or false – if the bot is auto tracing

ManualSpawn

true or false – if the bot is not respawning automatically

Invulnerable

true or false – if true, bot cannot die

VisionTime

float, delay between two synchronous batches

ShowDebug

boolean, if true additional debug is displayed from this bot in server window (the console that was used to start the server)

ShowFocalPoint

boolean, if true the bots focal point is displayed in the game

DrawTraceLines

boolean, if true the automatic ray tracing lines (ATR messages) are displayed in the game. Has issues, does not work on some UT2004 copies

JOIN

sent when player joins the server.

Id

an id of the joining player

Name

the name of the joining player

LEFT

sent when player leaves the server.

Id

an id of the leaving player.

Name

the name of the leaving player

MAPCHANGE

sent when the map is changed (server will lost the connection).

MapName

text name of the map

PAUSED

sent when the or the bots are paused.

RECEND

response to STOPREC command.

RECSTART

response to REC command.

RESUMED

sent when the game and the bots are unpaused.

Chapter 8: GameBots command line parameters

Usually user will run GameBots server through command line (it is advised to do so). The syntax of command line commads looks like this:

```
YOUR_UT04_FOLDER\ucc server
Name_of_desired_map?game=GameBots_gametype?variable1=value1?variable2=value2?variable3=\
```

GameBots mod adds several special attributes, that can be used to configure GameBots without setting the things in ini file. Here is the list of GameBots parameters as well as of some usefull original UT parameters:

- Mutator variable - we select here which mutators we want to run in our game. If we want to use multiple mutators at once, we will use "," as delimiter (eg. mutator=BotAPI.GBHudMutator,BotAPI.PathMarkerMutator).
- TimeLimit variable - with this we set how long will the game lasts until mapchange in minutes. (when mapchange occurs, GameBots get disconnected).
- GoalScore variable - sets the ammount of frags, which has to be acquired for a player or bot to win the game (when somebody wins the game, the game ends, mapchange will occur and GameBots will be disconnected again).
- BotServerPort variable - the server will listen on selected port for connections. Be carefull, if the port is taken by another process or another dedicated server instance, you will not be able to connect to your server instance. This port range allowed in GameBots: 2000 - 32000.
- ControlServerPort variable - The same as above, except it is used for control server connections. Always set bot server and control server port to different variables.
- bRandomPorts variable - can be set to True or False (eg. bRandomPorts=True). If set to true random free ports for bot connection and control connection will be picked up (nevertheless the other settings). These ports will be displayed in the server console window after the beginning of the game.

So, the command that would run GameBots BotDeathMatch game type on map DM-Flux2 with both GameBots mutators on looks as follows. It will also set time limit to 20 minutes, goal score to 20 frags. The server will listen on port 3024 for bot connections and on port 3025 for control connections. Example:

```
YOUR_UT04_FOLDER\ucc server
DM-Flux2?game=BotAPI.BotDeathMatch?mutator=BotAPI.PathMarkerMutator,BotAPI.GBHudMutator?
```

Other command that would pick random ports looks like this:

```
YOUR_UT04_FOLDER\ucc server
```

```
DM-Flux2?game=BotAPI.BotDeathMatch?mutator=BotAPI.PathMarkerMutator,BotAPI.GBHudMutator?
```

Chapter 9: GameBots INI file

GB have an ini file, located in the SYSTEM folder of the game UT04 ("BotAPI.ini"). Syntax is as follows. Enclosed by a pair of "[]" is the name of the class you want to configure. Below is a list of all configurable classes with brief description of what can be configured. More information can be found directly in the ini file in comments.

- [BotAPI.RemoteBot] - here we configure remote bot variables (maximum speed, if he will be spawned manually, etc.). Some interesting variables to set here:

  bDrawTraceLines=true - boolean. If we should draw the rays defined by auto tracing behavior in the game.

  bShowFocalPoint=false - boolean. If we should draw bot focal point in the game.

  bPerfectLocationAim=false boolean. If the bot should have perfect aim also for location targets and stationary targets. If set to true bot accuracy setting will be used just when aiming to another bot or pawn (by target variable).

  bAutoTrace=false - boolean. If the auto tracing behavior is enabled.

  bAutoSpawn=true - boolean. If the bot will be automatically respawned after death.

- [BotAPI.BotConnection] - here we set if the game will be pausable for bots, if they will be allowed to cheat and also delay between synchronous messages (by VisionTime).

  bSynchronousMessagesOff=false - boolean. Enables disables synchronous messages for the bots.

  visionTime=0.250 - float. Delay between two synchronous batches in seconds.

  bAllowCheats=True - boolean. If the bots are allowed to cheat - that means if they can spawn items for themselves by ADDINV command and if they can be set to be invulnerable.

  bNewProtocol=True - boolean. If the GameBots should use new or old initiation protocol. (Pogamut2 requires new)

  bAllowPause=true - boolean. If the bots are allowed to pause the game.

- [BotAPI.ControlConnection] - this configures features of control connection.

  UpdateTime=0.3000 - float. In seconds - delay between two IPLR batches.

  bNewProtocol=True - boolean. Old or new initiation protocol. (Pogamut2 requires new)

  bAllowPause=True - boolean. Can we pause the game or not.

- [BotAPI.BotDeathMatch] - (generally [BotAPI.DesiredGameTypeToConfigure]). Here we configure desired game type. We can set time limit here, maximum number of players, goal score, etc. bAllowControlServer enables or disables possibility of control connections. Here we also set the ports on which will be run bot server and control server.

  bAllowControlServer=True - boolean. If the control server will be allowed or not .

  BotServerPort=3000 - integer. Default port for BotServer (from 2000 to 32000).

  ControlServerPort=3001 - integer. Default port for ControlServer (from 2000 to 32000).

  bRandomPorts=false - boolean. If we want to use random ports (used ports will be displayed in the server console window screen). This will override set ports.

TimeLimit=120 - integer. Time limit how long before the game ends in minutes.

GoalScore=25 - integer. How many points player needs to score to win (game will end).

- [BotAPI.BotServer] and [BotAPI.ControlServer] - here we set maximum number of connections that we will accept for control connections and for remote bots.

  MaxConnections=10 - integer. Maximum connections to the server.

The names of the attributes, that can be set are self explanatory and commented if necessary.

Chapter 10: GameBots Mutators

GB feature three mutators, which add to the game additional debugging information. It is PathMarkerMutator, GBHudMutator and GBNoWeaponMutator. They can be run by starting the server by console command - example of command that will launch GB server with both mutators from console:

```
YOUR_UT04_FOLDER\ucc server
DM-TrainingDay?game=BotAPI.BotDeathMatch?mutator=BotAPI.PathMarkerMutator,BotAPI.GBHudMu
```

Or they can be manually selected in GUI of UT04.

PathMarkerMutator visualize all navigation points in the game that are not inventory spots (inventory spots are visualized already by the respawning item).

GBNoWeaponMutator disables all weapons for both player and bots. Also it deletes all weapons from the map. It is GameBots extension dependent (might not work without it).

GBHudMutator changes players HUD, so additional information are displayed - namely current location of the player, names of surrounding navigation points and graph of navigation points reachability grid (drawn by red lines in the game). Following keys may be used to configure these features, when the game is running:

- **Insert** - enables, disables graph of navigation points reachability grid
- **Home** - enables, disables displaying names of surrounding navigation points
- **PageUp/PageDown** - increases, decreases radius of navigation points, which will have their names displayed.

Chapter 11: Acknowledgments

We would like to thank to the authors of the old GameBots (version for Unreal Tournament 2000) as our GameBots used their version as its base. And also this user documentation is extended version of their previous GameBots network API.