Gamebots API list

NOTE: This is only UP TO DATE version of GB API list. Holds for the trunk version of GB (compatible with Pogamut2.41 client)! (21.10.2009)

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.

Bot messages and commands

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
- EN 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
  - !isible true if the bot can see the flag
  - State whether the flag is "Held" "Dropped" or "Home"
- GA " information about the game
  - amage " odi#ier holds the current damage modifier for the game. (each damage is multiplied by this before applied to bots)
  - 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
  - omPoints 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
  - " yFlag sent in CTF games. Whether the flag is "Held" "Dropped" or "Home".
  - EnemyFlag sent in CTF games. Whether the flag is "Held" "Dropped" or "Home"
- IN! an object on the ground that can be picked up
  - Id a unique id for this inventory item, assigned by the game.
  - E%ent can be See or Pickup. INV can come also asynchronously now when we pick up an item. In that

case Event is Pickup.
    - Amount holds the amount of goodies we the item contains (amount of ammunition, health, shield, etc.)
    - Location an absolute location
    - Reachable true if the bot can run here directly, false otherwise
    - &lass a string representing type of object
- " ’ ! 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
    - amageTrig true if the mover needs to be shot to activated.
    - &lass class of the mover
    - Is"o%ing boolean, true if the mover is actually moving
    - !elocity vector with the velocity of the mover
- NA! 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
    - !isible 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 any (otherwise "None").
    - Item&lass Holds respawned item class at this NavigationPoint$ if 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.
    - RoamingS(ot boolean. Some ambush point, where is good chance to intercept approaching opponents.
    - Sni(ingS(ot boolean. Point good for sniping.
    - Pre#ered)ea(on Class of the weapon that should be prefered when using this point for AIMarker specified action.
- ’" identical attributes to NAV above except for Controller (see above). Represents a domination point in BotDoubleDomination$ game.
    - &ontroller 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
    - !elocity 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
    -)ea(on 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
    - !elocity 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
    -)ea(on 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
    - &urrentAmmo how much ammo the bot has left for current weapon
    - &urrentAltAmmo how much alt ammo the bot has left for current weapon (used just for AssaultRifle$, other weapons have same value here as in CurrentAmmo)
    - Adrenaline how much adrenaline the bot has. (0 to 100)
    - 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

Asynchronous messages

- A G adrenaline gained. Whenever we get new adrenaline - by pickup or by killing someone.
    - Amount amount of adrenaline gained. integer.
- AIN sent when we get new weapon or ammunition for weapon we do not have yet. Sent just once per weapon type or per new ammunition type (notify new object in our inventory, NOT pickup). For weapons we have additional attributes (otherwise just Id and Class are sent).
    - Id a unique id for this inventory item, assigned by the game. Unique, but based on a string describing the item type
    - &lass a string representing type of the object
    - Sni(ing bool. If it is sniping weapon.
    - "elee bool. If it is melee weapon.
    - PrimaryInitialAmmo int. Initial primary ammunition.
    - "a*PrimaryAmmo int. Max primary ammunition.
    - SecondaryInitialAmmo int. optional. Initial secondary ammunition (some weapons will have here the same value as for primary).
    - "a*SecondaryAmmo int. optional. Max secondary ammunition (some weapons will have here the same value as for primary).
- B"P 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
- &'NF&H 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
    - "anualS(a+n 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
    - In%ulnerable boolean, if true the bot cant be killed. This can be changed just when cheating is enabled on the server (bAllowCheats = True)
    - !isionTime float, ranges from 0.1 to 2 seconds. This will change the period between two synchronous batches
    - Sho+ ebug boolean, if true some additional debug information will be logged to server window
    - Sho+FocalPoint boolean, if set to true an actor will appear in the game on the location the bot is actually looking at
    - ra+TraceLines 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
    - Synchronous'## boolean. It informs if sending of all GB synchronous messages is enables/disables.
- &)P 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
    - &lass a string representing type of weapon
- A" took damage
    - amage amount of damage taken
    - amageTy(e a string describing what kind of damage
    - Instigator if we see attacker, we will send his Id here
- IE this bot died

    - ,iller unique ID of player that killed the bot if any (may have walked off a ledge)
    - amageTy(e 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
    - amage amount of damage done
    - amageTy(e a string describing what kind of damage
- HELL' B'T 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
    - Source class of the item
    - Rotation the rotation from where the sound comes
- -'IN sent when player joins the server.
    - Id an id of the joining player
    - Name the name of the joining player
- ,IL some other player died
    - Id unique ID of player
    - ,iller unique ID of player that killed them if any (may have walked off a ledge)
    - amageTy(e 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
- "AP&HANGE sent when the map is changed (bot will lost the connection).
    - "a(Name text name of the map
- NF' 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.

    - Gamety(e what you are playing (BotDeathMatch, BotTeamGame$, …)
    - Le%el name of map in play
    - TimeLimit maximum time game will last
    - FragLimit number of kills needed to win game (BotDeathMatch only)
    - GoalTeamScore number of points a team needs to win game (BotTeamGame, BotCTFGame, BotDoubleDomination)
    - "a*Teams max number of teams. Valid team range will be 0 to (MaxTeams – 1) (BotTeamGame, BotCTFGame, BotDoubleDomination)
    - "a*TeamSi.e max number of players per side (BotTeamGame, BotCTFGame, BotDoubleDomination)
    - GamePaused boolean, true if the game is currently paused
    - BotsPaused boolean, true if just the bots are paused (if GamePaused$ true, everyone will be paused even if this is set to true or false)
- PA/SE sent when the or the bots are paused.
- PR- incoming projectile likely to hit you. May give you a chance to dodge.
    - !elocity holding the projectile current velocity vector

    - S(eed holding the projectile current speed
    - Location holding the projectile current location
    - Time estimated time till impact
    - irection base vector representing the direction of the projectile. Best chance to dodge is to probably count perpendicular vector to this projecting it to flat space.
    - 'rigin the location the projectile is flying from
    - amageRadius if the projectile has splash damage, how big it is – in ut units
    - Ty(e 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.

- R&H 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
- RE&EN sent as a response to STOPREC command.

- RE&START sent as a response to REC command.
- RES/"E 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
    - !elocity 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
    -)ea(on what weapon the character is holding
    - Firing 0 means is not firing, 1 - firing, 2 - alt firing
- SP) 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.
- TEA"&HANGE response of the CHANGETEAM command.
    - Success boolean. If true team change was succesfull (it won't be succesfull if we are changing to a team we already are in).
    - esiredTeam integer. This is the team we wanted to change to (0 for red,1 for blue, etc..).
- THR')N send if THROW command ends successfully (bot will drop a weapon).
- TR& 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
- TRG trigger message. When we trigger some trigger.
    - Actor should be the trigger.
    - E%entInstigator should be the id of player, who triggered this.
- !&H some part of the bot body changed the zone (volume changed).

    - Id unique id of zone entered
    - Pain&ausing true or false if we get some damage when we stay at this zone
- !"S 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
- !"T 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
-)AL 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
- 0&B bot changed zones. Entire bot now in new zone.
    - Id unique id of zone entered

Batch in#o messages

Na%(oint in#o

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

- SNA! start of navpoint message block.
- INA! info about one navpoint.
    - Id a unique id for this pathnode, assigned by the game
    - Location absolute location of the Na%Point
    - 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.
    - RoamingS(ot boolean. Some ambush point, where is good chance to intercept approaching opponents.
    - Sni(ingS(ot boolean. Point good for sniping.
    - Pre#ered)ea(on Class of the weapon that should be prefered when using this point for AIMarker specified action.
    - Neigh1number2 (Neigh0, Neight1, etc…) here are information about one Na%Point reachable from current Na%Point. One Na%Point 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 Na%Point
    - 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
    - &ollisionR how wide can we be to pass this path
    - &ollisionH how big can we be to pass this path
    - Force ouble-um(boolean. True if the bot needs to perform double jump to get here.
    - Needed-um(vector. Holds how big jump vector needs to be produced in order to get to the point.
- ENA! end of navpoint message block.

In%entory in#o

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.

- SIN! start of inventory message block.
- IIN! info about one inventory item.
  - Id a unique id for this item, assigned by the game
  - Location absolute location
  - &lass a string representing type of object
- EIN! end of inventory message block.

Bot con#igure commands

- A IN! adds inventory of supported class to the bot (players are not supported). 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 GB A IN! classes e*am(les.
  - &lass string of the name of the class. `Example: ADDINV {Class xWeapons.FlakCannonPickup}`
- A RA3 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.
  - irection 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
- &HANGETEA" command for changing the bot team. Responds with TEAMCHANGE message.
  - Team integer. This is the team we want to change to (0 for red,1 for blue, etc..).
- &HATTR 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)
  - Adrenaline sets the bot adrenaline, can range from 0 to 100. The bot has to be alive.
- &HE&,REA&H 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
- &'NF this command configures features of the bot
  - AutoTrace enables AutoTrace$ feature
  - "anualS(a+n 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
  - In%ulnerable 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)
  - !isionTime between 0.1 to 2 seconds, it sets how long should be GB idle before running next checkvision test
  - Sho+ ebug boolean, if true some additional debug information will be logged to server window
  - Sho+FocalPoint boolean, if set to true an actor will appear in the game on the location the bot is actually looking at
  - ra+TraceLines 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
  - Synchronous'## boolean. It enables/disables sending of all GB synchronous messages.
- FTRA&E 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
- GETINFO in response to this command, server will send you the IINV info batch message.
- GETNAVINFO 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
- GIVEINFO 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$, team is meaningless
    - "anualSpawn sets if the bot will be respawned after death manually by RESPAWN command or automatically
    - AutoTrace enables auto tracing (it can be enabled later by CONF command)
    - Location specify start location, if unspecified, then random
    - Rotation specify start rotation, if unspecified, then random
    - Skin sets the bot current skin, for more information look at SETSKIN command
    - esiredSkill Can range from 0 to 7 - it is float, but you should use just integers (1,2,...). This sets the bot accuracy. 1 lowest, 7 highest. Shouldn't have any other effect.
    - esiredAccuracy float from 0 to 1. This should tweak bot accuracy a little. So far it had very little effect on bot accuracy. When you want to change accuracy significantly use DesiredSkill$ 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 INFO 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
- 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. Start GB server from console, otherwise command might not work 100%.
    - Skin string, which specifies the bot appearance. You need to supply package and skin (mesh) name. Examples here: GB SETSKIN E*ample
- 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

Bot action commands

- &HANGE) EAP ' N 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
- & " ' ! E the bot will continuously move straight ahead according to his actual rotation.
  - S (eed sets the speed of the movement. Ranges from 0.1 to 2. This number multiplies default speed of the bot
- & ' " B ' will perform one of the UT combos - can be used when adrenaline is at 100. Will start to drain adrenaline. When at 0 the combo stops.
  - Ty (e name of the combo class you wish the bot to perform (can be "xGame.ComboBerserk", "xGame.ComboDefensive", "xGame.ComboInvis" or "xGame.ComboSpeed" - without the "")
- ' GE causes bot to dodge.
  - irection - vector, will be normalized. The direction of the dodge.
- - / " P causes bot to jump.
  - ouble-um (boolean, if true the bot will perform double jump
  - -um (0 float. If we want to jump by an exact force. Can't be higher then double jump limit (680 units) - if higher then set to this limit. If "JumpZ" provided, "DoubleJump" attribute will be ignored.
- " ESSAGE send a message to the world or just your team. This will likely have some restrictions placed on it soon.
  - Te * t string to send
  - Global if True it is sent to everyone. Otherwise (or if not specified), just your team
- " ' ! E the bot will continuously move first to Location1 and then to Location2, movement between locations will be continuous without any delay.
  - S (eed sets the speed of the movement. Ranges from 0.1 to 2. This number multiplies default speed of the bot
  - Location 6 vector of the first location
  - Location 7 vector of the second location
- R ' TATE turn a specified amount.
  - Amount amount in UT units to rotate. May be negative to rotate counter clockwise
  - A * is if provided as Vertical, rotation will be done to Pitch. Any other value, or not provided, and rotation will be to Yaw
- R / NT ' 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.
  - S (eed 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")
- SET & R ' / & H will crouch/uncrouch the bot.
  - & rouch if true the bot will crouch, if false the bot will uncrouch, after respawn this will be reset - bot will uncrouch
- SET) AL , set whether you are walking or running (default is run).
  -) al 4 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
- SH ' ' T start firing your weapon. If send without parameters, or if Target not visible and location not supported, bot will shoot at his focal point (if the focal point will change, the direction of shooting change too). If location specified, the bot will shoot at location even if his focal point will be changed (note, that bot cannot shoot backwards, if you will turn too much, you wouldn't be able to hit the location). If target

specified and visible, bot will follow the target and shoot (again cannot shoot backwards).
    - Location location you want to shoot at. Normal rules for a location specification
    - Target the unique id of your target.
    - 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

&ontrolSer%er 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.

&ontrolSer%er 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}
- &HANGE"AP will change map to MapName$ - map must exist on server (wont be tested), will send MAPCHANGE message.
    - "aName name of the new map
- &HANGETEA" command for changing the bot team. Responds with TEAMCHANGE message.
    - Id string. The id of the target bot.
    - Team integer. This is the team we want to change to (0 for red,1 for blue, etc..).
- &HATTR 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
- &'NF 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)

    - "anualS(a+n 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
    - In%ulnerable boolean, if true the bot cant be killed. This can be changed just when cheating is enabled on the server (bAllowCheats = True)
    - !isionTime between 0.1 to 2 seconds, it sets how long should be GB idle before running next checkvision test
    - Sho+ ebug boolean, if true some additional debug information will be logged to server window
    - Sho+FocalPoint boolean, if set to true an actor will appear in the game on the location the bot is actually looking at
    - ra+TraceLines 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
    - Synchronous'## boolean. It enables/disables sending of all GB synchronous messages
- &'NFGA"E Used for configuring the game variables.
    - amage"odi#ier float. Range from 0 (excluded) to 10. Each damage is multiplied by this before applied to bots (numbers are rounded). This attribute can be also set in the ini file. (In BotAPI‡Bot eath"atch part).
    - "a*S4ill float. Range from 0 (excluded) to 7. Set maximum available skill for remote and epic bots. Won't change the skill of bots already in the game. (Can be set in ini file in BotAPI‡Bot eath"atch).
- &'NS'LE You can run all console command by this.
    - &ommand string. The exact console command (as you would type to console).
- EN PLRS Will stop exporting of IPLR messages synchronously.
- GET"APS Will request map list from the server. Server will respond with IMAP batch map info message.
- GETNA!S 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.
- ,I&, will kick RemoteBot$ from the server.
    - Id of the bot to be kicked
- PASS)'R Send password to the server. For more information see ControlServer$ command SETPASS.
    - Pass+ord string. Holds the password.
- PA/SE 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".
- 5/IT will close the connection.
- REA 3 command. Response to HELLO message. The server will send standard game NFO message.
- RE& 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
- RESPA)N 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
- SETGA"ESPEE will set speed of the game.
    - S(eed 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.
- SETL'&, Will disable new connections to botserver and or control server - depends on parameters. If last ControlServer$ instance is leaving. ControlServer$ lock will be canceled.
    - BotSer%er boolean. If BotConnections$ should be locked.
    - &ontrolSer%er 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

    - Pass+ord string. Sets the password.
- SPA)NA&T'R This command will try to spawn an actor of desired UT class specified by Type attribute at the supported location with supported rotation. Note that if the object will not fit in the area where you want to spawn it, it may not be spawned. Use this command with caution - spawning too many actors can lead to UT engine being slowed down considerably. Also there is no way how to delete previously spawned actors from the game at this moment. To summarize this command of GameBots is in debug stage.
    - Ty(e string. UT class specifying the item that should be spawned. ADDINV examples will work here: GB A IN! classes e*am(les
    - Location location, where the object should be spawned.
    - Rotation starting rotation of the object.
- 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.
    - /nrealBots boolean. All UnrealBots$ will be exported.
- ST'PRE& will stop recording a demo. Is confirmed by RECEND message.

&ontrolSer%er messages

- ALI!E 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.
- &'NF&H 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
    - "anualS(a+n 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
    - In%ulnerable boolean, if true the bot cant be killed. This can be changed just when cheating is enabled on the server (bAllowCheats = True)
    - !isionTime float, ranges from 0.1 to 2 seconds. This will change the period between two synchronous batches
    - Sho+ ebug boolean, if true some additional debug information will be logged to server window
    - Sho+FocalPoint boolean, if set to true an actor will appear in the game on the location the bot is actually looking at
    - ra+TraceLines 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
    - Synchronous'## 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).
- HELL'&'NTR'L SER!ER message, sent immediately after socket establish.
- I"AP 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
    - "anualS(a+n true or false – if the bot is not respawning automatically
    - In%ulnerable true or false – if true, bot cannot die
    - !isionTime float, delay between two synchronous batches
    - Sho+ ebug boolean, if true additional debug is displayed from this bot in server window (the console that was used to start the server)
    - Sho+FocalPoint boolean, if true the bots focal point is displayed in the game
    - ra+TraceLines boolean, if true the automatic ray tracing lines (ATR messages) are displayed in the game. Has issues, does not work on some UT2004 copies
- -'IN 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
- "AP&HANGE sent when the map is changed (server will lost the connection).
    - "a(Name text name of the map
- PA/SE sent when the or the bots are paused.
- RE&EN response to STOPREC command.
- RE&START response to REC command.
- RES/"E sent when the game and the bots are unpaused.
- TEA"&HANGE response of the CHANGETEAM command.
    - Id string. Id of the bot.
    - Success boolean. If true team change was succesfull (it won't be succesfull if we are changing to a team we already are in).
    - esiredTeam integer. This is the team we wanted to change to (0 for red,1 for blue, etc..).

Created by: michal.bida. Last Modification: Thursday 18 of February, 2010 12:00:01 CET by michal.bida.
The original document is available at htt(s9::artemis8ms8m##8cuni8c.:(ogamut:ti4i;
inde*8(h($(age<Gamebots = 7>API = 7>list