BFBC2 PC Server Administration

This is the server admin manual for BFBC2 PC Server R13.

Contents

Game server operation

When the game server first starts up, it reads a set of configuration files from disk. Some of these are managed by the RSP, and some by the server administrator.

The game server will then cycle through a series of maps. Game clients can connect to the server and play on the maps.

Control of the game server is done through a “Remote Administration” interface. This is a TCP port (kind of like a terminal interface). There are both Python scripts and GUI tools which control the game server through this mechanism.

Players can indirectly communicate with the GUI tools by sending special chat commands, which the GUI tools react upon.

The game server writes a set of log files to disk while it is running; these can be inspected by the server admin.

Files which may be accessible to the server admin

EA decides which files the RSP may make available to the admin. It is up to each RSP how to facilitate this, and the extent to which access is given. The list below contains all the files which the RSP is allowed to give the server admin full access to:

AdminScripts/*.txt
AdminLogs/*.log
pb/svss/*
pb/svlogs/*
pb/*.cfg
pb/pbbans.dat
pb/pbucon.use
pb/sv_viol.log
pb/sv_cheat.log
banlist.txt
maplist.txt
reservedslotslist.txt

In addition, the server admin may have limited access to a few lines in ServerOptions.ini.

Remote administration interface

The remote administration interface is a two-way channel for sending and receiving commands from the game server. Before the remote administration can be used, a remote admin password must be set, either via AdminScripts/startup.txt or ServerOptions.ini.

Do notice that the remote admin interface is normally case sensitive.

Commands

There's a Python script, called "CommandConsole.py", that can be used to connect to the remote administration interface. Once connected, there is an assortment of commands available that can be sent. See "BFBC2 PC Remote Administration Protocol.pdf" for the full list.

Events

The game server can also send events when specific things happen in-game. For instance, when a player joins or leaves the server, when a round ends, or when anyone says anything through the chat. The Python script called "EventConsole.py" can be used to listen to these events.

GUI tools

There are several GUI tools constructed, which make it easier to control the game server. We'd recommend that you use them rather than relying on CommandConsole.py / EventConsole.py for everyday use.

Startup script

The game server will process the file named AdminScripts/startup.txt during bootup. Each line in that file will be executed as a remote administration command. If there are errors, they will be detailed in the status log.

Log files

Admin log

Files named AdminLogs/AdminLog*.log log every command & event that the remote administration interface performs.

Status log

Any files named AdminLogs/StatusLog*.log describe what the server is doing. If your server is acting strangely, or perhaps ignoring some of your configuration options, look in the latest log file – you might find some hints there.

Accounts, soldier names, and GUIDs

Every BFBC2 PC player has exactly one EA account. The player can then have one or more soldier names.

The PunkBuster GUID is tied to the EA account. So is the "EA GUID".

The PB GUID is used with all PB-services, while the EA GUID is used with any non-PB-related functions in the game server.

PunkBuster GUIDs are 32-digit hexstrings.

EA GUIDs are the prefix "EA_" followed by a 32-digit hexstring.

Player slots

How many players does a server support?

This is determined by 3 factors:

- The current gamemode.
- The RSP has a max-cap which they can set per server
- Finally, the admin also has a max cap that can be set (vars.maxPlayerLimit)

The current max number of players is the minimum of the three, however never less than 8.

The current effective value can be queried through vars.currentPlayerLimit.

VIP players, reserved slots, and the join queue

Older versions of the BFBC2 PC servers had a system where slots could be reserved permanently. This has been abandoned in favor of a kick-on-demand system (similar to that of BFBC2). In addition, there is a join queue.

Defining VIP players

The server has an internal list of VIP players. Upon startup, the server reads it from a file named reservedslotslist.txt. The list of VIP players can also be edited on-the-fly through the use of the **reservedSlots.*** commands.

The list of VIP players can contain up to 500 entries.

When a non-VIP player joins a server…

If there are slots left, the player will be able to join directly.

Otherwise, the player is placed in a join queue. ("You are person 3 out of 3 in the queue for this server.")

When a VIP player joins a server…

If there are slots left, the player will be able to join directly – just like a non-VIP player.

If there are no slots left, the server will randomly choose a non-VIP player, and kick him/her to make room for the VIP player.

If the server happens to be full of other VIP players, then the VIP player will be placed in a join queue.

reservedslotslist.txt format

Each line in the file lists a soldier name.

Ban system

The game server has an internal ban system. This system is independent from PunkBuster's banlist. At startup, ban entries are read from the file named banlist.txt. During runtime, the **banList.*** commands can be used to manipulate the banlist.

Players can be banned either on their soldier name, or on their EA GUID. Banning someone on their soldier name is not particularly effective – if it's a determined griefer then he/she will just create a new soldier and return . Banning someone on their EA GUID is much more effective.

To find out someone's EA GUID, perform admin.serverInfo while that person is playing on your server. Or – inspect the AdminLog.

People can be banned either for a few seconds, until the end of the current round, or permanently.

The banlist can contain up to 10.000 entries.

banlist.txt format

Each entry in the banlist occupies 5 lines.

The first line specifies what the ban is on:

guid – ban on EA GUID
name – ban on soldier name
ip – ban on game client IP address

The second line specifies the GUID/name/IP that the ban applies to

The third line specifies the duration of the ban:

perm – permanent
round – until the end of the current round
seconds – until the given time is reached

The fourth line contains the timestamp for a "seconds"-type ban; otherwise it is unused.

The fifth line contains the reason for being banned. Max length 80 characters.

Communicating with other players

Sending messages from the admin interface

The **admin.yell** command can be used to display across players' screens. It looks quite obnoxious so use it sparingly.

The **admin.say** command sends regular chat messages to players.

Profanity filter

The game server supports a profanity filter. It filters chat messages that are sent from game clients. It is on by default. You can enable/disable it through the **vars.profanityFilter** command.

Messages sent via **admin.say** or **admin.yell** do not get profanity filtered.

Map handling

Overview

BFBC2 PC maps are divided into four different gamemodes. The game server can have one gamemode active at a time. It also has a list of maps. The maplist determines in which sequence the maps will be played, and how many rounds will be played on each map.

Upon startup, the maplist.txt file is read. During runtime, the **mapList.*** commands can be used to edit the set of maps.

When the same map is played for several rounds, all 2-team gamemodes stipulate that the teams will switch sides after a run. This way, a 2-round session of Rush will have players play both attackers and defenders.

Controlling map switching

mapList.* can be used to edit the maplist while the server is running.

mapList.nextLevelIndex controls which position in the maplist that the server currently is at.

admin.runNextRound switchies to the next round, without finishing the current.

admin.restartRound makes all players reload the current map, and restarts the current round.

admin.endRound declares a specific team as the winning team, and then moves directly to the end-of-round screen.

maplist.txt format

The first line in the file identifies the gamemode.
Subsequent lines specify map names, and optionally the number of rounds that the map should be played. Default number of rounds is 2.

Autobalance

There is an autobalancer built into the game. It is active for Conquest and Rush gamemodes. It can be enabled/disabled through **vars.teamBalance**.

The autobalancer waits until there difference between number of players in both teams is 2 or greater.

After 20 seconds, a warning is sent.

After 40 seconds, players not in squads will be moved.

After 60 seconds, full squads will be moved.

After 150 seconds, squads will be forcefully split to even out the teams.

Team-kill kicker

When friendly fire is enabled, there is a system which can automatically kick players for excessive teamkilling. It tracks how many kills (in total) each player has performed during the current round. Also, it has a scoring system, where each kill increases the score by a set amount, and the score decreases at a constant rate.

If either the kill count or the kill score goes above ser thresholds, the player is kicked for teamkilling.

You can control the settings of this functionality using **vars.teamKill***.

Idle timeout

If a player doesn't give any input within a specific period of time, he/she will be kicked due to idling. You can change the time interval / disable the idle timeout through **vars.idleTimeout**.

Ranked and unranked servers

Servers can run in two modes: ranked, and unranked.

When servers run in ranked mode, player progression (scoring etc) is reported to EA's master servers. Players will gain unlocks etc through continued play on ranked servers.

On the other hand, there are more rules on ranked servers than on unranked. Some server functions (such as setting a game password) are unavailable when running a ranked server.

Ranked servers must also run PunkBuster.

The Battlefield Rules of Conduct apply to all ranked servers.

Unranked servers do not allow people any progression or stats tracking, but give more leeway as to how the game is configured.

Reconfiguring the game modes

This is usually done through two sets of console commands:

One which changes level-independent settings, and another which changes level-dependent settings.

Normal, modified, and hardcore game modes

If all settings on the game server are in their default state, the server runs in "Normal" mode. Some settings can be changed while the server remtains in "Normal" mode.

Changing some settings will convert the server to "Modified" mode. See info below.

Changing the Hardcore setting will convert the server to "Hardcore" mode.

Normal/Modified/Hardcore, and the server browser

Players can see whether a server is normal / modified / hardcore in the server browser. A grey skull icon indicates a "Modified" mode. A yellow skull indicates a "Hardcore" mode.

If the player chooses to filter for hardcore OFF, only Normal servers are retrieved.

If the player chooses not to filter for hardcore, all servers are retrieved.

If the player chooses to filter for hardcore ON, only Hardcore servers are retrieved.

Normal/Modified/Hardcore, and Play Now

Play Now is mainly used by casual players. As such, it will only send players to Normal servers. In addition, it will attempt to send players to servers which are not full, not empty, and in the same part of the world. Plus, it will attempt to pair up players with other players of equal skill.

Level-independent settings

The following level-independent settings are available:

vars.serverName controls the name of the server, as seen in server browser

vars.adminPassword controls password used for login to remote admin interface

vars.gamePassword - if set, players must enter a password when connecting to server

vars.hardCore – when set, player health is reduced, most in-game UI is disabled, friendly fire is enabled and other things.
Setting this makes the server "Hardcore".

vars.ranked - when set, stats are reported to EA's master servers.

Currently, this is set via ServerOptions.ini and read-only while the server is running.

When set, vars.gamePassword is forced off and vars.punkBuster is forced on.

vars.rankLimit – when set, players with a higher rank than the specified value are not allowed into the server.

vars.teamBalance – when set, the autobalance system (described elsewhere in this document) is active.

vars.friendlyFire – when set, people can inflict damage on others in the same team.
Setting this makes the server "Modified".

vars.killCam – when set, a killed player gets to see a close-up of his/hers killer for a few seconds.
Disabling killCam makes the server "Modified".

vars.miniMap – when set, a minimap is available in the bottom-left corner of the screen during play.
Disabling minimap makes the server "Modified".

vars.crossHair – when set, guns have crosshairs in the center of the screen.
Disabling crosshairs makes the server "Modified".

vars.3dSpotting – when set, spotted targets are marked with icons in the 3D world.
Disabling this makes the server "Modified".

vars.miniMapSpotting - when set, spotted targets are marked with icons on the minimap.
Disabling this makes the server "Modified".

vars.thirdPersonVehicleCameras - when set, 3rd person vehicle cameras are enabled.
Disabling this makes the server "Modified".

Level-specific settings

When the server loads a new level, it looks through a pool of level-specific setting. If it finds anything that applies, then its value is used to override the default values for the level.

Contexts

Level-specific settings come as three different contexts: either they apply to all levels, or only levels within one gamemode, or one specific level.

When the server loads a new level, it will prefer settings for a narrower context; i.e. if there is an overall setting, a setting for the current gamemode, and a setting for the current level, then the setting for the current gamemode will be used.

Changing settings

levelVars.set is used to set a level-specific setting. For instance, to double the ticket count for all CONQUEST levels, you would do 'levelVars.set gamemode CONQUEST tickets 200'.

levelVars.get reads the value of a specific setting.

levelVars.evaluate determines what value of a current setting applies to the current level; useful when there are multiple contexts that apply to the current level and it's difficult to tell which has highest priority.

List of implemented settings

tickets <percent> (default: 100)
This is a scale-factor for the original number of tickets on a level
for Conquest, number of tickets for both teams
for Rush and Squadrush, number of tickets for the attacking team
for Squad Deathmatch, winning ticketcount

ticketBleedSpeed <percent> (default: 100)
This is a scale-factor for the original bleed-speed for a level
specifies how quickly tickets should bleed when one side has dominance

vehicleSpawnRate <percent> (default: 100)
This is a scale-factor for the original vehicle spawn rate on a level
specifies how quickly vehicles should respawn; this does not affect other caps that are built into the levels, such as "Max X vehicles of type Y on this level"

vehiclesDisabled <true|false> (default: false)
When disabled, vehicles and stationary anti-vehicle weapons are removed from the level.
Disabling vehicles makes the server "Modified".

startDelay <seconds> (default varies from gamemode to gamemode)
pecifies how long the spawn-delay will be once the first player has loaded in to a map

respawnDelay <seconds> (default: 15-20 somewhere)
when a player is killed, it takes this long until he/she can spawn in again.
Minimum allowed value is 10 seconds.