

BFBC2 PC Remote Administration Protocol

This is the remote-administration protocol used by BFBC2 PC Server R9 prerelease.

It is work-in-progress; features are first added to the game, and then controlling commands are added to the Remote Administration interface.

Contents

About.....	2
Low-level protocol	2
Packet format.....	2
int32	2
Word	2
Packet.....	2
Protocol behaviour.....	3
Comments	3
Parameter formats.....	4
String	4
Boolean	4
HexString.....	4
Password	4
Filename.....	4
Clantag	4
Player name	4
Team ID	4
Squad ID	4
Player subset.....	5
Timeout.....	5
Id-type	5
Player info block.....	5
Server events	6
Summary	6
Player events.....	6
Misc.....	7
Client commands	8
Summary	8
Misc.....	9
Query.....	11

Communication.....	11
Level	11
Kick/List players	13
Banning	13
Reserved slots	14
Maplist	15
Variables.....	17

About

This document describes how to communicate with the Remote Administration interface that is present in BFBC2 PC servers. The protocol is bidirectional, and allows clients to send commands to the server as well as the server to send events to clients.

The protocol is designed for machine-readability, not human-readability. It is the basis for all graphical remote administration tools.

Low-level protocol

Packet format

int32

32-bit unsigned integer

1 byte	bits 7..0 of value
1 byte	bits 15..8 of value
1 byte	bits 23..16 of value
1 byte	bits 31..24 of value

Word

int32	Size	Number of bytes in word, excluding trailing null byte
char[]	Content	Word contents -- must not contain any null bytes
char	Terminator	Trailing null byte

Packet

int32	Sequence	Bit 31: 0 = The command in this command/response pair originated on the server 1 = The command in this command/response pair originated on the client Bit 30: 0 = Request, 1 = Response Bits 29..0: Sequence number (this is used to match requests/responses in a full duplex transmission)
int32	Size	Total size of packet, in bytes
int32	NumWords	Number of words following the packet header

Word[N] Words N words

A packet cannot be more than 16384 bytes in size.

Protocol behaviour

The client communicates with the server using a request/response protocol. Each request contains a sequence number which grows monotonically, a flag which indicates whether the command originated on the client or the server, and one word containing the command name. In addition to this, a command can have zero or more arguments.

Every request must be acknowledged by a response. The response includes the the same sequence number, and the same origin flag. However, it has the response flag set.

Sequence numbers are unique within one server-client connection. Thus, the same sequence number can be used when the server is communicating with different clients.

Responses must contain at least one word. The first word can be one of the following:

OK	- request completed successfully
UnknownCommand	- unknown command
InvalidArguments	- Arguments not appropriate for command
<other>	- command-specific error

OK is the only response which signifies success.
Subsequent arguments (if any) are command-specific.

The server is guaranteed to adhere to this protocol specification. If the client violates the protocol, the server may close the connection without any prior notice.

Comments

The format of the Words portion of a packet is designed such that it shall be easy to split it into individual words in both C++ and Python. Any numerical arguments are always transferred in string form (not in raw binary form).

The protocol is designed to be fully bidirectional.

Parameter formats

String

An 8bit ASCII string. Must not contain any characters with ASCII code 0.

Boolean

Two possible values:

true
false

HexString

A stream of hexadecimal digits. The stream must always contain an even number of digits. Allowed characters are:
0123456789ABCDEF

Password

A password is from 0 up to 16 characters in length, inclusive. The allowed characters are:
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789

Filename

A filename is from 1 up to 240 characters in length, inclusive. The allowed characters are:
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789._-

Clantag

A clan tag is from 0 to an unknown number of characters in length. At the time of writing, it is unclear which the allowed characters are.

Player name

The “player name” (referred to as “Soldier name” in-game) is the persona name which the player chose when logging in to EA Online. One EA Account can have multiple personas.

A player has a name from 4 to 16 characters in length, inclusive. The allowed characters are:

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
_ - & () * + . / : ; < = > ? [] ^ { | } ~ <space>

When a player is creating a new persona, it is compared against all other persona names; the new name must be unique. The following characters are ignored during the comparison:

- _ <space>

Team ID

An integer.

Team 0 is neutral. Depending on gamemode, there are up to 16 non-neutral teams, numbered 1..16.

Squad ID

An integer.

Squad 0 is “no squad”. Depending on gamemode, there are up to 16 squads numbered 1..16.
Note that squad ID are local within each team; that is, to uniquely identify a squad you need to specify both a Team ID and a Squad ID.

Player subset

Several commands – such as `admin.listPlayers` – take a player subset as argument.

A player subset is one of the following:

<code>all</code>	- all players on the server
<code>team <team number: integer></code>	- all players in the specified team
<code>squad <team number: integer> <squad number: integer></code>	- all players in the specified team+squad
<code>player <player name: string></code>	- one specific player

Timeout

Some commands, such as bans, take a timeout as argument.

A timeout is one of the following:

<code>perm</code>	- permanent
<code>round</code>	- until end of round
<code>seconds <number of seconds: integer></code>	- number of seconds

Id-type

Some commands, such as bans, take an id-type as argument

An id-type is one of the following:

<code>name</code>	- Soldier name
<code>ip</code>	- IP address
<code>guid</code>	- Player guid

Player info block

The standard set of info for a group of players contains a lot of different fields. To reduce the risk of having to do backwards-incompatible changes to the protocol, the player info block includes some formatting information.

<code><number of parameters></code>	- number of parameters for each player
<code>N x <parameter type: string></code>	- the parameter types that will be sent below
<code><number of players></code>	- number of players following
<code>M x N x <parameter value></code>	- all parameter values for player 0, then all parameter values for player 1, etc

Current parameters:

<code>name</code>	<code>string</code>	- player name
<code>guid</code>	<code>GUID</code>	- player GUID, or "" if GUID is not yet known
<code>teamId</code>	<code>Team ID</code>	- player’s current team
<code>squadId</code>	<code>Squad ID</code>	- player’s current squad
<code>kills</code>	<code>integer</code>	- number of kills, as shown in the in-game scoreboard
<code>deaths</code>	<code>integer</code>	- number of deaths, as shown in the in-game scoreboard
<code>score</code>	<code>integer</code>	- score, as shown in the in-game scoreboard
<code>ping</code>	<code>integer</code>	- ping (ms), as shown in the in-game scoreboard

Server events

Most commands require the client to be logged in. Before the client has logged in, only 'login.plainText', 'login.hash', 'logout', 'version', 'serverInfo' and 'quit' commands are available.

Summary

Command	Description
player.onJoin	Player with name <soldier name> has joined the server
player.onAuthenticated	Player with name <soldier name> has been authenticated + got GUID
player.onLeave	with name <soldier name> has left the server
player.onKill	Player with name <killing soldier name> has killed <killed soldier name>
player.onChat	Chat message has been sent to a group of people
player.onKicked	Player with name <soldier name> has been kicked
punkBuster.onMessage	PunkBuster server has output a message
server.onLoadingLevel	Level is loading
server.onLevelStarted	Level is started
player.onSquadChange	Player might have changed squad
player.onTeamChange	Player might have changed team

Player events

Request: player.onJoin <soldier name: string>
Response: OK
Effect: Player with name <soldier name> has joined the server

Request: player.onAuthenticated <soldier name: string> <player GUID: guid>
Response: OK
Effect: Player with name <soldier name> has been authenticated, and has the given GUID

Request: player.onLeave <soldier name: string>
Response: OK
Effect: Player with name <soldier name> has left the server

Request: player.onKill <killing soldier name: string> <killed soldier name: string>
Response: OK
Effect: Player with name <killing soldier name> has killed <killed soldier name>
##RSP Comment: onKill does not specify the weapon used to kill you opponent. This would be really handle to monitor our ranked servers and immediately identify if there is anything suspicious (stat-padding) going on

Request: player.onChat <source soldier name: string> <text: string> <target group: player subset>
Response: OK
Effect: Player with name <source soldier name> (or the server, or the server admin) has sent chat message <text> to some people

Comment: The chat text is as represented before the profanity filtering
If <source soldier name> is "Server", then the message was sent from the server rather than from an actual player
If sending to a specific player, and the player doesn't exist, then the target group will be "player" ""

Request: player.onKicked <soldier name: string> <reason: string>

Response: OK

Effect: Player with name <soldier name> has been kicked

Request: player.onSquadChange <soldier name: player name> <team: Team ID> <squad: Squad ID>

Response: OK

Effect: Player might have changed squad

Request: player.onTeamChange <soldier name: player name> <team: Team ID> <squad: Squad ID>

Response: OK

Effect: Player might have changed team

Misc

Request: punkBuster.onMessage <message: string>

Response: OK

Effect: PunkBuster server has output a message

Comment: The entire message is sent as a raw string. It may contain newlines and whatnot.

Request: server.onLoadingLevel <level name: string>

Response: OK

Effect: Level is loading

Request: server.onLevelStarted

Response: OK

Effect: Level is started

Client commands

Most commands require the client to be logged in. Before the client has logged in, only 'login.plainText', 'login.hash', 'logout', 'version', 'serverInfo', 'listPlayers' and 'quit' commands are available.

Summary

Command	Description
login.plainText <password>	Attempt to login to game server with password
login.hash	Retrieves the salt, used in the hashed password login process
login.hash <passwordHash>	Sends a hashed password to the server, in an attempt to log in
logout	Logout from game server
quit	Disconnect from server
version	Reports game server type, and build ID
listPlayers <players>	Return list of a group of players on the server, without GUIDs
eventsEnabled <enabled>	Set whether or not the server will send events to the current connection
help	Report which commands the server knows about
admin.runscript <filename>	Process file, runs script lines one-by-one, aborting processing upon error
punkBuster.pb_sv_command <command>	Send a raw PunkBuster command to the PunkBuster server
serverinfo	Query for brief server info
admin.yell <message, duration, players>	Display a message, very visibly on players' screens
admin.say <message, players>	Send a chat message to a group of players
admin.runNextLevel	Switch to next level
admin.currentLevel	Return current level name
mapList.nextLevelIndex	Get index of next level to be run
mapList.nextLevelIndex <index>	Set index of next level to be run
admin.restartMap	End current round, and restart with the same map
admin.supportedMaps <play list>	Retrieve maplist of maps supported in this play list
admin.setPlaylist <name>	Set the play list on the server
admin.getPlaylist	Get the current play list for the server
admin.getPlaylists	Get the play lists for the server
admin.kickPlayer <soldier name, reason>	Kick player <soldier name> from server
admin.listPlayers <players>	Return list of a group of players on the server
banList.load	Load list of banned players/IPs/GUIDs from file
banList.save	Save list of banned players/IPs/GUIDs to file
banList.add <id-type, id, timeout, reason>	Add player/IP/GUID to ban list for a certain amount of time
banList.remove <id-type, id>	Remove player/IP/GUID from ban list
banList.clear	Clears ban list
banList.list	Return list of banned players/IPs/GUIDs
reservedSlots.load	Load list of reserved soldier names from file
reservedSlots.save	Save list of reserved soldier names to file
reservedSlots.addPlayer <name>	Add <name> to list of players who can use the reserved slots
reservedSlots.removePlayer <name>	Remove <name> from list of players who can use the reserved slots
reservedSlots.clear	Clear reserved slots list
reservedSlots.list	Retrieve list of players who can utilize the reserved slots
mapList.load	Load list of map names from file
mapList.save	Save maplist to file
mapList.list	Retrieve current maplist
mapList.clear	Clears maplist
mapList.remove <index>	Remove map from list

mapList.append <name>	Add map with name <name> to end of maplist
mapList.insert <index, name>	Add map with name at the specified index to the maplist
vars.adminPassword <password>	Set the admin password for the server
vars.gamePassword <password>	Set the game password for the server
vars.punkBuster <enabled>	Set if the server will use PunkBuster or not
vars.hardCore	Set hardcore mode
vars.ranked	Set ranked or not
vars.rankLimit <rank>	Set the highest rank allowed on to the server
vars.teamBalance <enabled>	Set if the server should autobalance
vars.friendlyFire <enabled>	Set if the server should allow team damage
vars.currentPlayerLimit	Retrieve the current maximum number of players
vars.maxPlayerLimit	Retrieve the server-enforced maximum number of players
vars.playerLimit <nr of players>	Set desired maximum number of players
vars.bannerUrl <url>	Set banner url
vars.serverDescription <description>	Set server description
vars.killCam <enabled>	Set if killcam is enabled
vars.miniMap <enabled>	Set if minimap is enabled
vars.crossHair <enabled>	Set if crosshair for all weapons is enabled
vars.3dSpotting <enabled>	Set if spotted targets are visible in the 3d-world
vars.miniMapSpotting <enabled>	Set if spotted targets are visible on the minimap
vars.thirdPersonVehicleCameras <enabled>	ToDo

Misc

Request: login.plainText <password: string>
Response: OK - Login successful, you are now logged in regardless of prior status
Response: InvalidPassword - Login unsuccessful, logged-in status unchanged
Response: PasswordNotSet - Login unsuccessful, logged-in status unchanged
Response: InvalidArguments
Effect: Attempt to login to game server with password <password>
Comments: If you are connecting to the admin interface over the internet, then use login.hashed instead to avoid having evildoers sniff the admin password

Request: login.hashed
Response: OK <salt: HexString> - Retrieved salt for the current connection
Response: PasswordNotSet - No password set for server, login impossible
Response: InvalidArguments
Effect: Retrieves the salt, used in the hashed password login process
Comments: This is step 1 in the 2-step hashed password process. When using this people cannot sniff your admin password.

Request: login.hashed <passwordHash: HexString>
Response: OK - Login successful, you are now logged in regardless of prior status
Response: PasswordNotSet - No password set for server, login impossible
Response: InvalidPasswordHash - Login unsuccessful, logged-in status unchanged
Response: InvalidArguments
Effect: Sends a hashed password to the server, in an attempt to log in
Comments: This is step 2 in the 2-step hashed password process. When using this people cannot sniff your admin password.

password.

Request: logout
Response: OK - You are now logged out regardless of prior status
Response: InvalidArguments
Effect: Logout from game server

Request: quit
Response: OK
Response: InvalidArguments
Effect: Disconnect from server

Request: version
Response: OK BFBC2 <version>
Response: InvalidArguments
Effect: Reports game server type, and build ID
Comments: Game server type and build ID uniquely identify the server, and the protocol it is running.

Request: listPlayers <players: player subset>
Response: OK <player info>
Response: InvalidArguments
Effect: Return list of all players on the server, but with zeroed out GUIDs

Request: eventsEnabled [enabled: boolean]
Response: OK - for set operation
Response: OK <enabled: boolean> - for get operation
Response: InvalidArguments
Effect: Set whether or not the server will send events to the current connection

Request: help
Response: OK <all commands availble on server, as separate words>
Response: InvalidArguments
Effect: Report which commands the server knows about

Request: admin.runScript <filename: filename>
Response: OK
Response: InvalidArguments
Response: InvalidFileName - The filename specified does not follow filename rules
Response: ScriptError <line> <original error...> - Script failed at line <line>, with the given error
Effect: Process file, executing script lines one-by-one, aborting processing upon error

Request: punkBuster.pb_sv_command <command: string>
Response: OK - Command sent to PunkBuster server module
Response: InvalidArguments
Response: InvalidPbServerCommand - Command does not begin with "pb_sv_"
Effect: Send a raw PunkBuster command to the PunkBuster server
Comment: The entire command is to be sent as a single string. Don't split it into multiple words.

Query

Request: serverInfo
Response: OK <serverName> <current playercount> <max playercount> <current gamemode> <current map>
Response: InvalidArguments
Effect: Query for brief server info.
Comments: This command can be performed without being logged in.

Communication

Request: admin.yell <message: string> <duration [in ms]: integer> <players: player subset>
Response: OK
Response: InvalidArguments
Response: TooLongMessage
Response: InvalidDuration
Effect: Display a message, very visibly on players' screens, for a certain amount of time. The duration must be more than 0 and at most 60000 ms. The message must be less than 100 characters long.

Request: admin.say <message: string> <players: player subset>
Response: OK
Response: InvalidArguments
Response: TooLongMessage
Effect: Send a chat message to players. The message must be less than 100 characters long.

Level

Request: admin.runNextLevel
Response: OK
Response: InvalidArguments
Effect: Switch to next level
Comments: Always successful

Request: admin.currentLevel
Response: OK <name>
Response: InvalidArguments
Effect: Return current level name

Request: mapList.nextLevelIndex
Response: OK
Response: InvalidArguments
Effect: Get index of next level to be run

Request: mapList.nextLevelIndex <index: integer>
Response: OK
Response: InvalidArguments
Response: InvalidIndex - Level index not available in server map list
Effect: Set index of next level to be run to <index>

Request: admin.restartMap
Response: OK
Response: InvalidArguments
Response: LevelNotAvailable - server currently has no level loaded / level not available on server
Effect: End current round, and restart with the same map

Request: admin.supportedMaps <play list: string>
Response: OK <map names>
Response: InvalidArguments
Response: InvalidPlaylist <play list> - Play list doesn't exist on server
Effect: Retrieve maplist of maps supported in this play list

Request: admin.setPlaylist <name: string>
Response: OK - Play list was changed
Response: InvalidArguments
Response: InvalidPlaylist - Play list doesn't exist. Should be RUSH, CONQUEST, SQDM or SQRUSH.
Effect: Set the play list on the server.
Comments: Will only use maps supported for this play list. So the mapList might be invalid
Delay: Change occurs after end of round

Request: admin.getPlaylist
Response: OK <play list>
Response: InvalidArguments
Effect: Get the current play list for the server

Request: admin.getPlaylists
Response: OK <play lists>
Response: InvalidArguments
Effect: Get the play lists for the server

Kick/List players

Request: admin.kickPlayer <soldier name: player name, reason: string>
Response: OK - Player did exist, and got kicked
Response: InvalidArguments
Response: PlayerNotFound - Player name doesn't exist on server
Effect: Kick player <soldier name> from server
Comments: Reason text is optional. Default reason is "Kicked by administrator".

Request: admin.listPlayers <players: player subset>
Response: OK <player info>
Response: InvalidArguments
Effect: Return list of all players on the server

Banning

Request: banList.load
Response: OK
Response: InvalidArguments
Response: InvalidIdType
Response: InvalidBanType
Response: InvalidTimeStamp - A time stamp could not be read
Response: IncompleteBan - Incomplete ban entry at end of file
Response: AccessError - Could not read from file
Effect: Load list of banned players/IPs/GUIDs from file
Comment: 5 lines (Id-type, id, ban-type, time and reason) are retrieved for every ban in the list.
Entries read before getting InvalidIdType, InvalidBanType, InvalidTimeStamp and IncompleteBan is still loaded.

Request: banList.save
Response: OK
Response: InvalidArguments
Response: AccessError - Could not save to file
Effect: Save list of banned players/IPs/GUIDs to file
Comment: 5 lines (Id-type, id, ban-type, time and reason) are stored for every ban in the list.
Every line break has windows "\r\n" characters.

Request: banList.add <id-type: id-type> <id: string> <timeout: timeout> <reason: string>
Response: OK
Response: InvalidArguments
Response: BanListFull
Effect: Add player to ban list for a certain amount of time
Comments: Adding a new player/IP/GUID ban will replace any previous ban for that player/IP/GUID
timeout can take three forms:
perm - permanent [default]
round - until end of round

seconds <integer> - number of seconds until ban expires

Id-type can be any of these

name – A soldier name

ip – An IP address

guid – A player guid

Id could be either a soldier name, ip address or guid depending on id-type.

Reason is optional and defaults to “Banned by admin”; max length 80 chars.

The ban list can contain at most 100 entries.

Request: banList.remove <id-type: id-type> <id: string>
Response: OK
Response: InvalidArguments
Response: NotFound - Id not found in banlist; banlist unchanged
Effect: Remove player/ip/guid from banlist

Request: banList.clear
Response: OK
Response: InvalidArguments
Effect: Clears ban list

Request: banList.list
Response: OK <player ban entries>
Response: InvalidArguments
Effect: Return list of banned players/IPs/GUIDs.
Comment: The list starts with a number telling how many bans the list is holding.
After that, 5 words (Id-type, id, ban-type, time and reason) are received for every ban in the list.

Reserved slots

Request: reservedSlots.configFile [filename: filename] - **disabled for security reasons atm**
Response: OK - for set option
Response: OK <filename> - for get option
Response: InvalidArguments
Response: InvalidFileName - Filename does not follow filename rules
Effect: Set name of reserved slots configuration file

Request: reservedSlots.load
Response: OK
Response: InvalidArguments
Response: AccessError - File not found; internal reserved slots list is now empty
Effect: Load list of soldier names from file. This is a file with one soldier name per line.
If loading succeeds, the reserved slots list will get updated.
If loading fails, the reserved slots list will remain unchanged.

Request: reservedSlots.save
Response: OK
Response: InvalidArguments
Response: AccessError - Error while saving
Effect: Save list of reserved soldier names to file. This is a file with one soldier name per line.
Comment: If saving fails, the output file may be unchanged or corrupt.

Request: reservedSlots.addPlayer <soldier name: player name>
Response: OK
Response: InvalidArguments
Response: PlayerAlreadyInList - Player is already in the list; reserved slots list unchanged
Effect: Add <soldier name> to list of players who can use the reserved slots.

Request: reservedSlots.removePlayer <soldier name: player name>
Response: OK
Response: InvalidArguments
Response: PlayerNotInList - Player does not exist in list; reserved slots list unchanged
Effect: Remove <soldier name> from list of players who can use the reserved slots.

Request: reservedSlots.clear
Response: OK
Response: InvalidArguments
Effect: Clear reserved slots list

Request: reservedSlots.list
Response: OK <soldier names>
Response: InvalidArguments
Effect: Retrieve list of players who can utilize the reserved slots

Maplist

Request: mapList.configFile [filename: filename] - **disabled for security reasons atm**
Response: OK - for set option
Response: OK <filename> - for get option
Response: InvalidArguments
Response: InvalidFileName - Filename does not follow filename rules
Effect: Set name of maplist configuration file

Request: mapList.load
Response: OK - Maplist loaded
Response: InvalidArguments
Response: AccessError - File not found, internal maplist is now empty

Response: InvalidPlaylist - Play list doesn't exist. Should be RUSH, CONQUEST, SQDM or SQRUSH.
Response: InvalidMapName <name> - Map with name <name> doesn't exist in playlist/gamemode
Effect: Load list of map names from file. This is a file with one map name per line.
Comments: If loading succeeds, the maplist will get updated.
If loading fails, the maplist will remain unchanged.

Request: mapList.save
Response: OK - Maplist saved
Response: InvalidArguments
Response: AccessError - Error while saving, on-disk maplist file possibly corrupted now
Effect: Save maplist to file. This is a file with one map name per line.
Comments: If saving fails, the output file may be unchanged or corrupt.
Every line break has windows "\r\n" characters.

Request: mapList.list
Response: OK <map names>
Response: InvalidArguments
Effect: Retrieve current maplist

Request: mapList.clear
Response: OK
Response: InvalidArguments
Effect: Clears maplist
Comments: If server attempts to switch level while maplist is cleared, nasty things will happen

Request: mapList.remove <index: integer>
Response: OK - Map removed from list
Response: InvalidArguments
Response: InvalidIndex - Index doesn't exist in server map list
Effect: Remove map from list.

Request: mapList.append <name: string>
Response: OK - Map appended to list
Response: InvalidArguments
Response: InvalidMapName - Map doesn't exist on server
Effect: Add map with name <name> to end of maplist
Comment: Remember to specify playlist before adding maps

Request: mapList.insert <index: integer, name: string>
Response: OK - Map inserted to list
Response: InvalidArguments

Response: InvalidMapName - Map doesn't exist on server or negative index
Effect: Add map with name at the specified index to the maplist

Variables

Request: vars.adminPassword [password: password]
Response: OK - for set operation
Response: OK <password> - for get operation
Response: InvalidArguments
Response: InvalidPassword - password does not conform to password format rules
Effect: Set the admin password for the server, use it with an empty string("") to reset

Request: vars.gamePassword [password: password]
Response: OK - for set operation
Response: OK <password> - for get operation
Response: InvalidArguments
Response: InvalidPassword - password does not conform to password format rules
Response: InvalidConfig - password can't be set if ranked is enabled
Effect: Set the game password for the server, use it with an empty string("") to reset

Request: vars.punkBuster [enabled: boolean]
Response: OK - for set operation
Response: OK <enabled: boolean> - for get operation
Response: InvalidArguments
Response: InvalidConfig - punkbuster can't be disabled if ranked is enabled
Response: StartupOnlyCallNotAllowed - this command can only be executed from startup.txt
Effect: Set if the server will use PunkBuster or not

Request: vars.hardCore [enabled: boolean]
Response: OK - for set operation
Response: OK <enabled: boolean> - for get operation
Response: InvalidArguments
Effect: Set hardcore mode
Delay: Works after map change

Request: vars.ranked [enabled: boolean]
Response: OK - for set operation
Response: OK <enabled: boolean> - for get operation
Response: InvalidArguments
Response: StartupOnlyCallNotAllowed - this command can only be executed from startup.txt
Effect: Set ranked or not. If enabled: game password will be removed and punkbuster enabled

Request: vars.rankLimit <rank: integer> **##QA: Says 'OK' but still allow higher ranked players to join**
Response: OK - for set operation
Response: OK <rank: integer> - for get operation
Response: InvalidArguments
Effect: Set the highest rank allowed on to the server (integer value).
Comment: To disable rank limit use -1 as value

Request: vars.teamBalance [enabled: boolean]
Response: OK - for set operation
Response: OK <enabled: boolean> - for get operation
Response: InvalidArguments
Effect: Set if the server should autobalance

Request: vars.friendlyFire [enabled: boolean]
Response: OK - for set operation
Response: OK <enabled: boolean> - for get operation
Response: InvalidArguments
Response: LevelNotLoaded - for set operation
Effect: Set if the server should allow team damage
Delay: Works after round restart
Comment: Not available during level load.

Request: vars.currentPlayerLimit
Response: OK <nr of players: integer> - for get operation
Response: ReadOnly - if you try to send any arguments
Response: InvalidArguments
Effect: Retrieve the current maximum number of players
Comment: This value is computed from all the different player limits in effect at any given moment

Request: vars.maxPlayerLimit
Response: OK <nr of players: integer> - for get operation
Response: ReadOnly - if you try to send any arguments
Response: InvalidArguments
Effect: Retrieve the server-enforced maximum number of players
Comment: Setting the user-defined maximum number of players higher than this has no effect

Request: vars.playerLimit [nr of players: integer]
Response: OK - for set operation
Response: OK <nr of players: integer> - for get operation
Response: InvalidArguments
Response: InvalidNumberOfPlayers - Player limit must be in the range 8..32
Effect: Set desired maximum number of players
Comment: The effective maximum number of players is also effected by the server provider, and the game engine

Request: vars.bannerUrl [url: string]
Response: OK - for set operation
Response: OK <url: string> - for get operation
Response: InvalidArguments
Response: TooLongUrl - for set operation
Effect: Set banner url
Comment: The banner url needs to be less than 64 characters long
The banner needs to be a 512x64 picture smaller than 127kb
Example: admin.setBannerUrl http://www.example.com/banner.jpg

Request: vars.serverDescription <description: string>
Response: OK - for set operation
Response: OK <description: string> - for get operation
Response: InvalidArguments
Response: TooLongDescription - for set operation
Effect: Set server description
Comment: The description needs to be less than 400 characters long
##Request from RSPs: In addition being able to enter a new line would be great, BF2142 used the "|" character as newline.

Request: vars.killCam [enabled: boolean]
Response: OK - for set operation
Response: OK <enabled: boolean> - for get operation
Response: InvalidArguments
Effect: Set if killcam is enabled
Delay: Works after map switch

Request: vars.miniMap [enabled: boolean]
Response: OK - for set operation
Response: OK <enabled: boolean> - for get operation
Response: InvalidArguments
Effect: Set if minimap is enabled
Delay: Works after map switch

Request: vars.crossHair [enabled: boolean]
Response: OK - for set operation
Response: OK <enabled: boolean> - for get operation
Response: InvalidArguments
Effect: Set if crosshair for all weapons is enabled
Delay: Works after map switch

Request: vars.3dSpotting [enabled: boolean]
Response: OK - for set operation
Response: OK <enabled: boolean> - for get operation

Response: InvalidArguments
Effect: Set if spotted targets are visible in the 3d-world
Delay: Works after map switch

Request: vars.miniMapSpotting [enabled: boolean]
Response: OK - for set operation
Response: OK <enabled: boolean> - for get operation
Response: InvalidArguments
Effect: Set if spotted targets are visible on the minimap
Delay: Works after map switch

Request: vars.thirdPersonVehicleCameras [enabled: boolean]
Response: OK - for set operation
Response: OK <enabled: boolean> - for get operation
Response: InvalidArguments
Effect: <todo>
Delay: Works after map switch

##QA: Works but is bugged. If you change the setting and someone is in a vehicle in 3rd person view when at end of round, that player will be stuck in 3rd person view even though the setting should only allow 1st person view.