AD Framework

in-game server administration software
for PR:BF2 gameservers

INSTALLATION AND USER GUIDE

Author: AfterDune
Date: 14 January 2011
Framework version: 1.3

Table of Contents

Introduction

Back in 2008, I was a member of the .:iGi:. clan. We had a popular PR server which unfortunately crashed a lot, caused by an unstable BF2CC installation. We decided to run without BF2CC, upon which I created "igi_admin", a very small and simple Python script to display and/or set the next map. The server administrators like it a lot, so more functionality was added. Later on, a smart balance script was written, to even out the teams.

When more and more people started calling me "AD" and the scripts were used by many servers, I changed the name to "AD Framework", representing a collection of Python scripts to make the life of PR:BF2 server administrators a lot easier.

This document describes the AD Framework from A to Z. It will give a brief overview of the framework, a guide to install and configure it, followed by a detailed description of how to use each module.

I hope you find the AD Framework very helpful. Check the "contact"-chapter to see where you can leave your feedback - it's much appreciated.

With best regards,

AfterDune

Available modules

At this moment, there are four modules available within the AD Framework;

The first and most advanced module is "AD Admin". This module provides an in-game administration service that is used to control the server's maplist, manage players and change gameplay settings. Many commands are very similar to BF2CC's commands. This is very convenient for server administrators that switch to the AD Framework.

The second module is called "AD Smart Balance". The module is used to control teambalance. It evens out the teams in a *smart* way.

The third module is called "AD Logs", which in short is a module that logs playerchat.

The fourth module is called "AD Announcer" and takes care of auto-announcements, such as welcome and quit messages and server announcements.

More detailed information about each module can be found in following chapters.

Installation guide

This section explains how to install the AD Framework.

Installation files

The installation zipfile has the following directory structure:

```
admin/
  ad_framework/
    logs/
    ad_admin.py
    ad_announcer.py
    ad_init.py
    ad_framework.py
    ad_framework_settings.py
    ad_logs.py
    ad_smartbalance.py
mods/
  pr/
    python/
      game/
        __init__.py
```

Uninstalling older versions

Before installing a new version of the AD Framework, it's always a good idea to backup the files. For example, the old directory could be renamed from `ad_framework` to `ad_framework_old`. Make sure to add the adminhashes again, so all the admins have the correct rights. For more information about settings, please refer to chapter "Configuration", which explains all the available settings.

How to install

Unzip the contents of the zipfile into your Battlefield 2 directory. If your OS asks you to overwrite files, choose "yes".

On Unix/Linux systems, make sure the `<battlefield2-dir>/admin/ad_framework/logs/` directory is writeable by the user running the BF2 server.

The installation is now complete. You should now configure the AD Framework.

Configuration

After the AD Framework is installed, make sure to customize the settings in `ad_framework_settings.py`. Below is a step-by-step walk through the all the available settings.

Global settings

Turn on/off debugging for all modules
`debugging = True/False`

Make use of Punkbuster when kicking/banning players
`usePunkbuster = True/False`

Enable/disable the sponsormessage. This is a message to the public, saying your server uses the AD Framework. Displaying this message is optional, not required, but would be much appreciated
`sponsorMessageEnabled = True/False`

The text of the sponsormessage (do not change)
`sponsorMessage = [text]`

The interval of the sponsormessage in minutes
`sponsorMessageInterval = [int]`

The version of the running AD Framework (do not change)
`version = [text]`

AD Smart Balance Settings

Enable/disable smart balancing
`smb_enabled = True/False`

The difference in teams when smart balancing is active
`smb_difference = [int]`

Exclude players from smart balancing
`smb_excludeList = [array]`

Swap teams on round start. This option can be used by servers that can't/don't use Modmanager to handle this
`smb_swapTeamsOnStart = True/False`

AD Logs Settings

Enable/disable logging of chat
```
log_enabled = True/False
```

AD Announcer Settings

Enable/disable announcements
```
ann_enabled = True/False
```

Enable/disable join/welcome messages
```
ann_joinMessageEnabled = True/False
```

The text that is displayed when `ann_joinMessageEnabled` is set to `True`. The variable [playername] will automatically be replaced by the player's name.
```
ann_joinMessage = [text]
```

Enable/disable disconnect messages
```
ann_disconnectMessageEnabled = True/False
```

The text that is displayed when `ann_disconnectMessageEnabled` is set to `True`. The variable [playername] will automatically be replaced by the player's name.
```
ann_disconnectMessage = [text]
```

Enable/disable timed messages
```
ann_timedMessagesEnabled = True/False
```

The automated messages/announcements that are displayed when `ann_timedMessagesEnabled` is set to `True`. The values should be like this: `[start after x seconds]:[repeat every x seconds]:[text to display]`.
```
ann_timedMessages = [dictionary]
```

Example:
```
ann_timedMessages = [
    "10:300:Work together and play fair!",
    "20:600:Join us on Mumble!",
]
```

AD Admin Settings

Enable/disable admin commands
```
adm_enabled = True/False
```

Enable/disable logging of succesfully executed admin commands. These logs are saved in `<battlefield2-dir>/admin/ad_framework/logs/ad_admin.txt`
```
adm_logging = True/False
```

The amount of minutes a player can't join the server after being kicked
```
adm_kickTime = [int]
```

The amount of minutes a player can't join the server after being temporary banned `adm_banTime = [int]`

The symbol preceding every admin command. This has to be a single character. Some server administrators prefer to use "@" or "#" (etc.) instead of the default "!"
`adm_commandSymbol = [single character]`

Maximum altitude (in meters) a player can be "flung" using `!fly`
`adm_maxAltitude = [int]`

Width of the screenshot Punkbuster takes
`adm_pbSsWidth = [int]`

Height of the screenshot Punkbuster takes
`adm_pbSsHeight = [int]`

Defining administrators and their powerlevels. The lower the powerlevel, the more power they have
`adm_adminHashes = [dictionary]`

Defining the powerlevels for each command. Administrators can only execute commands of which the powerlevel is the same or higher as their own
`adm_adminPowerLevels = [dictionary]`

Defining the default Mumble message. This text will be returned after `!mumble` is issued.
`adm_mumble = [text]`

Custom reasons that can be used as the reason argument with `!k`, `!kick`, `!b`, `!ban`, `!w`, `!warn`, `!s`, `!say`, `!st`, `!sayteam`, `!resign`, `!kill`, `!tb`, `!stopserver`, `!r` and `!report` commands
`adm_reasons = [dictionary]`

Enable/Disable rules.
`adm_rulesEnabled = True/False`

Defining the rules of the server. Five rules is the max, as a player can only see five lines of text at a time. Remove lines if desired. The rules will be displayed in a personal message after `!rules` is issued.
`adm_rules = [array]`

The complete maplist of PR. This is used by the framework as "reference", so it knows what maps are *available*. This is definitely not the maplist for your server. Leave this untouched. However, if you run custom maps, you can add them to this list
`adm_mapListAll = [array]`

How to retrieve a playerhash

First of all, a playerhash looks like this: `dbf00cb299a40976c6b416af3ad6ab33`. There are various ways to retrieve a playerhash.

Hash command

The easiest way to retrieve a player's hash, is to issue the command `!hash`. This will return a player's hash in a personal message.

BF2 server console

On the BF2 server console, type: `admin.listPlayers` to retrieve a list of connected players, including their playerhash.

RCON tools

Not everybody has access to the server console. RCON tools that connect to the BF2 server should (in most cases) also be able to retrieve a player's hash.

Other methods

If you encounter problems retrieving a player's hash, it's best to visit the Project Reality forums and ask for feedback. A lot of experienced server administrators are more than willing to help.

Module: AD Admin

This module provides an advanced in-game administration service, similar to BF2CC's in-game commands. A list of all the available commands and how to use them can be found below.

Map control

The following commands are to control the maplist.

Set the next map, where layer can be `inf`, `alt` or `std`. The mapID can be retrieved by typing `maplist.list` in the console (default key: ~).

```
!setnext [mapID]
!setnext [partial mapname] [partial gamemode] [layer]
```

Run the next map

```
!runnext
```

Save the maplist

```
!save
```

Remove a map from the maplist

```
!remove [mapID]
```

Change the next map and run it, where layer can be `inf`, `alt` or `std`.

```
!change [mapID]
!change [partial mapname] [partial gamemode] [layer]
```

Restart the map (this restarts the round)

```
!restart
```

Reload the map (this reloads the entire map)

```
!reload
```

Player control

The following commands allow player management. If a command requires a `reason` as argument and this reason is a keyword defined in `ad_framework_settings.py`, it will display the defined text instead.

Resign a player from command or squad position.
```
!resign [partial playername] [reason]
```

Retrieve a player's hash. The hash will be returned in a personal message to the admin.
```
!hash [partial playername]
```

Kill a player
```
!kill [partial playername] [reason]
```

Kick a player off the server
```
!k    [partial playername] [reason]
!kick [partial playername] [reason]
```

Warn a player
```
!w    [partial playername] [reason]
!warn [partial playername] [reason]
```

Ban a player
```
!b   [partial playername] [reason]
!ban [partial playername] [reason]
```

Temporary ban a player. Bans a player for a certain amount of time (default 180 minutes). This temporary ban is lifted when the server is restarted
```
!tb [partial playername] [reason]
```

Take a screenshot from a player through Punkbuster
```
!ss [partial playername]
```

Fling a player (max altitude is 1000 meters)
```
!fly [partial playername] [altitude]
```

Switch a player to the other team. When `now` is the second argument, the player gets switched immediately (killing him first)
```
!switch [partial playername]
!switch [partial playername] [now]
```

Unban a previously banned player. This function is only available when punkbuster is enabled.
```
!ub    [partial playername]
!unban [partial playername]
```

Announcements and text messages

Server announcement. This command displays `text`, which the entire server can see.
```
!s   [text]
```

```
!say [text]
```

Team announcement. This command is similar to `!s`, but only your team, `us`, or their team, `them`, can see it

```
!st      [us/them] [text]
!sayteam [us/them] [text]
```

Display help information (shows all available commands)

```
!help
```

Server- and Pythoncommands

Turn on/off smart balancing. If no arguments are passed, it displays if AD Smart Balance is enabled/disabled. If `difference` is defined, it will set the allowed balance different to that value.

```
!ab
!ab [on/off]
!ab [on/off] [difference]
```

Turn on/off debugging

```
!debug [on/off]
```

Reinitialize the settingsfile

```
!init
```

Teamswap all players. Note that it swaps all players without killing them, only use this when the server is in "pre-game" mode (when the timer is still running)

```
!swapteams
```

Stop the server. This command is more succesful on Linux/Unix systems, especially when a auto restart script is used.

```
!stopserver
```

Turn on/off map- and kickvotes. If no argument is passed, it displays if voting is enabled/disabled.

```
!vote
!vote [on/off]
```

Open commands

These commands can be used by everyone in the server, not just admins. Open commands have a fixed powerlevel of 777, based on chmod's 777 rights, meaning "everybody has full access on this element".

This command is for players to know if any admins are online. The command returns a personal message with a list of online admins.

```
!admins
```

Find out if the server uses Mumble. This command returns a personal message with a short text about the use of Mumble on the current server. This message can be defined by the server administrators.

```
!mumble
```

Reporting a player is used to let in-game administrators know someone is not happy about the way another player is behaving. This command will send a "personal message" to all the in-game administrators. The following command can be used to report a specific player:

```
!r      [partial playername] [reason]
!report [partial playername] [reason]
```

To report anything else, simply type:

```
!r      [reason]
```

```
!report [reason]
```

Display the rules of the server (if defined):

```
!rules
```

Displays the next map in a PM

```
!shownext
```

Display the version of the AD Framework and the enabled modules

```
!version
```

Custom commands

As of version 1.0 of the AD Framework, custom commands are no longer supported. At least not by default. Server administrators can still add their own commands, if they want AD Admin to support more features.

Before making any changes to the scripts, please make a suggestion on the forums first. Perhaps it is something other server administrators can benefit from as well and therefor could be implemented by default. It's not forbidden to make changes, but I do not support it, nor are you allowed to distribute the files under a different name.

If you are determined to add your own commands, here's a way how to do it;

1. In `ad_framework_settings.py`, add an entry to `adm_adminPowerLevels` to define the powerlevel of the command, like: `"command": [powerlevel]`
2. In `ad_admin.py`, add an entry to `adminCommands`, like: `"command": FunctionName`
3. In `ad_admin.py`, add the function you are referring to, preferrably using a layout like this:

```
# What does this function do, enter text here
def FunctionName(cmd, args, p):
    try:
           --insert your code here--

           # Log(command, adminname, victimname, text)
           Log(cmd, p.getName(), "", "")
    except:
        adf.Debug("Exception in FunctionName()")
```

Note: This module can be enabled/disabled in the `ad_framework_settings.py` file.

Module: AD Smart Balance

The Smart Balance module is a more improved version of BF2's "autobalance". The module works as follows;

If a player dies, connects to the server or wants to switch teams, the module will check if it causes balance issues if the player is switched. If it doesn't, the player is switched.

Smart balancing only occurs when:

- The player is dead
- The player is not in the reserved list (see below)
- The player is not a squad leader
- The player is not a commander
- The difference is greater than `x` (defined in `ad_framework_settings.py`)

The module balances the teams when in pregame mode as well (when the countdown is running).

Reserved list

The `ad_framework_settings.py` file contains a "reserved list". Server administrators can put (clan)tags or (partial) playernames in this list, to prevent them from being switched.

Mind you, ALL players that have one of the defined entries in their name or tag, will not be switched. Let's say your clan/communitytag is "[PR]", every player that has "[PR]" wherever in their playername or tag, will not be switched.

The following players will be switched:
{PR}AfterDune
AfterDune(PR)
etc.

The following players will not be switched:
[PR]AfterDune
AfterDune[PR]
etc.

Note: This module can be enabled/disabled in the `ad_framework_settings.py` file.

Module: AD Logs

The Logs module logs all chat on the server in the global, team and squad channels. The logfiles are stored in `<battlefield2-dir>/admin/ad_framework/logs/chatlog_[date].txt`. Logfiles will be created for every round. The content of a logfile looks similar to this:

```
Map          : 7gates
Gamemode     : gpm_cq
Team 1       : CH
Team 2       : GB
ROUND STARTED: 2009-04-15 19:14:55

[19:14 TEAMKILL] A. Endsmine TEAMKILLS D. Aberin
[19:17 TEAM 2  ] AfterDune: get some!
[19:18 SQUAD 1 ] AfterDune: let's go, squad!!!
[19:19 SQUAD 1 ] P. Stormworth: chaaaarge!
[19:18 TEAMKILL] N. White TEAMKILLS M. Hart
[19:19 GLOBAL  ] M. Hart: why did you do that??
[19:21 TEAM 1  ] D. Aberin: I can't hold them, need backup!
[19:22 TEAM 1  ] F. Lindblom: copy, on our way
```

Note: This module can be enabled/disabled in the `ad_framework_settings.py` file.

Module: AD Announcer

This module makes use of Punkbuster's announcer functionality, so Punkbuster must be enabled in order for this module to work. AD Announcer takes care of welcome/quit messages, as well as auto-announcements, such as serverrules.

Note: This module can be enabled/disabled in the `ad_framework_settings.py` file.

Questions and known issues

Why use the AD Framework

Many (PR) servers encounter issues while running BF2CC. The AD Framework provides a very easy-to-use replacement for in-game admin commands, logging and smart balancing.

If you're not convinced or simply want to know why servers use the framework, feel free to create a new thread on the Project Reality forums and ask for feedback from server administrators. They can tell you exactly why they use it, what they like and what they dislike.

ModManager

Many servers have ModManager installed to perform auto teamswitches on roundstart and use the built-in "announcer", which enables server administrators to display custom messages.
At this point, AD Smart Balance can take care of auto teamswap on round start. From version 1.0 of the AD Framework, this is an option in `ad_framework_settings.py`.
Announcers aren't possible yet, but may be implemented in a future release.

AD Framework on Multiplay servers

You can only use the AD Framework if you run a dedicated server. If you're on a shared server, you have to use Clanforge, BF2CC's RCON mode or Multiplay's own in-game admin commands.

AD Framework and BF2CC

I do not recommend to run both the AD Framework in combination with BF2CC. The AD Framework was originally built to replace BF2CC's in-game features, so could interfere with each other. Some servers do use it though, but it's best to change the admin command symbol to something else than the default "!". In these cases, server administrators often choose to go for "@" or "#".

RCON connection issues

This has happened to a handful of server administrators, but fortunately there is a fix. Please visit this link for Jagular's solution, or this one for a revised solution by Ash2Dust.

Admin commands don't work

Please check the following things:
- Did you install the AD Framework correctly?
- Did you edit `ad_framework_settings.py`?
- Did you add playerhashes to the admin commands array?

If it still doesn't work, turn on debugging, this option can be set in `ad_framework_settings.py` (requires a server restart) or through `!debug on` (on the fly, but setting is not saved).

If you have done all the above, but it still isn't working as it should, please post the issue on the Project Reality forums. Server administrators are more than willing to assist you.

Special thanks

In alphabetical order, I'd like to thank the following people for helping me get this far, with testing, developing, etc. (I'm very sorry if this list is not complete, feel free to comment, so I can add you!);

.:iGi:., [MoL]jaVi, Amok@ndy, Ash2Dust, BigHope, BloodyDeed, c0ca, dapriest, Epoc, Epoch, Jagular, Lysander, NM|rotten, Rebiv4, Third Forces

...and all the server administrators that have chosen to administrate their PR gameserver with the AD Framework!

Contact

If you wish to contact me, feel free to do so by sending me a PM (personal message) on the Project Reality forums, or reply in the "AD Framework" thread, located in the (private) Server Admin forums.