EF 1 Custom Content and Settings

Location of Config Files/Mods/Maps

Standard Location

Even though it may be a bit confusing in the beginning, there is not only one place where the settings for the game are stored. With a standard installation from CD, without mods or newer versions than 1.2, the files are located under `baseEF` in the EF games folder or in the corresponding folders of the mods in the games directory. For the mod Pinball for example, instead of the subfolder `baseEF` this is the folder `pinball`. Inside those folders you might find the folders *screenshots* and *saves*, depending on whether you did create screeenshots or saved a single player game.

This however, might not be possible: The worst case scenario here is an NTFS file system which prevents the game from saving the settings at all. (See also Virtual Store on the matter.) With the binaries of ioEF or Lilium Voyager things look different again. Because here (as it should have been for the original version of Raven) the files are stored in the user directory. This has the advantage that every user can have his own configuration and installed mods, maps, models, etc. and no user needs write access to the game directory. The following list considers the main mod, `baseEF` to be the target of the savings. For other mod names change the `baseEF` part into the name of the folder the mod requires.

OS	EF 1.2 and older	Lilium Voyager	cMod	ioEF
Windows up to Win Vista	Game folder	`%appdata%\Lilium Voyager\baseEF`	Game folder	`%appdata%\STVEF\baseEF`
Windows from Win 7 on	`%UserProfile%\AppData\Local\VirtualStore\<name of your EF game folder>\` (only when stored in folders `Program Files` or `Program files (x86)` of your Windows drive)	`%appdata%\Lilium Voyager\baseEF`	Game folder or `%appdata%\STVEF\baseEF` (if no writing access is given to the game folder)	`%appdata%\STVEF\baseEF`
Linux	untested (requires Wine)	`~/.local/share/lilium-voyager/baseEF`	Game folder or `~/.stvef/` (if no write access to game folder)	`~/.stvef/baseEF`
MacOS	untested	`~/Library/Application Support/Lilium Voyager/baseEF`	Game folder or `~/Library/Application Support/STVEF` (if no write access to game folder)	`~/Library/Application Support/STVEF/baseEF`

Custom location

On ioQuake derived versions (Lilium Voyager, Tulip Voyager, cMod, ioEF) there is also the possibility to explicitly tell the game, where to look for and place player's data. This is done by setting the variable `fs_homepath` during server or client start. For example

```
./liliumvoyhm.x86_64 +set fs_homepath ~/my_ef_folder
```

makes the game look for data in the folder `my_ef_folder` in the current user's home folder (for Linux systems). This way it is possible, to kind of have multiple EF installations separately, e. g. with different (incompatible) mods active while actually having it installed just once. Files present inside the games installation folder, e. g. `*.pk3` files, are always found as well. So for running different mods with the same installation it is advisable, to strictly separate the configurations and mod files in different folders, while having only the basic game with only its original files inside the installation folder.

Name of the Configuration Files

EF Variation	Holomatch Config	Singleplayer Config
Vanilla	hmconfig.cfg	efconfig.cfg
ioEF	hmconfig.cfg	-
Lilium Voyager	hmconfig.cfg	-
cMod	cmod.cfg	-

cMod Config File Renamed

cMod tries to behave in a similar fashion as original EF 1.2, meaning, it stores the config and downloaded files in the game folder. It does *not* store the config in the `baseEF` folder and the configuration file is called `cmod.cfg`. So depending on the circumstances, it might also be, that this file is saved in another location, as is described for Standard EF from Win 7 on.

Maps, Mods and Models

As with all popular games that allow modding, EF offers maps, mods, or models (player skins) created by private individuals. Basically the same applies to all three: Download from the provider, *read the Readme* and act accordingly. If there is no readme or no installation instructions, copy the `*.pk3` files into the `baseEF` directory of the EF installation. All maps and models, but also most mods, are created as `*.pk3` files. A `*.pk3` file is basically nothing more than a renamed `*.zip` file containing all the data needed to run the map, model, or mod. A few maps need the expansion pack for EF, because otherwise needed textures are missing or the map won't work. After you have copied the files into the proper folders, you should be able to select the maps or models directly in the game. A few maps are an exception, because they are only accessible with the command `/map <mapname>` (not in the selection menu). `<mapname>` is of course the name of the map you want to load. (This is because there is no map description included. This would be an `*.arena` file in the folder `scripts`, which describes (among other things) which game types are playable with it.) Often you can only tell if mods are used in the game when weapon skins are changed or something similar occurs.

Mods

Mods can enrich the gaming experience considerably, e.g. the pinball mod (the opponent must be shot out of the playing field instead of being fragged). However, some have the problem that the original game can no longer be played. Mods are not to be mistaken for modes, which are built in by default, e.g. the disintegration mode (Instagib with Phaser Compression Rifle). Many mods come with their own QVM file, which can change the game play entirely. Here are some videos about mods for EF 1:

- Let's Play Star Trek: Enterprise - Regeneration
- Asteria Update and Video Testing
- RPG-X Station Asteria - Shuttlepod Full Launch

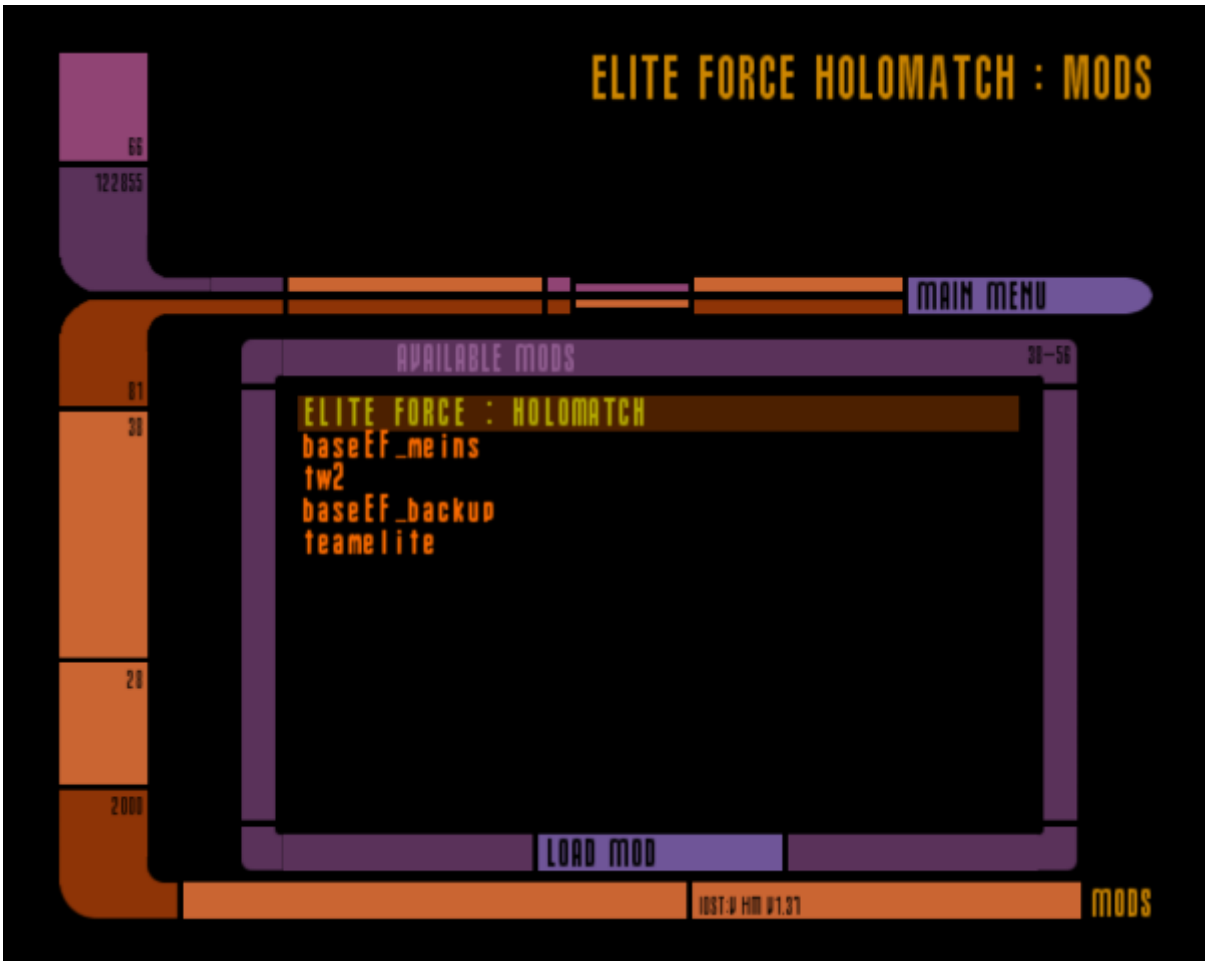
And some actual mods:

- Teamelite, a specialist mod in the style of Team Fortress for EF

- Star Trek: Enterprise NX01 Demo, an (unfinished) mod playing on the Enterprise NX-01
- Star Trek: The Argas Effect, a single-player mod, which plays in the TOS era.
- Nebula 2 - Revenge of the Borg
- Star Trek Voyager Game Project ~~(new EF based on Unreal Engine 4, awesome graphics in comparison)~~ (withdrawn due to certain precautions after lawsuit pressure by CBS)
- Elite Force: Graphic Overhaul Project, New graphics for EF
- Pinball: The goal of the game is to shoot the opponent out of the map, not to kill him by regular fragging.
- RPG-X: A role play game based on EF. See also http://www.griffinendurance.com/asteria/.
- Q32EF: Short for Quake 3 for Elite Force, this mod translates some entities of native Q3 maps to EF entities, making Q3 maps playable for EF. This is actually a mod, so Q3 maps will have to go into the corresponding mod folder.

You can activate mods either with the console of from the menu:

When in the main menu, click on `Mods`, and you get a list of mods you can activate:



Simply select the mod to be activated and click on **Load mod**. The default Mod is **Elite force : Holomatch**. Activating a mod from the console works by adding `+set fs_game <modname>` to the execution command.

Several mods are still used today, running permanently on many servers. Below are a few of them:

- EFAdmin
- Gladiator Arena (Beta Version)
- Power Weapons (Pwrweapon)

- SuperFFA
- TOS Weapons II (tw2)
- xFreeze
- vv-CTF

pak92.pk3

Some of you may have wondered why version 1.37 is displayed, although you were 100% sure that you started the original binary (possibly cracked), which would be version 1.2. Under these circumstances, it makes sense to check the `baseEF` folder to see if the file `pak92.pk3` is located there. This has the same effect. It is part of Thilo's work. There is a distinction between doing something in the binaries (or libraries) and precompiled QVMs (a kind of sandboxing method of Quake 3 to compile code portably). Some of the functionality is implemented in the QVMs that are shipped in pk3 files. Even without your binary being version 1.37 you will get a version 1.37 display through the pk3 archive. But it is obviously a bit confusing that the displayed version doesn't have to be the same as the binary version anymore. Some things that have been changed (probably based on Thilo's diffs):

- Improved random functions in `q_math`,
- Change in shield thickness (there was a bug here that caused the thickness to depend on how the shield was activated),
- Prohibition of suicides in assimilation mode (if used correctly, it would prevent you from assimilating a Federation player, because he could kill himself quickly before, and spawn again),
- Exclusion of forbidden characters in the nickname,
- `localhost` doesn't need a password to join anymore and
- Client-side ignore functions.
- ioEF always rounds values down, while with pak92 it will round properly. This helps jumping out of trenches once again, which is not possible with pure ioEF.

pak92.pk3 is *not* required when playing/hosting a match with vanilla EF, Lilium Voyager or cMod as they all use the original EF calculation methods.

Models

In principle, models are skins for the character, weapons or even new crosshairs. But models have a drawback: If the server doesn't have them as well, the client switches to the standard model. So (after some tests with EF 1.2) you don't become invisible or something similar. If the server also has the model, it will be downloaded from the clients anyway and will be available for everyone.

Maps

See article Star Trek: Voyager Elite Force Maps on this subject.

See Also

Configuration

Star Trek: Voyager Elite Force