Startup

GENERAL USAGE: **eip.exe <key> <rom>**

Options:

eip.exe <sys_key> <rom>	Launches a specified emulator's rom using a "vector". A vector is simply the combination of a **<sys_key>** and a **<rom>** separated by a space. The **<sys_key>** is defined in arcadeEIP and identifies a specific emulator (examples could be "mame" or "atari2600"), and **<rom>** is the name of the game rom to be launched with that emulator. **<rom>** is typically just the rom file's root name, but may optionally include its filename extension and path. Note that the path is not normally needed if a **search_path** was specified in arcadeEIP.ini). For example: **>eip.exe mame defender** This will launch the game *Defender* in MAME. **Direct Switching** If this command line is run WHILE an instance of arcadeEIP is already executing, arcadeEIP will detect the other instance and tell that instance to SWITCH its game (rather than actually create a second instance). The second instance will then terminate itself. This feature permits external tools such a button devices (like the Elgato Stream Deck) or other background tasks to tell arcadeEIP to switch games through a simple command line mechanism. For example, if *Defender* is running as per the command line above, and an external device or process issues this command in the background: **>eip.exe atari2600 space invaders** This will cause the instance of arcadeEIP running the game *Defender* in MAME to exit that game, and switch to the game *Space Invaders* using whatever Atari 2600 emulator has been specified in arcadeEIP. Note that arcadeEIP can also be told to switch games using other methods, such as hot-strings; however, using this method is generally simpler, and does not require arcadeEIP to already be running to use, which makes it more flexible.

eip.exe	Launches the default emulator without a specified emulator or rom. When this happens, arcadeeIP will run either the seed rom (specified in the **[General]** section of arcadeEIP.ini), or a game from the autoplay list if that feature is turned on.
eip.exe <fe_key>	Providing the **fe_key** of a front-end as an argument will cause arcadeEIP to be launched in its "Front-End" wrapper mode. This is useful for front-ends that don't support run before/run after settings. In this mode, arcadeEIP will launch the front-end whose identity and options are defined in the **[Front_End_<fe_key>]** section of arcadeEIP.ini. It will then launch any **run_apps** defined in that section (which might, for example, change joystick/button configuration for that specific front-end), and return them to the settings in the **[Front_End_os]** section after exiting. For example: **>eip.exe bigbox** Note that while the front-end is running, any logging it performs will be written to the file, **arcadeEIP_fe.log**. Also note that arcadeEIP's use of other settings in the front-end's .ini section (such as **start/exit_screen**, **hide_cursor**, **window_name**, etc.) do not require the front end to be launched in this manner. It is only needed to supply support for run before/after apps to the front-end.
eip.exe <rom>	If **<rom>** is provided without a **<sys_key>**, then a search will be conducted in all emulators having a **search_path** defined, and if found, launched with the emulator in which it was associated. This format is not recommended for formal use; however, it is handy for quickly launching and/or testing games by typing them in at the keyboard. For example, if eip.exe is in your system path, typing in something quickly into the Windows Run box like this: **>eip zax** Will launch Zaxxon in the first emulator it is found in (or maybe not, if no rom can be found containing the text "zax")

OTHER EXAMPLES:

>eip.exe	Launches the seed rom or rom from the autoplay list.
>eip.exe mame berzerk	Launch Berzerk in MAME
>eip.exe berzerk	Launch Berzerk using the first emulator for which the rom is found.

`>eip.exe berz`	Also launches berzerk (because “berz” is a sufficient match to “berserk”)
`>eip.exe mame D:\Emulators\MAME\roms\berzerk.zip`	Launch berzerk in MAME using full path to rom
`>eip.exe coleco "D:\MyRoms\Coleco\Donkey Kong.bin"`	Launch “Donkey Kong” in coleco emulator using the full path
`>eip.exe coleco donk`	If a **`search_path`** is defined, will launch first rom having “donk” as a substring

Notes:

Normally, all parameters and options to launch a game will be included in the arcadeEIP.ini file, and so no additional command line options should be required to run its emulator fully configured. Nevertheless, options for additional command line arguments are available for special cases. There are three types:

Overrides: These are optional parameters that will override certain specific settings in arcadeEIP.ini, such as apps or the bookend screens, or the **`-use_args`** option, which can be used with front-ends that hardcode the command line in order to provide compatibility with the uniform command line.

Directives: Optional parameters that represent built-in command line features, including listing all systems, all roms associated with a system, searching for roms, viewing the log, creating proxy files in bulk, etc.

Passthrough Parameters: Allows command line options of your choice to be passed through to specific emulators. For example, you could add a passthrough parameter such as **`<-video,1>`** for mame’s **`-video`** parameter allowing it to be optionally specified on the command line such as:

```
>eip.exe mame berserk -video gdi
```

Overrides

These *optional* parameters may be used to override settings in arcadeEIP.ini. May appear anywhere on command line in any order.

GENERAL USAGE: **-[**parameter_name**]=<**arg1**><,**arg2**>…** (*must have no spaces*)

-use_args=<sys_key><,index>	This option is only needed if your front-end (for example, GameEx) hard-codes the command line for emulators (like MAME) in such a way that it is not compatible with arcadeEIP's uniform command line. Adding this option to the parameters will effectively override the command line to use a specified emulator (**sys_key**) and the specified indexed argument of the input command line. See example below.
-start_screen=<secs><,color>	Can be used to overrides the arcadeEIP.ini **start_screen** setting. See that option for more information.
-exit_screen=<secs><,color>	Can be used to overrides the **exit_screen** setting. See that option for more information.
-run_apps=<app_key><,app_key>	Can be used to overrides the **run_apps** setting. See that option for more information.

EXAMPLES:

-use_args=mame,1

If the front-end hard-codes its MAME command with extra arguments that arcadeEIP can't use, like:

berzerk -rompath D:\Emulator\MAME\roms

then adding **-use_args=mame,1** to this parameter list will cause the effective command line to effectively become:

mame berzerk

To experiment with how this works from the command line try typing in the command line:

eip.exe berzerk -rompath D:\Emulator\MAME\roms -use_args=mame,1 -debug

and then review the log to observe how it is processed.

-start_screen=3,Navy
Override **start_screen** to display for a minimum of 3 seconds with a background color of Navy Blue

-exit_screen=-1
Override **exit screen**. Do not display the exit screen

-run_apps=
Set **run_apps** to blank to *not* run any apps.

Directives

These parameters perform special operations and only work with the util.exe command line utility.

GENERAL USAGE: util.exe <options> – see examples

-show_log<=log_name> **-log<=log_name>**	Display the last log file. Will display **arcadeEIP.log** if no **log_name** is specified. Must be used by itself on the command line. Alternate short form.
-debug_mode<=fe_key> **-debug<=fe_key>**	Run in debug mode. If no **fe_key** is specified, "os" will be used by default. May be added after uniform command line arguments. Alternate short form.
-systems	List all systems defined in arcadeEIP.ini by **sys_key**.
-list [sys_key]	Lists all roms in search path for this emulator (must have a **search_path** defined for this to work). Must be used by itself on the command line.
-find <sys_key> rom	Can be used to discover exactly what rom will be launched using the parameters. Using `-find` lists the first instance of the rom name found (must have a **search_path** defined for this to work). If a partial name is given, the first best match will be displayed. If a **sys_key** is provided only the emulator associated with that **sys_key** will be searched. If a search term has more than one word (like "donkey kong" double quotes are encouraged.
-findall <sys_key> rom	Using `-findall` performs a fuzzy search of all roms in all emulators having that name (must have a **search_path** defined for this to work). If a partial name is given, all roms containing that string will be displayed. If a **sys_key** is provided only the emulator associated with that **sys_key** will be searched. An asterisks (*) will be placed in front of the rom that will be run if using the same parameters. If a search term has more than one word (like "donkey kong" double quotes are encouraged.
-createproxy [source] [destination] [emulator][rom vector] <prefix>	Creates *vector type* proxy files in the *destination* folder for all the roms in a *source* folder. Each created file will contain one line with two parameters (*emulator* and a *rom vector*; see below). *Source* is a path formatted similar to a **search**_path; that is, it is a folder spec ending with a wildcard file specifier that may have more than one file extension added. Remember to add double-quotes if path contains a space. For example: **"D:\Emulators\My Emu\roms*.bin,zip"**

	Destination is similar but must contain exactly one wildcard file extension. This will be the extension used on all the created proxy files. Note that proxy files can go anywhere, so they may go in the same folder as the source (providing the extensions are different) or an entirely different folder. For example: **"D:\Emulators\My Emu\proxies*.txt"** *Emulator* is the **sys_key** of the target emulator, it is inserted into the proxy file as the first parameter. *Rom vector* determines the value of the second parameter in the proxy file. The following values may be used: • rom_name - inserts the rom name (no extension) • rom_file - inserts the rom filename (with file extension) • rom_full_path - inserts the full rom path and filename (***recommended***) • [rom_name] - inserts the [rom_name] template, which will be filled-in at runtime. • [rom_file] - inserts the [rom_file] template, which will be filled-in at runtime. • [rom_full_path] - inserts the [rom_full_path] template, which will be filled-in at runtime. *Prefix* is optional, but if included will prefix the filename with a string of your choice (such as @!@), plus add the *Emulator* key as well. This is specifically intended to support the creation of direct-switch hotstring files that can be placed in the \Direct folder.
-createcfg	Creates a arcadeEIP.ini file template that you can then modify. You must not have an existing arcadeEIP.ini file in the folder in order for this to work.
-createcfgexample	Creates a file called classic_example.ini, which is a highly notated version of arcadeEIP.ini that provides a tutorial and configuration examples. It can be renamed as arcadeEIP.ini to use as the actual configuration file, although it is probably better to use the sparser (and less cluttered) version generated by -createconfig once you understand how the file works.
-clean	Clears all dynamic history and state information as well as all picks, custom lists, state, history, attract mode list, ratings, and favorites. Does *not* affect any configuration settings.
-sound	Lists all sound devices. This is used to help determine what value to use in the **sound_device=** setting in the **[General]** section of arcadeEIP.ini. See documentation for the sound_device option in *Configuration Reference Guide.pdf* for more information.

EXAMPLES:

`-show_log`	Display **`arcadeEIP.log`**.
`-log`	Display **`arcadeEIP.log`** (short form)
`-log arcadeEIP_fe`	Display the **`arcadeEIP_fe.log`** (front-end launch) log if it exists.
`-debug_mode`	Run in debug mode (using default “os” front-end section settings).
`-debug`	Run in debug mode (short form).
`-debug_mode=gameex`	Run in debug mode (using gameex front-end section settings).
`-list atari2600`	Lists all roms in the **`search_path`** for the emulator defined by **`sys_key=atari2600`**
`-find kong`	Searches all emulators having a search_path for the first instance of one containing “kong”
`-find mame pac`	Searches mame emulator for first instance of a rom containing the text “pac”
`-findall kong`	Returns all roms from all emulators whose names contain the text “kong”
`-findall atari2600 “donkey k”`	Returns all roms from atari2600 emulator containing the text “donkey k”
`-createcfg`	Create sparce configuration file.
`-createcfgexample`	Create commented configuration file.
`-clean`	Deletes history, states, favorites, custom lists, and attract list
`-sound`	Lists all sound devices and provides their names and numeric indexes.
`-createproxy D:\Emulators\MyEmu\roms\*.bin,zip D:\Emulators\MyEmu\proxies\*.txt atari2600 rom_full_path`	Creates vector-type proxy files for all roms ending with .bin and .zip from the D:\Emulators\MyEmu\roms folder and places them in the D:\Emulators\MyEmu\proxies folder. Each proxy file will end with the extension .txt and will contain the full path to the rom.
`-createproxy D:\Emulators\MAME\roms\*.zip D:\arcadeEIP\temp\*.txt mame rom_full_path @!@`	Creates direct-switch files of all MAME roms using the prefix “@!@” and the emulator key in the designated folder (example file name: **`@!@mame_galaga.txt`**)