# Censor Server Module

For silEnT mod, an Enemy Territory Modification

Version 1.0

# TABLE OF CONTENTS

# 1. Background

## 1.1. General

Censor server module is an extension to silEnT mod, an Enemy Territory game modification. Enemy Territory is a free to play game created and released by SplashDamage as a sequel to Return to Castle Wolfenstein, a game created by id Software.

The server censor module implements censoring of messages sent by players as well as player names. Messages and names are censored by replacing “bad” words with asterisk characters, so that the word is not offending anymore.

## 1.2. SilEnT mod censoring possibilities

Up to now silEnT mod offered censoring system based on 2 components:

- g_censor and g_censorNames cvar words list,
- Neil Toronto's censor system.

Both are just the list of words that are considered abusing and will be replaced with asterisk characters. The latter is built in the mod and can be extended with the words from the former.

Neil Toronto's censor filter is able to catch some symbol and number replacements and spaces. It also adds some common words and common words with "swears" in them that are white-listed. For example, it will not censor "assassin" but it will censor "ass".

Extending the words list can be done by adding words to the following cvars:

- *g_censor,*
- *g_censorNames*

in the *silent.cfg* file, e.g.: set g_censor "word1, word2, word3".

The above solution has limitations, one of them is the length of the cvar string, the other is the limited number of words.

Some languages have complicated or extensive declination, so it's really hard to prepare the list of words that would represent all possibilities of declination.

## 1.3. Regular expressions

As a solution to the problem described above, silEnT mod introduces the new way of censoring: censoring by the regular expressions.

For those who don't know what regular expressions (often succinctly called as regexes ) are, let's describe it as patterns used to match character combinations in strings. For example, to look for a specific name in a phone list or all of the names that start with the letter 'a'.

I'm sure you are familiar with the use of "wildcard" characters for pattern matching. For example, if you want to find all the Microsoft Word files in a directory, you search for "*.doc", knowing that the asterisk is interpreted as a wildcard that can match any sequence of characters. Regular expressions are just an elaborate extension of this capability.

Regular expressions allow to map relatively small set of patterns to larger set of possible values those patters could match.

SilEnT mod uses regex syntax compatible with Perl regex.

## 1.4. Examples

New possibilities emerge with the regular expressions when it comes to searching words to censor. Let's consider we want to censor the following sentence:

*You are a bunch of* ***muffins****! All admins are crazy* ***muffins****! I* ***muffin*** *you!*

We want to catch all muffin-ish words, so in the stanrard censor system, we would have to add:

*muffin, muffins* to g_censor cvar.

With regex one would have to define something like this:

\w*muffin\w*

which decodes like below:

- \w* - match zero or more the "word" characters (\w – word character, * - zero or more),
- muffin – the explicit character sentence,

additionally it has to be in that specific order.

SilEnT censor would output the following in that case:

You are a bunch of ***! All admins are crazy ***! I *** you!

Let's consider something more interesting. It often happens that players in order to not get censored write something like below:

m u f f i n i n g    admins, m_u_f_f_i_n you!

It would probably sneak unnoticed for the old censor system.

What about new regex censor?

With the following regex patter we would catch it:

\w*(?:\s|_)*m(?:\s|_)*u(?:\s|_)*f(?:\s|_)*f(?:\s|_)*i(?:\s|_)*n\w*

Let's decode it:

- \w* - match zero or more the "word" characters (\w – word character, * - zero or more),
- (?:\s|_)* - not catching group, zero or more

  - (?:) - means we want to catch a group but we don't want to catch it into the results,
  - \s|_ - match a whitespace character or underscore (\s – whitespace, | - or, _ - underscore)
  - * - match the whole group sequence zero or more times.

Again regex silEnT censor system would detect such thing and would return:

*** i n g    admins,*** you!

As you can see it opens up new horizons when it comes to censoring and it really comes in handy.

# 2. Configuration

## 2.1. Configuring silEnT Mod Server

To enable the use of the censor module, the server game must be configured for it. This is done with "modules.cfg" file and by adding specific data block about the statistics module into that file. The modules.cfg file is automatically read every time the server initializes the qagame game. A block like the following needs to be added to the configuration file:

```
[censormodule]
path = servermodules
file = censormodule
config = censormodule.cfg
enabled = 1
```

- *[censormodule]* ; This identifies the module that is configured.
- *path* ; This is the path to the directory where the module shared library (censormodule.dll/so) is located. This path is relative to the "fs_homepath\fs_game" directory. I.e. it is appended to  a path that begins from the directory where your qagame for silEnT mod is located.
- *config* ; This is the configuration file which is read by the statistics module. Do note that this value can also include relative path to the file.
- *enabled* ; If this is set to 1, the module is enabled, if it is set to 0 it is disabled. This allows enabling and disabling the module without making other changes to the configurations.

## 2.2. Configuring the Censor Module

Censor module is configured with "*key = value*" pairs read from a configuration file that is pointed to by the generic *modules.cfg* configuration. Commenting is done with ';' character. If the ';' character appears anywhere in the configuration file, the rest of that line is ignored. The file name is passed to the module from silEnT mod.

### 2.2.1. Example config file contents

Let's take a look at the example configuration file below:

```
;bad polish words
pw1 = \w*babeczk\w*|\w*truskawk\w*|\w*pomidor\w*
pw2= \w*cebul\w*

;bad english words
ew1 = \w*(?:\s|_)*m(?:\s|_)*u(?:\s|_)*f(?:\s|_)*f(?:\s|_)*i(?:\s|_)*n\w* ;with spaces
ew2 = \w*beer\w*
```

*Text 1: Example censor module config file with patters*

Patterns can be named for better organizing, grouping (pw1, pw2). After the key there ought to be the equals sign (=) after which the pattern itself should be entered. Comments are marked in gray.

It's worth mentioning that *pw1* pattern from the example matches more than one word, it actually matches:

- something with *babecz* inside,
- something with *truskawk* inside,
- something with *pomidor* inside.

Such thing is especially useful when used with polish alike languages, where the declination is complicated and there might be many possible options, still swears, that could be created from the single stem, eg.:

- przy**pomidor**zyc, **pomidor**nac, od**pomidor**owac, do**pomidor**zyc, do**pomidor**owac,
- na**pomidor**zyc, no**pomidor**owany, od**pomidor**uj, o**pomidor**owac, o**pomidor**uj, o**pomidor**ujcie,
- etc,

and the list is not finished yet, it's not even the half of possibilities :)

That depicts the problem of creating the full censor system with the old censoring approach.

### 2.2.2. How to write regular expressions for silEnT mod censor module

SilEnT mod censor module regular expressions semantics is compatible with Perl regular expressions semantics.

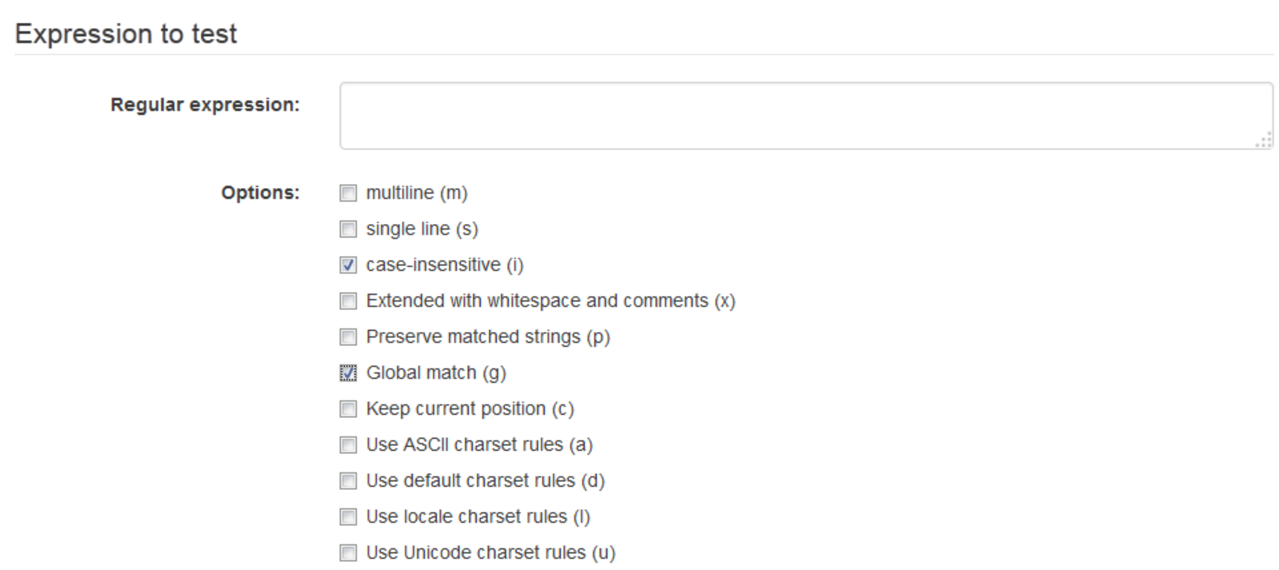
Visit this link for some hints:

http://perldoc.perl.org/perlre.html#Regular-Expressions

To test regular expressions we recommend this site:

http://www.regexplanet.com/advanced/perl/index.html

just remember to set it up like below (check “case insensitive” and “global match”):



Expression to test

**Regular expression:**

**Options:**

☐ multiline (m)
☐ single line (s)
☑ case-insensitive (i)
☐ Extended with whitespace and comments (x)
☐ Preserve matched strings (p)
☑ Global match (g)
☐ Keep current position (c)
☐ Use ASCII charset rules (a)
☐ Use default charset rules (d)
☐ Use locale charset rules (l)
☐ Use Unicode charset rules (u)

*Illustration 1: Regex test setup*

As you can conclude from the above censor module matches case-insensitive, so this reduces the possible combinations of patterns, so instead of:

[C,c]ranberry

it's enough to write just:

cranberry