Bad Word Filter PRO

Keep your games civilized

Documentation

Date: 23.03.2022

Version: 2022.1.0

https://www.crosstales.com

Table of Contents

Thank you for buying our asset "Bad Word Filter"!

If you have any questions about this asset, send an email to bwf@crosstales.com.

Please don't forget to rate it or write a little review – it would be very much appreciated.

1. Overview

The "Bad Word Filter" (BWF) is a profanity/obscenity filter and is exactly what the title suggests: a tool to **filter swearwords** and other "bad sentences".

There are multiple uses for the "Bad Word Filter" in your projects, but the three most obvious would be **user names** (e.g. for high-scores), in a **chat** within the game and **character names**. If you don't want some wannabe-funny-guy to use the user name "a55-face", "S+alin" or any other word you don't approve of, just enable the "Bad Word Filter" and instead of the swearword something like this comes out: #$@&%*!

In our library included are the following **25 languages** (bad words as **regular expressions** which match **ten thousands** of word **variations**):

Arabic, **Chinese**, **Czech**, **Danish**, **Dutch**, **English**, **Finnish**, **French**, **German**, **Greek**, **Hindi**, **Hungarian**, **Italian**, **Japanese**, **Korean**, **Norwegian**, **Persian**, **Polish**, **Portuguese**, **Russian**, **Spanish**, **Swedish**, **Thai**, **Turkish** and **Vietnamese**.

Furthermore, you can add (or remove) any word and language you'd like!

We also included the following special filters: domains (**URLs/emails**), **reserved names** (from games, movies, comics etc.), **global bad words**, **emojis** (miscellaneous symbols), excessive **capitalization** and **punctuation**.

The "Bad Word Filter" works with **any language** and **writing system**. It is easily **customizable**, runs on **all Unity platforms** and the **source code** (including all bad words provided) is also contained within the package.

2. Features

2.1. Filter function

- **Review**, **select**, **read** and **replace**:

    - **Bad words** and **inappropriate phrases**
    - **Domains, URLs** and **email addresses**
    - Excessive **capitalization** (such as "HEY DUDE")
    - Excessive **punctuation** (e.g. "!!11111")

- All four filters can be used **separately** and changed during **run-time**

2.2. Languages

Contains over **5'000** of **regular expressions** in **25** languages - equivalent to **tens of thousands** of word variations!

Supports **any language** and **any writing system**:

- Arabic, Cyrillic, Chinese, Greek, Japanese etc.
- Including ltr / rtl (left to right or right to left)
- Automated (easy) pluralization of English terms

2.3. Flexible & expandable

- **Easy modifications/additions** to the existing sources ("bad words")
- **Multi-threaded** and **lightning fast** - even with **thousands** of words!
- **C# delegates** and **Unity events**
- Use the preconfigured providers for **resources**, **files** and **URLs** - or add your **own provider** (e.g. for XML, JSON)
- **Meta data** for sources (e.g. **descriptions**, **icons**)
- All **sources** ("bad words") provided

2.4. Documentation & control

- **Test** all the functions in the **editor**
- Powerful **API** for **maximum control**
- Detailed **demo scenes**
- Comprehensive **documentation** and **support**
- Full **C# source code**

2.5. Compatibility

- Supports **all build platforms**
- Works with **Windows**, **Mac** and **Linux** editors
- Compatible with **Unity 2018.4 – 2022**
- Works with **Online Check**
- **PlayMaker** actions

3. Demonstration

The asset comes with demos scene to show the main usage.
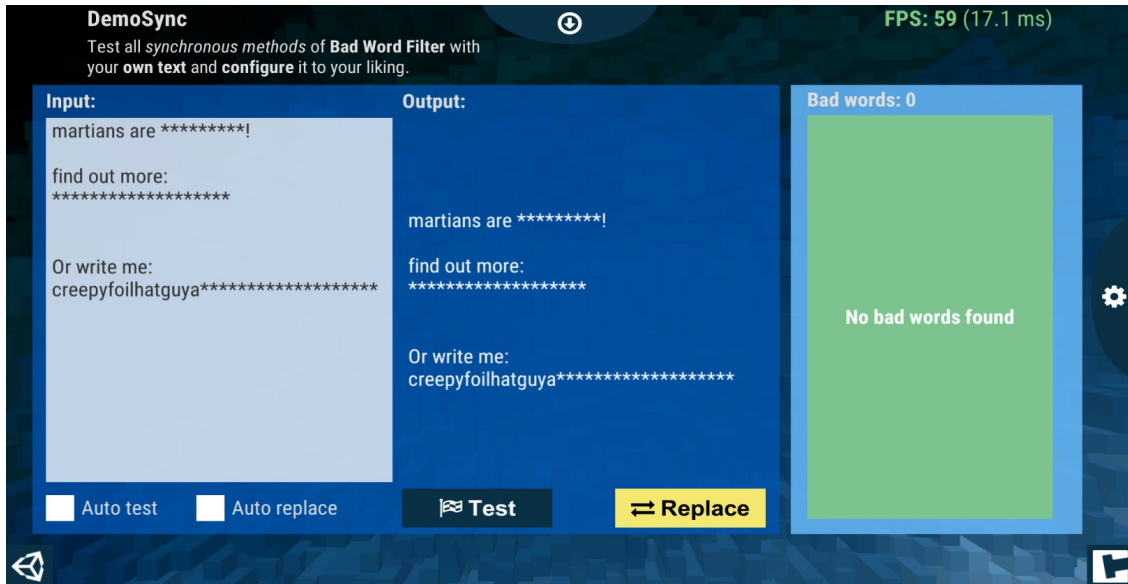
3.1. Note about real-time/automatic checks and replaces

Due to performance impact, it isn't recommended to call the methods of "BadWordFilter" every frame (like in "Update"-method). Use check/replace intervals of **250ms or more** (see "GUIMain.cs" for an example implementation) or use the async methods.

4. Setup

BWF has global settings under "Edit\Preferences..." and under "Tools\BWF PRO\
Configuration...":

The "Bad Word Filter" consists of four separate parts which all can be used
independently:

1. **BadWordManager** for filtering "bad words"

2. **CapitalizationManager** for preventing extensive capitalization

3. **DomainManager** for filtering URLs and emails

4. **PunctuationManager** for preventing extensive punctuation

To access the whole functionality, please use the **BWFManager** instead.

4.1. Schema

The following graphic explains the relationships between all relevant components:
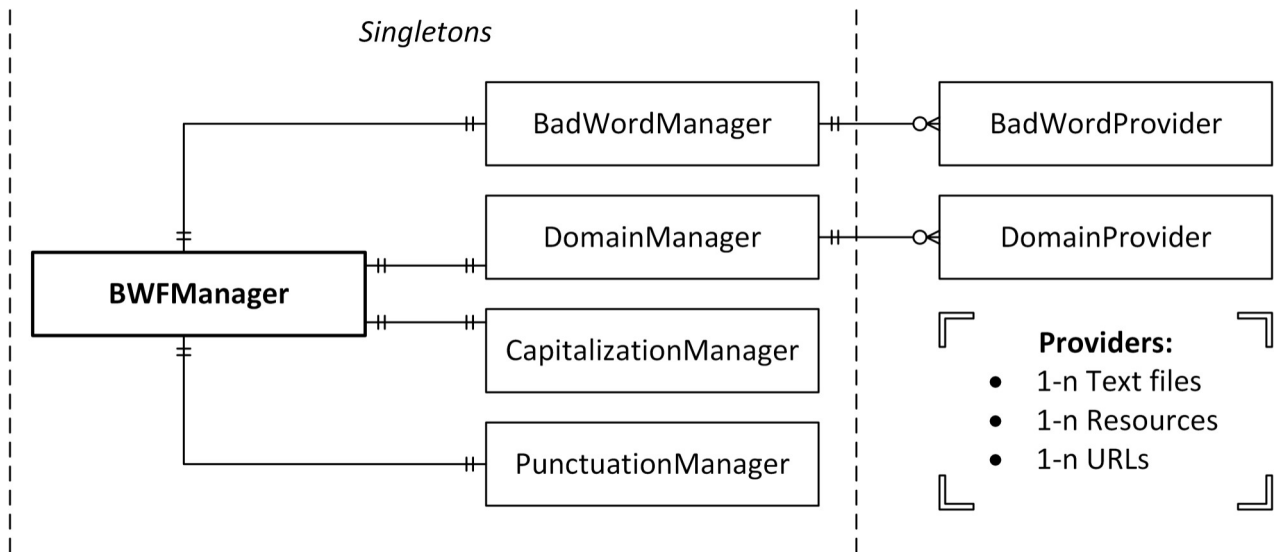
4.2. BadWordManager

The BadWordManager is the main component for filtering **bad words** and **sentences** in strings.

4.2.1. Bad Word Provider LTR

This is the slot for all **left-to-right** (ltr) based **source providers** (like English).

For more information about providers, please see below.

4.2.2. Bad Word Provider RTL

This is the slot for all **right-to-left** (rtl) based **resources** (like Arabic).

For more information about providers, please see below.

4.2.3. Replace Chars

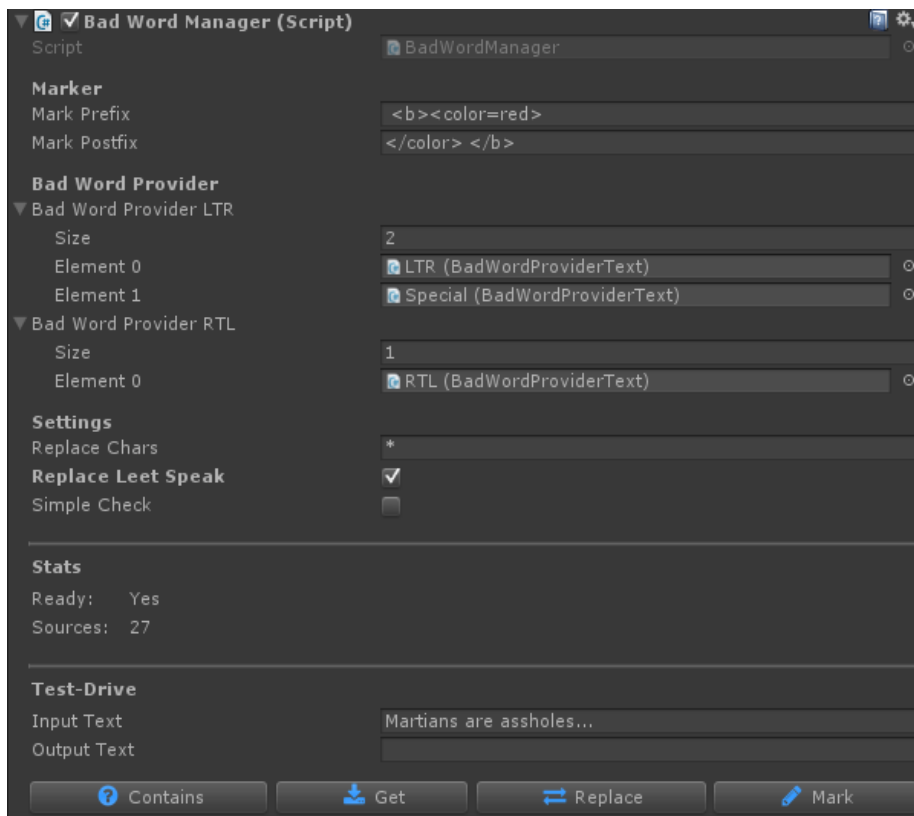These are the desired **1-n replace characters** which were used to replace bad words/sentences. Multiple replace characters will be used to generate random strings for bad words or sentences.

4.2.4. Replace Leet Speak

This option replaces all leet speak characters in a word through the proper meaning, e.g. a$$ will be detected as "ass".

4.2.5. Simple Check

This option enables the "Simple Check"-mode for the manager. This detects the bad words in all sentences, regardless of any regex-settings. This is required for **Chinese**, **Japanese**, **Korean** and **Thai**.

4.3. CapitalizationManager

The CapitalizationManager is the main component for filtering excessive **capitalization** in strings.
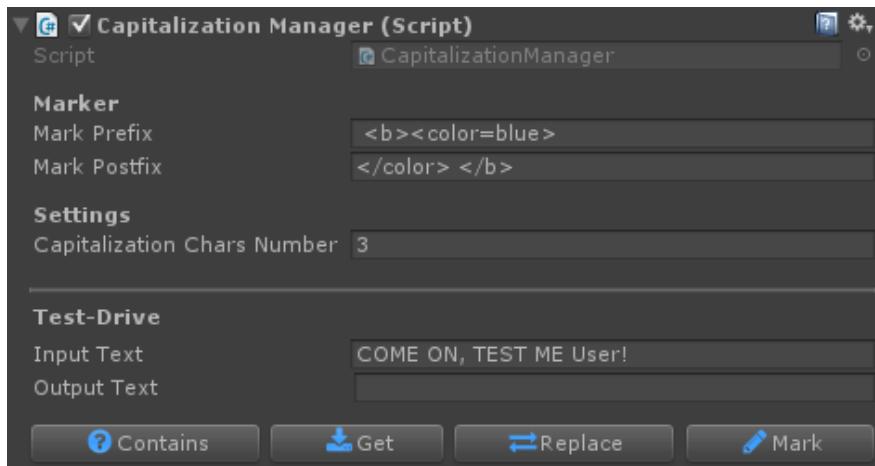
4.3.1. Capitalization Chars Number

Defines the number of allowed capital letters in a row.

4.4. DomainManager

The DomainManager is the main component for filtering domains, **URLs** and **emails** in strings.
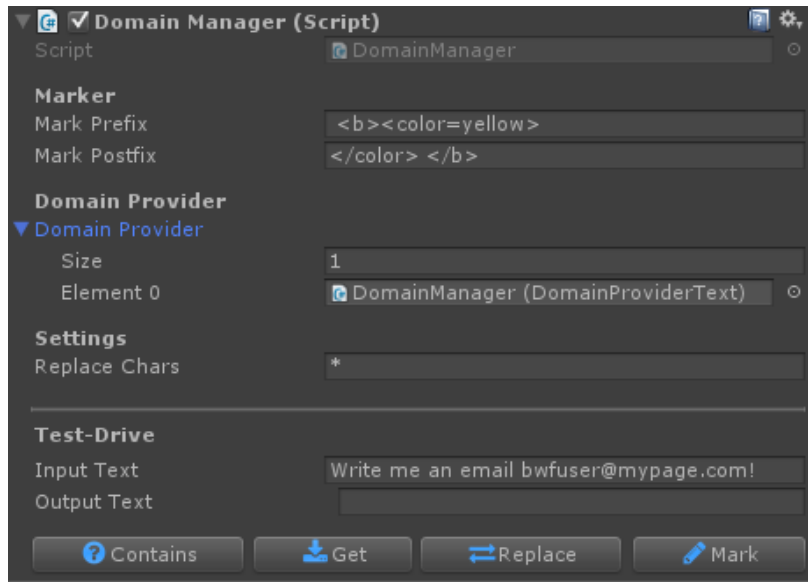
4.4.1. Domain Provider

This is the slot for all **domain providers**.

4.4.2. Replace Chars

These are the desired **1-n replace characters** which were used to replace domains, URLs and emails.

4.5. PunctuationManager

The PunctuationManager is the main component for filtering excessive **punctuation** in strings.
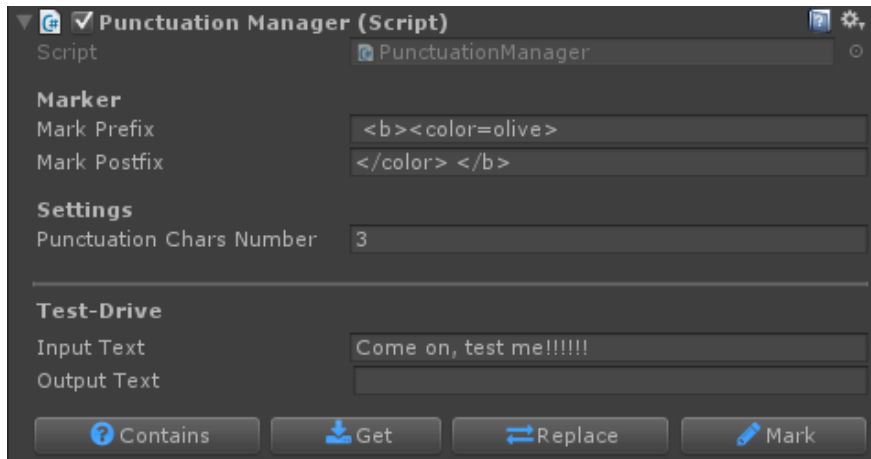
4.5.1. Punctuation Chars Number

Defines the number of allowed punctuation letters in a row.

4.6. BWFManager

The „BWFManager" simply unites all available managers to a "single-point-of-entry".

There are four ways to add it to a project:

1. Add the prefab **BWF** from Assets/Plugins/crosstales/BadWordFilter/Resources/Prefabs to the scene

2. Or go to *Tools => BWF PRO => Prefabs => **BWF***

3. Right-click in the *hierarchy-window => BWF PRO => **BWF***

4. Add it from the Prefabs-tab:
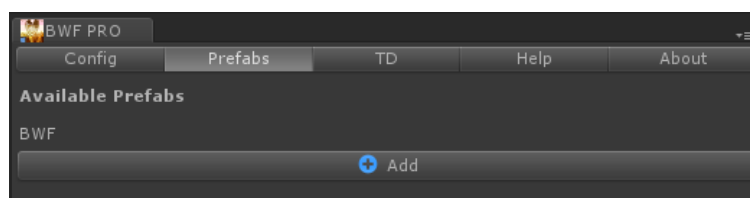
5. Providers

Providers are a collection of sources (e.g. "English bad words" or "Internet domains"). The main benefit is the extensibility of this concept:

1. Unity resources

2. Accessing files on the local machine

3. Accessing files on a web server

You can easily extend the base classes and build whatever you like (e.g. a provider to access data from XML or JSON).

Providers are used by the BadWordManager and DomainManager.

5.1. BadWordProviderText

This is the default provider for sources:
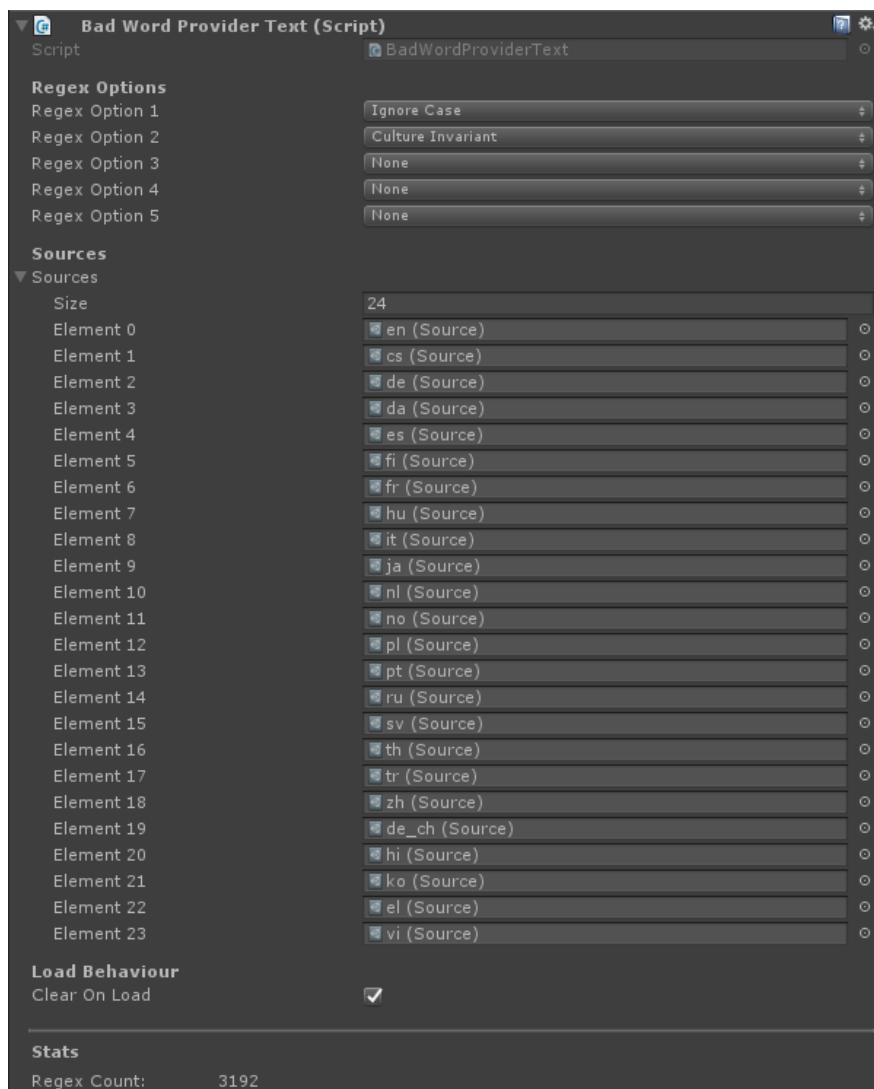
The parameters are explained below.

5.1.1. **Name**

Identify the provider by a meaningful name – this is only for us humans.

5.1.2. **RegexOption 1 – 5**

The RegexOptions 1 – 5 are for **fine-tuning** the match and replace conditions.

Warning: don't change these parameters if you don't know what you are doing. It could dramatically change the **accuracy** and have a big **negative performance** impact.

For further information please take a look at:

https://msdn.microsoft.com/de-de/library/system.text.regularexpressions.regexoptions%28v=vs.110%29.aspx

5.1.3. **Sources**

These are the 1-n sources for the provider.

Name	Name of the source
Description	Description for the source
Icon	Icon to represent the source (e.g. country flag)
URL source	Local or remote text file containing all regular expressions for this (see below)
Resource	Text file containing all regular expressions for this source (see below)
Clear on Load	Clears all bad words when 'Load' is called

5.2. **DomainProviderText**

This provider is similar to the "BadWordProviderText".

6. Sources

6.1.1. Included bad word sources

File name	Content	RegEx-count
ar.txt	Arabic	49
cs.txt	Czech	41
da.txt	Danish	10
de.txt	German	135
de_ch.txt	Swiss-German	18
Emoji.txt	Emojis and miscellaneous symbols	3
en.txt	English	342
es.txt	Spanish	178
fa.txt	Persian	45
fi.txt	Finnish	178
fr.txt	French	149
global.txt	Global forbidden words	52
gr.txt	Greek	77
hi.txt	Hindi	96
hu.txt	Hungarian	96
it.txt	Italian	211
ja.txt	Japanese	273
ko.txt	Korean	113
names.txt	Forbidden user names (from games, movies, comics etc.)	193
nl.txt	Dutch	257
no.txt	Norwegian	82
pl.txt	Polish	42
pt.txt	Portuguese	67
ru.txt	Russian	166
sv.txt	Swedish	107
th.txt	Thai	31
tr.txt	Turkish	27
vi.txt	Vietnamese	46
zh.txt	Chinese	450

You can modify the existing resources as you like (edit, add or remove words/sentences).

6.1.2. Included domain source

We included the resource "domains.txt" which contains all official domains from:

http://data.iana.org/TLD/tlds-alpha-by-domain.txt

But as usual, you can modify this list or add your own with specific URLs or emails.

6.1.3. How-to create your own sources

A source must be an **UTF-8** (without BOM) text file (with the extension "txt"). It can contain the words/sentences and an optional comment (delimited by the first "#"), e.g.:

```
mybadword#this is a bad word related to xyz
a really bad sentence#this is a really, really bad sentence!
#This is a commented line!
wordwithnocomment
```

The file can contain *any number* of words or sentences in *any case* and *order*, separated by new lines. Duplicates don't matter, so if for some reason you put in the same word twice, it will still work smoothly.

The hash-sign (#) is used to comment lines.

6.1.4. Regular expressions (RegEx)

We also support regular expressions (RegEx) for bad words/sentences. This means that you can modify the matching criteria with RegEx. You can write a resource and put the RegEx instead of "simple" bad words/sentences. Here are some actual examples:

ass:

This will only match (in "Fuzzy" mode) the word "ass" and *no word parts*, as in "p**ass**ive", "m**ass**ive", "**ass**assin" etc.:

\bass\b

step by step:

\b	marks the start of a word
ass	the word
\b	marks the end of word

deep throat:

That's a simple one, it would match "**deep   throat**", "**deepthroat**", "**deep---throat**" etc.:

deep(-|)*throat

step by step:

deep   the first word part

(-|)*   matches - or " " (space) 0-n times

throat   the second word part

arsch:

That's also a simple RegEx and will match words like "fett**arsch**", "**arsch**loch", "riesen**arsch**fresse" etc.:

\b\w*arsch\w*\b

step by step:

\b   marks the start of a word

\w*   any alphanumeric character matching 0-n times

arsch the word

\w*   any alphanumeric character matching 0-n times

\b   marks the end of word

shit:

We would like to match "mega**shit**", "**sh!+**", "**sh1z**storm", "fat**shit**ter" etc.:

\b\w*[5s]h[i!1][z+t7]\w*\b#shit

step by step:

\b   marks the start of a word

\w*   any alphanumeric character matching 0-n times

[5s]   matches 5 or s exactly once

h   the letter H

[i!1]   matches i, ! or 1 exactly once

[z+t7] matches z, +, t or 7 exactly once

\w*   any alphanumeric character matching 0-n times

\b   marks the end of word

RegEx quantifiers:

*   matches 0-n times

+   matches 1-n times

{2,n}   matches 2-n times

You can do all kinds of crazy matching stuff with RegEx and it's totally up to you what and how you match something:

http://regexlib.com/CheatSheet.aspx?AspxAutoDetectCookieSupport=1

7. API

The asset contains various classes and methods. The most important ones are explained here.

Make sure to **include** the **name space** in the relevant source files:

```
using Crosstales.BWF;
```

7.1. Managers

These are the important methods for the different managers.

7.1.1. Contains

Searches for bad words in a text and returns true if a bad word was found.

For example:

```
//check with all sources
bool isNotOk = BadWordManager.Instance.Contains("hello world");


//check with "english" and "german" as sources
bool isNotOk = BadWordManager.Instance.Contains("hello world", "english",
"german");
```

Or check it with the BWFManager:

```
//check with all managers and sources
bool isNotOk = BWFManager.Instance.Contains("hello world");


//check with the BadWordManager and all sources
bool isNotOk = BWFManager.Instance.Contains("hello world",
ManagerMask.BadWord);


//check with the BadWordManager and DomainManager and all sources
bool isNotOk = BWFManager.Instance.Contains("hello world", ManagerMask.BadWord
| ManagerMask.Domain);


//check with the BadWordManager and "english" and "german" as sources
bool isNotOk = BWFManager.Instance.Contains("hello world", ManagerMask.BadWord,
"english", "german");
```

7.1.2. **GetAll**

Searches for bad words in a text and returns a list with the found words.

For example:

```
//get with all sources
List<string> badwords = BadWordManager.Instance.GetAll("hello world");

//get with "english" and "german" as sources
List<string> badwords = BadWordManager.Instance.GetAll("hello world",
"english", "german");
```

Or get it from the BWFManager:

```
//get with all managers and sources
List<string> badwords = BWFManager.Instance.GetAll("hello world");

//get with the BadWordManager and all sources
List<string> badwords = BWFManager.Instance.GetAll("hello world",
ManagerMask.BadWord);

//get with the BadWordManager and DomainManager and all sources
List<string> badwords = BWFManager.Instance.GetAll("hello world",
ManagerMask.BadWord | ManagerMask.Domain);

//get with the BadWordManager and "english" and "german" as sources
List<string> badwords = BWFManager.Instance.GetAll("hello world",
ManagerMask.BadWord, "english", "german");
```

7.1.3. **ReplaceAll**

Searches and replaces all bad words in a text.

For example:

```
//replace with all sources
string clean = BadWordManager.Instance.ReplaceAll("hello world");
//replace with "english" and "german" as sources
string clean = BadWordManager.Instance.ReplaceAll("hello world", "english",
"german");
```

Or do it with the BWFManager:

```
//replace with all managers and sources
string clean = BWFManager.Instance.ReplaceAll("hello world");
```

```
//replace with the BadWordManager and all sources
```
string clean = BWFManager.Instance.ReplaceAll("hello world",
ManagerMask.BadWord);

```
//replace with the BadWordManager and DomainManager and all sources
```
string clean = BWFManager.Instance.ReplaceAll("hello world",
ManagerMask.BadWord | ManagerMask.Domain);

```
//replace with the BadWordManager and "english" and "german" as sources
```
string clean = BWFManager.Instance.ReplaceAll("hello world",
ManagerMask.BadWord, "english", "german");

7.1.4. **Replace**

Replaces all bad words in a text.

Use this method if you already have a list of bad words (e.g. from the 'GetAll()' method):

```
//replace a list of bad words with the BadWordManager
```
string clean = BadWordManager.Instance.Replace("hello world", badwords);

Or do it with the BWFManager:
```
//replace a list of bad words with all managers
```
string clean = BWFManager.Instance.Replace("hello world", badwords);

```
//replace a list of bad words with the BadWordManager
```
string clean = BWFManager.Instance.Replace("hello world", badwords,
ManagerMask.BadWord);

7.1.5. **Mark**

Marks the text with a prefix and postfix from a list of bad words:

```
//mark a list of bad words with the BadWordManager and the editor settings
```
string marked = BadWordManager.Instance.Mark("hello world", badwords);

```
//mark a list of bad words with the BadWordManager as "bold"
```
string marked = BadWordManager.Instance.Mark("hello world", badwords, "",
"");

Or do it with the BWFManager:
```
//mark a list of bad words with the BWFManager and the editor settings
```
string marked = BWFManager.Instance.Mark("hello world", badwords);

```
//mark a list of bad words with the BWFManager as "bold"
string marked = BWFManager.Instance.Mark("hello world", badwords, "<b>",
"</b>");
```

7.1.6. **Unmark**

Unmarks the text with a prefix and postfix.

For example:

```
//unmark a text with the BadWordManager and the editor settings
string unmarked = BadWordManager.Instance.Unmark("hello world");
```

```
//unmark a text with the BadWordManager and "bold" markings
string unmarked = BadWordManager.Instance.Unmark("hello world", "<b>", "</b>");
```

Or do it with the BWFManager:
```
// unmark a text with the BWFManager and the editor settings
string marked = BWFManager.Instance.Unmark("hello world");
```

```
//unmark a text with the BWFManager and "bold" markings
string marked = BWFManager.Instance.Unmark("hello world", "<b>", "</b>");
```

7.1.7. **Check if the managers are ready**

At the start-up of the "Bad Word Filter", all active managers must read and prepare the sources. This takes some time; to check if a manager is ready, all managers implement the isReady-property. Check it like this:

```
public IEnumerator myFunction() {
      while(!BWFManager.Instance.isReady) {
            yield return null;
      }

      //do your stuff
}
```

Or use the callback "OnBWFReady" from the BWFManager.

7.1.8. **Get all sources**

To get an alphabetically ordered list of all sources from a manager, use it like this:

```
List<Source> src = BadWordManager.Instance.Sources;
```

Or get it the BWFManager:
```
List<Source> src = BWFManager.Instance.Sources(ManagerMask.Badword);
```

7.1.9. **Change the replace characters**

This is only available for *BadWordFilter* and *DomainFilter*. To change the replace characters during run-time, do it like this:

```
BadWordManager.Instance.ReplaceChars = "?#@*&%!$";
DomainManager.Instance.ReplaceChars = "?#@*&%!$";
```

7.1.10. **Change the character number**

This is only available for *CapitalizationFilter* and *PunctuationFilter*. To change the character number during run-time, do it like this:

```
CapitalizationManager.Instance.CapitalizationCharsNumber = 5;
PunctuationManager.Instance.PunctuationCharsNumber = 6;
```

7.1.11. **Change the mode**

This is only available for *BadWordManager*. To change the mode during run-time, do it like this:

```
BadWordManager.Instance.Mode = Model.Enum.ReplaceMode.LeetSpeakAdvanced;
```

Available modes:
- `Default`
- `NonLettersOrDigits`
- `LeetSpeak`
- `LeetSpeakAdvanced`

7.1.12. Change to SimpleCheck-mode

This is only available for *BadWordManager*. To change the SimpleCheck-mode during run-time, do it like this:

```
BadWordManager.Instance.SimpleCheck = true; //Chinese, Japanese, Korean or Thai
```

7.2. Complete API

For more details, please see the BadWordFilter-api.pdf

8. Third-party support (PlayMaker etc.)

"Bad Word Filter" supports various assets from other publishers. Please import the desired packages from "Assets/Plugins/crosstales/BadWordFilter/3rd party".

9. Verify installation

Check if BWF is installed:

```
#if CT_BWF
     Debug.Log("BWF installed: " + Util.Constants.ASSET_VERSION);
#else
     Debug.LogWarning("BWF NOT installed!");
#endif
```

10. Upgrade to new version

Follow this steps to upgrade the version of "Bad Word Filter PRO":

1. Update "Bad Word Filter PRO" to the latest version from the "Unity AssetStore"
2. Inside the project in Unity, go to menu "File" => "New Scene"
3. Delete the "Assets/Plugins/crosstales/BadWordFilter" folder from the Project-view
4. Import the latest version downloaded from the "Unity AssetStore"

11.Important notes

After this setup, the "Bad Word Filter" is ready to use. It is important to know that it uses the **singleton**-pattern, which means that **once instantiated**, the "Bad Word Filter" will **live until** the application is **terminated**.

- Add only the **needed sources** (e.g. a product for "English"-speaking users maybe only needs "en" instead of all other unused languages).

- Perform only the **checks** you **need** (e.g. if excessive punctuation isn't an issue, don't check for it)

12. Problems, missing words, languages etc.

If you encounter any problems with this asset, just send us an email with a problem description, the Unity version and the invoice number and we will try to solve it.

We will add more bad words, languages and features over time.

If you miss some words or even an entire language, feel free to send us the data. Unfortunately, we don't speak every language on this beautiful planet, but we want to build the best bad word filter available and appreciate your effort to help us approach this goal.

If you send us some additional bad words, we'd appreciate it if you'd also include a description (in English or in German) of what the word means.

13. Release notes

See "VERSIONS.txt" under "Assets/Plugins/crosstales/BadWordFilter/Documentation" or online:

https://crosstales.com/media/data/assets/badwordfilter/VERSIONS.txt

14. Credits

"Bad Word Filter" contains some words from the following sources:

Wikipedia   http://en.wikipedia.org/wiki/Category:Profanity_by_language

Shutterstock   https://github.com/shutterstock/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words

The icons are based on:

[Open Icon Library](#)

[Font Awesome](#)

Special thanks to our contributing users:

Arabic:	Hadi Reda
Greek:	Nikiforos Papagiannopoulos
Italian:	Alessio Romagnolo
Japanese:	Yoko Vögeli
Russian:	Oleksandr Voropaiev
Spanish:	Sergio Silva
Swedish:	Christian Engvall
Vietnamese:	Tuan Nguyen

15. Contact and further information

crosstales LLC
Schanzeneggstrasse 1
CH-8002 Zürich

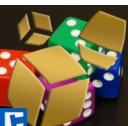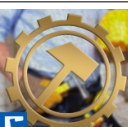|	
Homepage:	https://www.crosstales.com/en/portfolio/badwordfilter/
Email:	bwf@crosstales.com
AssetStore:	https://assetstore.unity.com/lists/crosstales-42213
Forum:	https://forum.unity.com/threads/bad-word-filter-pro-solution-against-profanity-and-obscenity.289960/
Documentation:	https://www.crosstales.com/media/data/assets/badwordfilter/BadWordFilter-doc.pdf
API:	https://www.crosstales.com/en/assets/badwordfilter/api
WebGL-Demo:	https://www.crosstales.com/media/data/assets/badwordfilter/webgl/

16. Our other assets

3D Skybox	Those beautiful packages contain professional 8k, HDR, stereoscopic 360° real-world skyboxes for your projects.
DJ	DJ is a player for external music-files. It allows a user to play his own sound inside any Unity-app. It can also read ID3-tags.
File Browser	File Browser is a wrapper for native file dialogs on Windows, macOS, Linux and UWP (WSA).
Online Check	You need a reliable solution to check for **Internet availability**? Here it is!
Radio	Radio allows implementing free music from Internet radio stations into your project..
RT-Voice	RT-Voice uses the computer's (already implemented) TTS (text-to-speech) voices to turn the written lines into speech and dialogue at run-time! Therefore, all text in your game/app can be spoken out loud to the player.
True Random	True Random can generate "true random" numbers for you and your application. The randomness comes from atmospheric noise, which for many purposes is better than the pseudo-random number algorithms typically used in computer programs.
Turbo Backup	Turbo Backup is the fastest and safest way to backup your Unity project. It only stores the difference between the last backup, this makes it incredible fast.
Turbo Builder	Turbo Builder creates builds for multiple platforms in one click. It works together with Turbo Switch to offer an incredible fast build pipeline.

Turbo Switch	Turbo Switch is a Unity editor extension to reduce the time for assets to import during platform switches. We measured speed improvements up to 100x faster than the built-in switch in Unity.