Bad Word Filter PRO

Keep your games civilized

Documentation

crosstales LLC

Date: 22. October 2016

Version: 2.7.2

Table of Contents

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

If you have any questions about this asset, send an email to assets@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 does 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 **23 languages** (bad words as **regular expressions** which match **ten thousands** of word **variations**):

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

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

- **Check, get, replace** and **mark** functions for:
    - **bad words** and **sentences**
    - domains, **URLs** and **emails**
    - excessive **capitalization**
    - excessive **punctuation**
- All four filters can be used **separately** and changed during **run-time**!
- Includes over **4'000 regular expressions** of bad words in **23 languages** (matches **ten thousands** of word **variations**)!
- Works with **any language** and **writing system** (incl. support for ltr/rtl) like Arabic, Cyrillic, Chinese, Japanese etc.
- **Customizable, extensible** and **easy modifications/additions** to the existing sources ("bad words").
- Powerful **API** to get **maximum control** as a developer.
- **Multit-threaded** and **blazing fast** even with **thousands** of words!
- Automatic (simple) **pluralization** of English words.
- Pre-configured providers for **resources**, **files** and **URLs.** Deliver the sources (e.g. "bad words") your way or implement your own **provider** (e.g. for XML, JSON)!
- **Meta data** for sources, like a **description** and **icon.**
- Developed for **Unity 5**
- Runs on **all Unity platforms** (extensively tested under Standalone, Webplayer, WebGL, iOS, Windows Phone and Android).
- **PlayMaker** actions!
- **Test-Drive** the bad word detection inside the **Editor**!
- Extensive **demo** scene, **tests**, **documentation**, **API** and **support**!
- Full C# **source code** and **all sources** ("bad words") provided.
- We are **committed** to all our assets! This means, we will add new bad words, languages and features over time!

3. Demonstration/Test

The asset comes with a demo and some test scenes to show the main usage.

The demo scene is located under "Assets/crosstales/BadWordFilter/Demo" and the test scenes reside under "Assets/crosstales/BadWordFilter/Test".

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 "OnGUI"- or "Update"-methods). Use check/replace intervals of **100ms or more** (see "GUIMain.cs" for an example implementation).

4. Setup

BWF has global settings under "Edit\Preferences..." and under "Tools\Bad Word Filter\Configuration...".

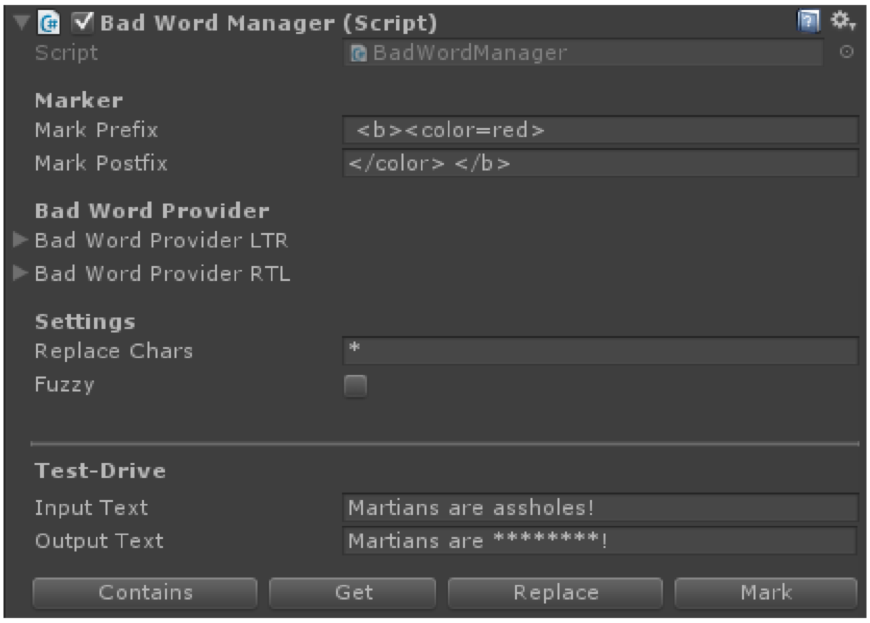
The "Bad Word Filter" consist of four separate parts which all can be used independently:

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

If you like to use the whole functionality, please use the **BWFManager** instead.

4.1. BadWordManager

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



Bad Word Manager (Script)

Script	BadWordManager
Marker	
Mark Prefix	`<b><color=red>`
Mark Postfix	`</color></b>`
Bad Word Provider	
Bad Word Provider LTR	
Bad Word Provider RTL	
Settings	
Replace Chars	*
Fuzzy	☐
Test-Drive	
Input Text	Martians are assholes!
Output Text	Martians are ********!

4.1.1. Mark Prefix and Postfix

These are the markers for detected **bad words** in a string. The default settings simply mark the words bold and red (e.g. for rich-text-components).

You can define your own prefix and postfix.

4.1.2. Bad Word Provider LTR

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

You can add as many providers as you want.

For more information about providers, please see below.

4.1.3. Bad Word Provider RTL

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

You can add as many files as you want.

For more information about providers, please see below.

4.1.4. Replace Chars

These are the desired **1-n replace characters** which were used to replace bad words/sentences. If you have multiple replace characters, a random string will be generated for every bad word or sentence.

4.1.5. Fuzzy

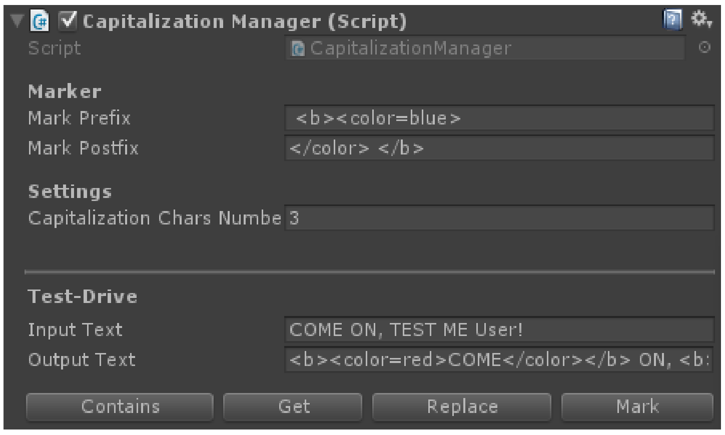
“Fuzzy” defines **how exact** the match will be. Without fuzziness, only exact matches are detected.

It depends on how you want to use the “BadWordManager” and how much time you would spend in creating all word variations/regular expressions.

Important: “Fuzzy” is much **more performance consuming** – so be careful!

4.2. CapitalizationManager

The CapitalizationManager is the main component for filtering excessive **capitalization** in strings.


Capitalization Manager (Script)

Script	CapitalizationManager

Marker

Mark Prefix	<b><color=blue>
Mark Postfix	</color> </b>

Settings

Capitalization Chars Numbe	3

Test-Drive

Input Text	COME ON, TEST ME User!
Output Text	<b><color=red>COME</color></b> ON, <b:

4.2.1. Mark Prefix and Postfix

These are the markers for detected excessive **capitalization** in a string. The default settings simply mark the letters bold and red (e.g. for rich-text-components).

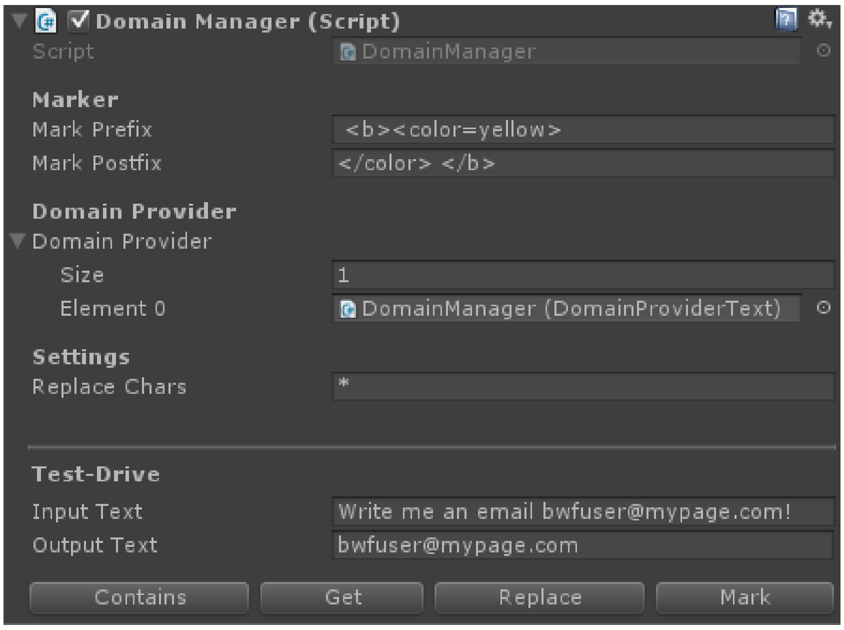
You can define your own prefix and postfix.

4.2.2. Capitalization Chars Number

Defines the number of allowed capital letters in a row.

4.3. DomainManager

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



Domain Manager (Script)

Script	DomainManager
Marker	
Mark Prefix	<b><color=yellow>
Mark Postfix	</color> </b>
Domain Provider	
Domain Provider	
Size	1
Element 0	DomainManager (DomainProviderText)
Settings	
Replace Chars	*
Test-Drive	
Input Text	Write me an email bwfuser@mypage.com!
Output Text	bwfuser@mypage.com

4.3.1. Mark Prefix and Postfix

These are the markers for detected domains, **URLs** and **emails** in a string. The default settings simply mark the words bold and red (e.g. for rich-text-components).

You can define your own prefix and postfix.

4.3.2. Domain Provider

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

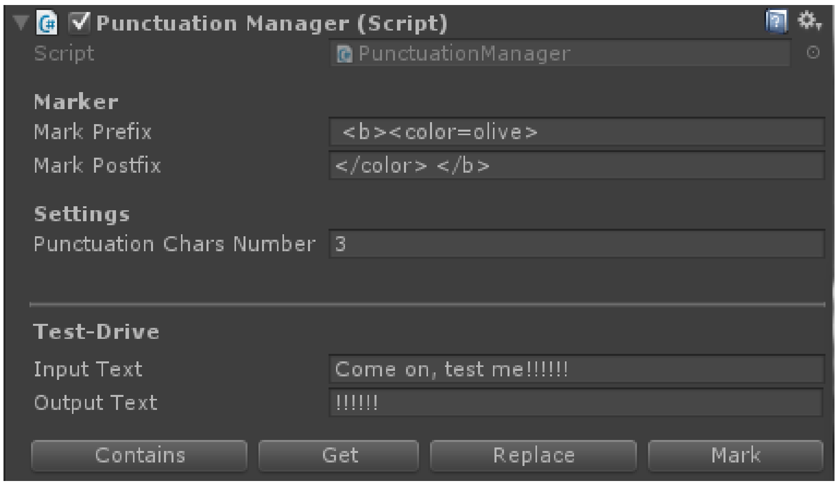
You can add as many providers as you want.

4.3.3. Replace Chars

These are the desired **1-n replace characters** which were used to replace domains, URLs and emails. If you have multiple replace characters, a random string will be generated for every bad word or sentence.

4.4. PunctuationManager

The PunctuationManager is the main component for filtering excessive **punctuation** in strings.


Punctuation Manager (Script)
Script: PunctuationManager

Marker
Mark Prefix: <b><color=olive>
Mark Postfix: </color></b>

Settings
Punctuation Chars Number: 3

Test-Drive
Input Text: Come on, test me!!!!!!
Output Text: !!!!!!

4.4.1. Mark Prefix and Postfix

These are the markers for detected excessive **punctuation** in a string. The default settings simply mark the letters bold and red (e.g. for rich-text-components).

You can define your own prefix and postfix.

4.4.2. Punctuation Chars Number

Defines the number of allowed punctuation letters in a row.

4.5. BWFManager

The BWFManager simply unites all available managers to a “single-point-of-entry”. Normally, BWFManager is added automatically to your scene. To add it manually to your project, there are two possibilities:

1. Add the prefab **BWF** to the scene
2. Or go to *Tools => BadWordFilter => Add => **BWF***

The configuration of the individual managers must be done separately (see previous pages).

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:

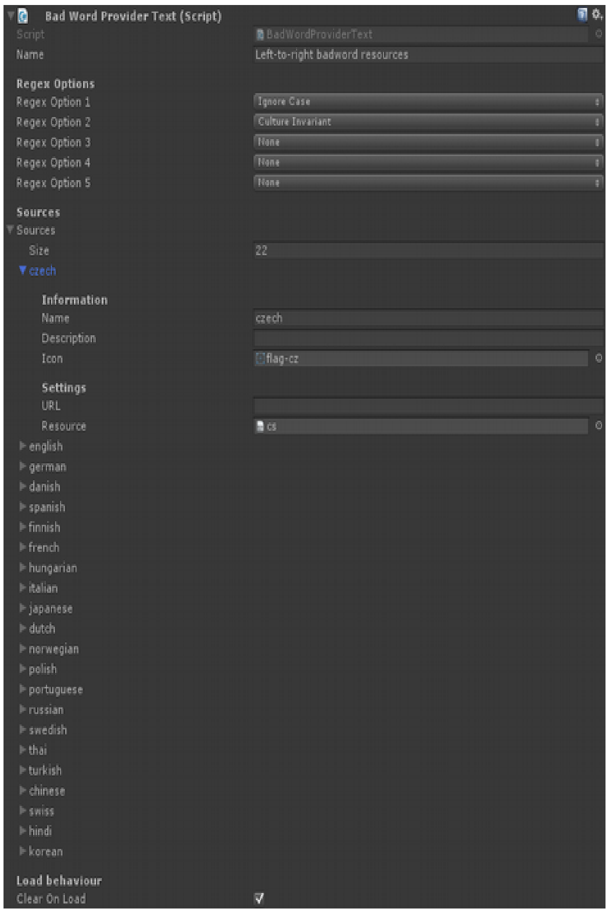
1. Unity resources
2. Accessing files on the local machine
3. Accessing files on a web server

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:



Bad Word Provider Text (Script)

Script	BadWordProviderText
Name	Left-to-right badword resources
Regex Options	
Regex Option 1	Ignore Case
Regex Option 2	Culture Invariant
Regex Option 3	None
Regex Option 4	None
Regex Option 5	None
Sources	
Sources	
Size	22
czech	
Information	
Name	czech
Description	
Icon	flag-cz
Settings	
URL	
Resource	cs
english	
german	
danish	
spanish	
finnish	
french	
hungarian	
italian	
japanese	
dutch	
norwegian	
polish	
portuguese	
russian	
swedish	
thai	
turkish	
chinese	
swiss	
hindi	
korean	
Load behaviour	
Clear On Load	✓

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

This 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	Local or remote text file containing all regular expressions for this source (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
ar.txt	Arabic
cs.txt	Czech
da.txt	Danish
de.txt	German
de_ch.txt	Swiss-German
Emoji.txt	Emojis and miscellaneous symbols
en.txt	English
es.txt	Spanish
fa.txt	Persian
fi.txt	Finnish
fr.txt	French

global.txt	Global forbidden words
hi.txt	Hindi
hu.txt	Hungarian
it.txt	Italian
ja.txt	Japanese
ko.txt	Korean
names.txt	Forbidden user names (from games, movies, comics etc.)
nl.txt	Dutch
no.txt	Norwegian
pl.txt	Polish
pt.txt	Portuguese
ru.txt	Russian
sv.txt	Swedish
th.txt	Thai
tr.txt	Turkish
zh.txt	Chinese

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

<u>shit:</u>

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
```

<u>step by step:</u>

\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

<u>RegEx quantifiers:</u>

*	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 "Bad Word Filter" consists of a powerful API for run-time access.

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

```
using Crosstales.BWF;
```

7.1. Managers

These are the 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.Contains("hello world");

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

Or check it with the BWFManager:

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

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

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

//check with the BadWordManager and "english" and "german" as sources
bool isNotOk = BWFManager.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.GetAll("hello world");
```

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

Or get it from the BWFManager:

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

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

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

//get with the BadWordManager and "english" and "german" as sources
List<string> badwords = BWFManager.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.ReplaceAll("hello world");
//replace with "english" and "german" as sources
string clean = BadWordManager.ReplaceAll("hello world", "english", "german");
```

Or do it with the BWFManager:

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

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

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

//replace with the BadWordManager and "english" and "german" as sources
string clean = BWFManager.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.Replace("hello world", badwords);
```

Or do it with the BWFManager:

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

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

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

Or do it with the BWFManager:

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

//mark a list of bad words with the BWFManager as “bold”
string marked = BWFManager.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.Unmark("hello world");

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

Or do it with the BWFManager:

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

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

7.1.7. Get the filter

You can get the filter for every manager like this:

```
BadWordFilter bwf = BadWordManager.Filter;
CapitalizationFilter cf = CapitalizationManager.Filter;
DomainFilter df = DomainManager.Filter;
PunctuationFilter pf = PunctuationManager.Filter;
```

Or get it from the BWFManager:

```
BadWordFilter bwf = ((BadWordFilter)BWFManager.Filter(ManagerMask.BadWord)));
CapitalizationFilter cf =
((CapitalizationFilter)BWFManager.Filter(ManagerMask.Capitalization)));
DomainFilter df = ((DomainFilter)BWFManager.Filter(ManagerMask.Domain)));
PunctuationFilter pf =
((PunctuationFilter)BWFManager.Filter(ManagerMask.Punctuation)));
```

7.1.8. 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 and to check if a manager is ready, all managers implement the isReady-property. Check it like this:

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

   //do your stuff
}
```

7.1.9. Get all sources

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

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

Or get it the BWFManager:

```
List<Source> src = BWFManager.Sources(ManagerMask.BadWord);
```

7.1.10. Change mark/unmark prefix and postfix

To change the prefix and postfix for the mark/unmark function during run-time, do it like this:

```
BadWordManager.Filter.MarkPrefix = "<color=blue>";
BadWordManager.Filter.MarkPostfix = "</color>";
```

Or do it with the BWFManager:

```
((BadWordFilter)BWFManager.Filter(ManagerMask.BadWord))).MarkPrefix = 
"<color=blue>";
((BadWordFilter)BWFManager.Filter(ManagerMask.BadWord))).MarkPostfix = 
"</color>";
```

7.1.11. 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.Filter.ReplaceCharacters = "?#@*&%!$";
DomainManager.Filter.ReplaceCharacters = "?#@*&%!$";
```

Or do it with the BWFManager:

```
((BadWordFilter)BWFManager.Filter(ManagerMask.BadWord))).ReplaceCharacters = 
"?#@*&%!$";
((DomainFilter)BWFManager.Filter(ManagerMask.Domain))).ReplaceCharacters = 
"?#@*&%!$";
```

7.1.12. 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.Filter.CharacterNumber = 5;
PunctuationManager.Filter.CharacterNumber = 6;
```

Or do it with the BWFManager:

```
((CapitalizationFilter)BWFManager.Filter(ManagerMask.Capitalization))).Characte
rNumber = 5;
((PunctuationFilter)BWFManager.Filter(ManagerMask.Punctuation))).CharacterNumbe
r = 6;
```

7.1.13. Change fuzzy mode

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

```
BadWordManager.Filter.Fuzzy = true;
```

Or do it with the BWFManager:

```
((BadWordFilter)BWFManager.Filter(ManagerMask.BadWord))).Fuzzy = true;
```

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 the „3rd party“-folder.

9. Upgrade to new version

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

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

10. 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**.

- **Instantiate** it inside your **first scene** or when you need it. This is normally automatically done.
- Add only the **sources** you **need** (e.g. if you create a product for "English"-speaking users, only add "en" instead of all other 20 "useless" languages).
- Perform only the **checks** you **need** (e.g. if excessive punctuation isn't an issue, don't check for it)

Remember: it must be instantiated before you try to access it! Otherwise it's not possible to test strings because no sources were loaded.

11. 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.

12. Release notes

See "README.txt"

13. 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

Some icons are from http://sourceforge.net/projects/openiconlibrary/

14. Contact and further information

crosstales LLC
Weberstrasse 21
CH-8004 Zürich

Homepage:	http://www.crosstales.com/en/assets/badwordfilter/
Email:	assets@crosstales.com
AssetStore:	https://www.assetstore.unity3d.com/en/#!/content/26255
Forum:	http://forum.unity3d.com/threads/bad-word-filter-aka-profanity-or-obscenity-filter.289960/
Documentation:	http://www.crosstales.com/en/assets/badwordfilter/BadWordFilter-doc.pdf
API:	http://www.crosstales.com/en/assets/badwordfilter/api
Web-Demo:	http://www.crosstales.com/en/assets/badwordfilter/run-webgl.html

15. Our other products

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.
Radio	Have you ever wanted to implement radio stations but don't want (or can't) pay an horrendous amount of money? Whenever you like to provide good sound from famous artists for your games or apps, tune in on one of the uncountable Internet MP3/OGG radio stations available for free.
RTVoice	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.
TPS	Turbo Platform Switch is a Unity editor extension to reduce the time for assets to import during platform switches. We measured speed improvements up to 50x faster than the built-in switch in Unity.