# KSP Localizer

Contents

[KSP Localizer 1](#_Toc199667703)

[Features 1](#_Toc199667704)

[Installation 1](#_Toc199667705)

[Usage Instructions 2](#_Toc199667706)

[INI file 3](#_Toc199667707)

[Reverts and cleanup 3](#_Toc199667708)

[Notes 3](#_Toc199667709)

KSP was localized in version 1.3. For a game with so many mods, it was disappointing to see the game localized without any tools to assist mod authors to do so.

There have been at least two utilities written over the years to address this. The first, written by @Dr. jet, is a python script that only localizes part file, science descriptions and ModuleManager patches. The second, written by @IgnorZ, is more ambitious, in that it works as a mod, and does more, reading the game database to extract strings. However, it also does not touch any CSharp code in any of the mods.

This is my version of a KSP Localization tool. I hope people find it useful.

## Features

* Intended for use by mod authors
* This tool scans `.cs` and `.cfg` files for localizable strings and generates:
  + `Localization/en-us.cfg` – for use in KSP
  + `Localization/en-us.csv` – for easy editing in spreadsheet tools
* It updates the source files with the localization tag.
* The original files are backed up to <file>.bak file for easy reverts
* Sections of code can be ignored using a named region
* This is run as a command line script. It’s written in CSharp, While it may be possible to run it on Linux or MacOS using the dotnet command, I haven’t tested it there; there is a shell script which should work.
* Writes out both CFG and CSV files; the CSV is supplied to make it easier for other people to work on translations in a spreadsheet.

## Installation

Download release zip file and unpack it somewhere. The zip file will contain a folder called KSPLocalizer. The entire folder should be copied locally.

There is a small utility CMD file **localizer.cmd** in the directory. This is intended to make it easier to call the script. If on Linux or MacOS, use the file **localizer.sh**. These files are intended to make it easier to use without having to change the PATH variable.

Edit the appropriate script file and set the path to the program.

## Usage Instructions

The program can be called directly (obviating the need for the shell script), or the shell script can be copied to wherever you want to run it from. I would NOT copying it to the top directory/folder of the mod.

The basic syntax of the command line follows. The only required line items are the mod directory and the prefix, all other options are optional:

localizer.cmd <ModDirectory> --prefix=<prefix> [ optional options> ]

--outdir=<path> Output path for `en-us.cfg` and en-us.csv files

--prefix=<string> Localization key prefix (default: `MyMod\_`)

--maxkeylength=<number> Maximum length for localization keys (default: 25)

--numerictags Use a sequential number for the tags

--separatePartsCfg Create a file for the part tags and one file for the code tags

--inifile=<file> Specify a ini file which contains include and exclude strings (example below

--csonly Only process .cs files

--cfgonly Only process .cfg files

--revert Restores the original files from the .bak files

--cleanbak Deletes all the .bak files

--help Display help

An example command line would look something like this.

Windows:

.\localizer.cmd L-Tech --prefix=LOC\_lTech --outdir=L-Tech/GameData/LTech/Localization --numerictags

Linux and MacOS:

./localizer.sh L-Tech --prefix=LOC\_lTech --outdir=L-Tech/GameData/LTech/Localization --numerictags

This is a command to localize a mod called L-Tech. It is run in the directory above L-Tech. The prefix to the tags will be LOC\_lTech, and the output files will be put into the directory L-Tech/GameData/LTech/Localization

This will:

* Scan files
* Generate localization entries
* Write modified copies of `.cs` and `.cfg` files
* Output `Localization/en-us.cfg` and `en-us.csv`

## INI file

An ini file is included which contains a few common strings to be ignored. It can also exclude and/or include specific files. The strings can be either simple strings or regular expressions. Both strings and files can be excluded, only files can be included. Full documentation for the file is at the head of the file. Multiple ini files can be used by specifying the --inifile=<file> to use your own file in addition to the stock one. Multiple ini files are allowed, the entries are additive.

Review the supplied ini file to understand what might be excluded. For example, “const string” is excluded since string constants have to be literal.

## Reverts and cleanup

These two options override all others. In other words, if either of them is on the command line in addition to any other options, the revert/cleanbak functions will be run instead.

## Notes

While pre-existing localized tags are not treated as strings, there is no mechanism to check for duplicates with any pre-existing tags. If the mod does have some pre-existing tags, the safest thing to do is just use a different prefix.

Pre-existing localization files may be overwritten if the names are the same.

Another quirk is the way KSP deals with Event strings. While normally code needs to call Localizer.Format to get the localized string, there is no need to do this for events since the event code itself will be calling the localizer. It's almost impossible to detect this, so this edge case is not dealt with.

Other than a tiny bit of performance loss due to the double call of Localizer.Format, there should be no change in functionality.

This is something the user will need to look at during the conversion process.

When running this on a C-Sharp code-based mod, there is no way to determine if a string is meant for internal use or not. This is where the --revert option and the NO\_LOCALIZATION region\_ comes in handy. Simply add the --revert to the existing line and run to undo all the changes. There are two tools available to fix this:

1. Use the   
     
   #region NO\_LOCALIZATION  
   ….  
   #endregion
2. to mark sections of code which should not be localized
3. Attributes are a trickier issue, since they require constants. The code should recognize attributes, but in the event it doesn’t in some cases, there is a way to define a region as attributes. You can use:

#region ATTRIBUTE\_LOCALIZATION

....

#endregion

1. Create a new ini file with your desired entries, then, use the  
    --inifile=inifile.ini to include it