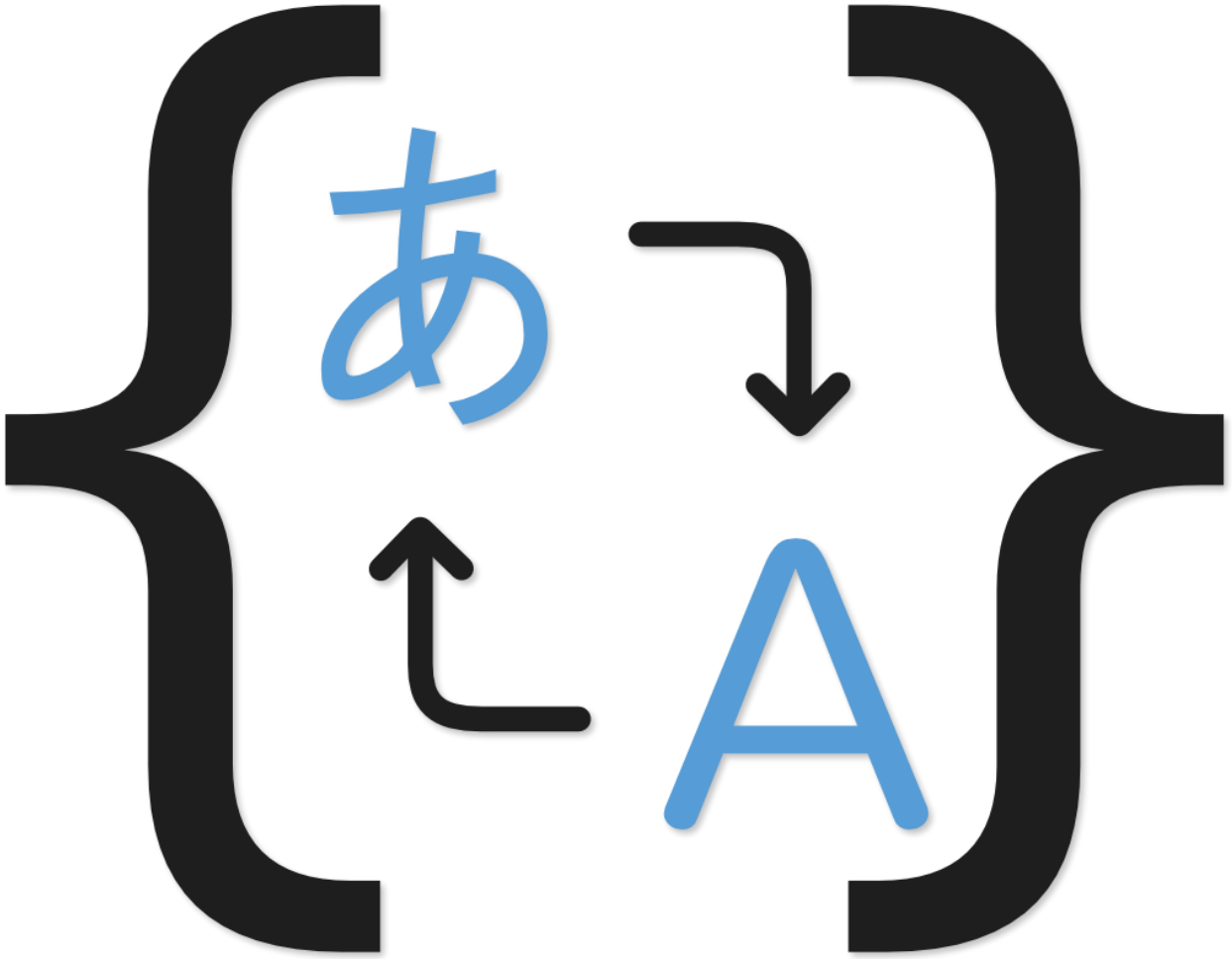
fabian.kober

JLoc Language Manager Documentation



Thank you for purchasing my JLoc JSON based Language Manager! On the following pages you will find a manual on how to use it as well as an API documentation.

Manual

The Basics

JLoc is short for JSON Localization, as this Language Manager works using JSON files residing in the Streaming Assets for all strings. This allows for direct editing of the files without the included editor as JSON is human readable, as well as changing out the string files in a build (because the Streaming Assets folder is just copied to the build location).

The string files have the following structure:

- Category1
    - Field1
        - Language1
        - Language2
        - …
    - Field2
        - Language1
        - Language2
        - …
    - …
- Category2
- -…

JLoc consist of four parts:

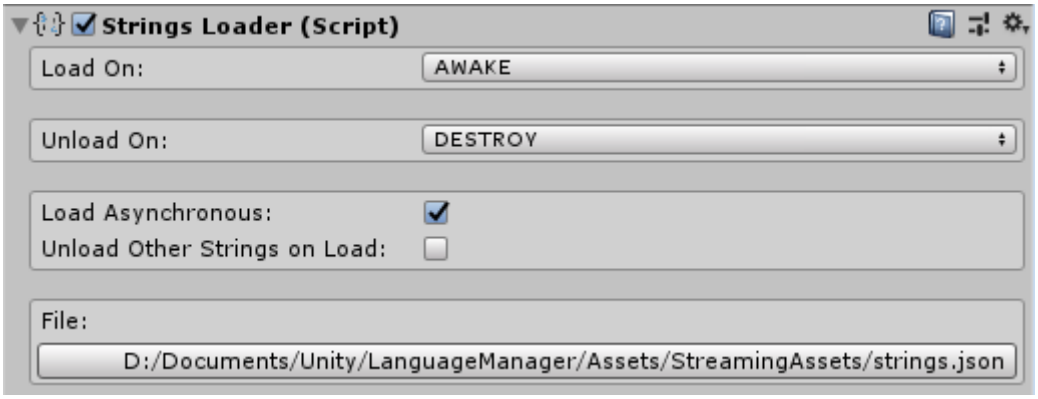
- The **Language Manager** which manages loading and unloading of string files, providing the strings in the proper language and changing the language
- The **Strings Loader** component which tells the Language Manager which file to load or unload
- The **Language Text Config** component which makes a Unity Text or Text Mesh Pro Text language dependent
- The **JLoc Editor** for setting up the Language Manager and creating and editing string files

If you are not using TextMeshPro in your Project and you are getting compiler errors, you need to comment out the first line in the Language Manager and the Language Text Config Scripts.

Language Manager

The Language Manager is the core of JLoc. It is a static class, which makes it scene independent (which means that it won't lose any data when the scene is changed). You can set up the settings for it via the Language Manager Editor. All other interaction with it happens via the other components in this package or via the API.

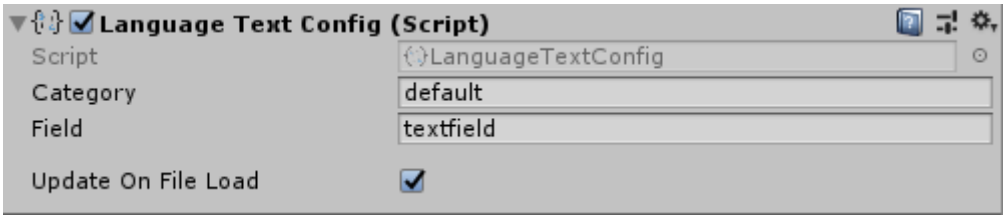
Strings Loader



This component manages loading and unloading strings files.

Property	Function
Load On	Defines the Unity Event function on which the specified file is loaded
Unload On	Defines the Unity Event function on which the specified file is unloaded (Set to none if the file should stay loaded, you can still manually unload it via the API)
Load Asynchronous	If true, the file is loaded in a separate thread.
Unload Other Strings on Load	If true, all files that are loaded when this loader starts loading will be unloaded prior to loading
File	The file inside the Streaming Assets file that will be loaded

Language Text Config



This component makes any Unity Text or Text Mesh Pro Text language dependent. It will automatically try to load the string specified via category and field in the currently set language. Whenever a new Language is set, it will update itself.

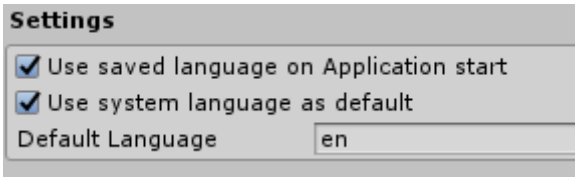
Property	Function
Category	The category inside the string files where the

	string of this text resides
Field	The field inside the category in the string files where the string of this text resides
Update on File Load	If true, this text will update any time a string file is loaded (This might be helpful if you are loading multiple files and the string for this one is in a file you are loading later)

JLoc Editor

The Language Manager Editor is the editor interface to JLoc. You can open it via *Window/JLoc Editor*. With it, you can change the settings of the Language Manager, manage available Languages and create and edit string files. It even has an option to generate a strings file with placeholders automatically.

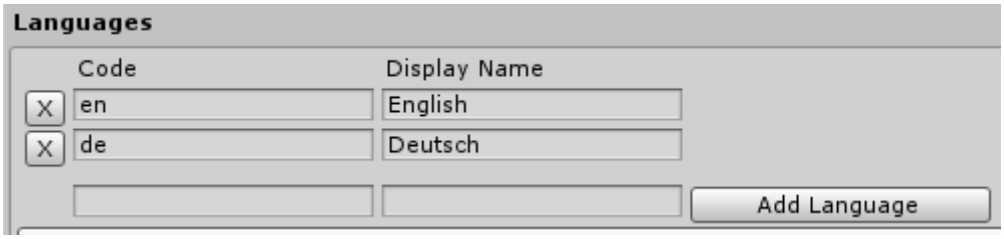
Settings



Here you can set the general settings of the Language Manager.

Property	Function
Use saved language on Application start	If this is set, the Language Manager will store the set language in the Player Prefs every time it is changed and will attempt to load the language saved in the Player Prefs at the next Application start. It is recommended you use this so the user does not have to set the language every time he starts the Application
Use system language as default	With this set, as long as there is no saved language the Language Manager will set the system language as current language, if it exists in the strings files. Otherwise the default language will be used
Default Language	Code of the language that should be used as a default when no other options apply. You should make sure that this language actually exists in your strings

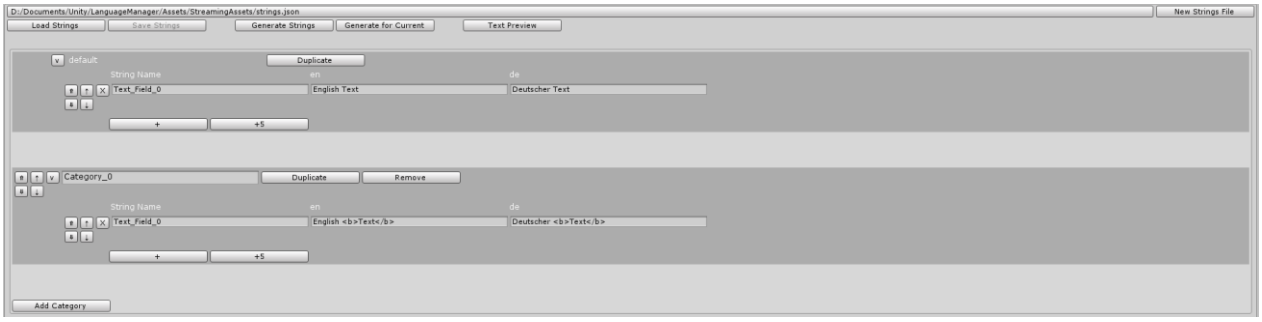
Languages



Here you can define which languages should be available for your application. These are global, so all string files contain exactly these languages and no more or less.

Property	Function
Code	Language Code of the Language. You should use the official codes here
Display Name	Display Name of the Language if it should be displayed anywhere with something other than its Code

Strings



This is where you can create and edit all the strings for your application. At the top you have some options, at the bottom you see the contents of the strings file which you can edit.

You can open an existing strings file by pressing the big button at the top. This will open an explorer window where you can load a file from inside the Streaming Assets path of your project. With the Button next to it, you can create a new file.

Below that, you have a few more buttons:

Button	Function
Load Strings	Pressing this reloads the current strings file which you can see the path of in the button where you selected it
Save Strings	This Button only exists when a file is currently loaded. It saves your changes to the file. If there are no changes to save, it is grayed out.
Generate Strings	By pressing this, you can create placeholders for all texts with a Language Text Config component inside the currently loaded strings file. Be aware that this will open all scenes in your project one by one and look for Language Text Config components in them!

Generate for Current	This will create placeholders for all texts with a Language Text Config component in the currently opened scene inside the currently loaded strings file.
Text Preview	This provides a window where you can see the text of the field you are editing in rich text format

In the area below this, you see all categories inside this strings file. You can add new ones, move them up or down, rename them, duplicate them or delete them.

If you expand them you will see all fields inside the category which consist of a name and then fields for all languages set in the Languages section. Again, you can move them up or down or delete them.

Be aware that strings are looked up by category → Field Name. No two fields in one category should have the same name and no two categories should share the name. As files can be loaded additively, if there are two files with the same category, these categories should not contain fields of the same name either if you are planning to load them together.

Other Included Tools

This package contains a few tools I programmed which it needs in order to operate.

One is a fully functional runtime C# JSON Object which can save and load to and from files both synchronously and asynchronously.

The other one is a Coroutine Host which can be used via a singleton and allows the use of coroutines on non MonoBehavior objects as well as tracking coroutines.

It would be too much to add their API to this documentation, but the Code should be enough to get started using them.

API

Language Manager

Properties

Property	Function
static bool Initialized	True once the Language Manager loaded its config and is fully initialized. Read-only
static bool HasStrings	True if at least one string file is fully loaded. Read-only
static bool CurrentlyLoadingStrings	True if the Manager is currently loading string files. Read-only
static string CurrentLanguage	Returns the code of the currently set lan-

	guage. Read-only
static string[] Languages	Returns an array of all language codes found in the config. Read-only
static string SystemLanguage	Returns the code of the system language

Callbacks

Callback	Function
static Action<string> OnLanguageChanged	Called every time the current Language is updated. It contains the language code as a parameter
static Action OnStringFileLoaded	Called every time a strings file is done loading

Public Vars

Variable	Function
const string DEFAULT_CATEGORY	Name of the default category that is present in every string file created via the JLoc Editor
const string LANGUAGES_KEY	Config key of the Language list. Used internally.
const string CONFIG_NAME	Name of the Config File. Used internally.
const string CONFIG_USE_SYSTEM_LANG_DEFAULT_KEY	Config key for the system language option. Used internally.
const string CONFIG_DEFAULT_LANG_KEY	Config key for the default language. Used internally.
const string CONFIG_USE_SAVED_LANG_KEY	Config key for the language saving option. Used internally.

Public Functions

static void LoadStringsFile(string path, bool async = true, bool unloadOther = false)

Loads a strings file

Paramenter	Function
path	Absolute path to the strings file
async	If true, the file will be loaded asynchronously in a separate thread
unloadOther	If true, all other loaded files will be unloaded prior to loading

static void ParseStrings(JSONObject strings)

Replaces escaped characters by their actual values (Line breaks, \, “, \\”). Used internally

Paramenter	Function
strings	Json object of a loaded strings file

static void UnloadStringsFile(string path)

Unloads a strings file

Paramenter	Function
path	Absolute path to the strings file

static void SetLanguage(string language)

Sets the provided language, invoking the OnLanguageChanged callback.

Paramenter	Function
language	Code of the language that should be set

static void SetText(Text textField, string name, string category = DEFAULT_CATEGORY)

Sets the text of a Unity Text field in the current language.

Paramenter	Function
textField	The Unity Text to set
name	The name of the field where the strings for this text reside
category	The category of the field for this text

static void SetText(Text textField, string name, string category, string language)

Sets the text of a Unity Text field in the provided language.

Paramenter	Function
textField	The Unity Text to set
name	The name of the field where the strings for this text reside
category	The category of the field for this text
language	Code of the language the text should be in

static void SetText(TMP_Text textField, string name, string category = DEFAULT_CATEGORY)

Sets the text of a Text Mesh Pro Text field in the current language.

Paramenter	Function
textField	The Text Mesh Pro Text to set
name	The name of the field where the strings for this text reside
category	The category of the field for this text

static void SetText(TMP_Text textField, string name, string category, string language)

Sets the text of a Text Mesh Pro Text field in the provided language.

Paramenter	Function
textField	The Text Mesh Pro Text to set
name	The name of the field where the strings for this text reside
category	The category of the field for this text
language	Code of the language the text should be in

static string GetString(string name, string category = DEFAULT_CATEGORY)

Returns the string at the provided location in the set language

Paramenter	Function
name	The name of the field where the string resides
category	The category of the field for string

static string GetString(string name, string category, string language)

Returns the string at the provided location in the provided language

Paramenter	Function
name	The name of the field where the string resides
category	The category of the field for string
language	Code of the language the string should be in

static string GetLanguageDisplayName(string languageCode)

Returns the display name of a language

Paramenter	Function
languageCode	The code of the language the display name should be returned of