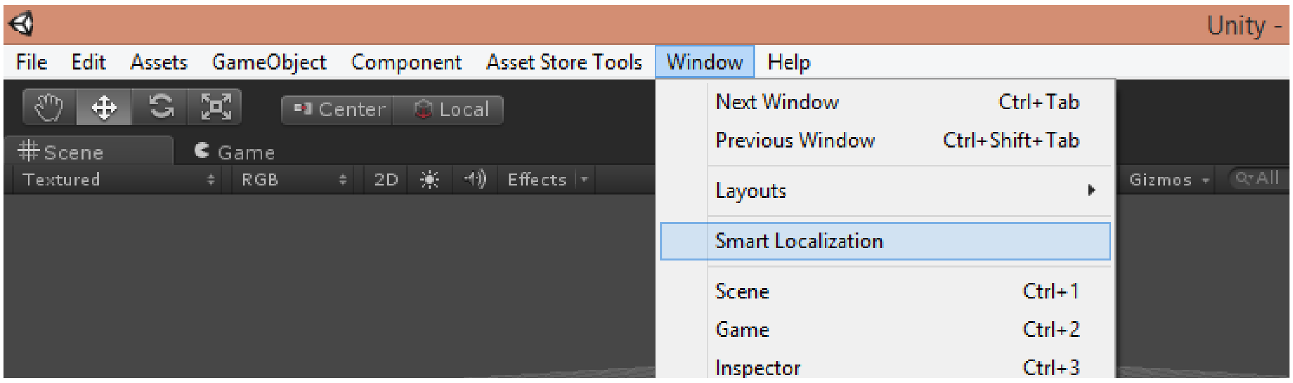
Getting started with Smart Localization 2.x

There has been quite some changes with the Smart Localization project since version 1.x. First of all, the development is no longer handled by Cry Wolf Studios. Since we decided to liquidate the studio, me(Niklas Borglund) and Jakob Hillerström decided to take over the project and release it under the team name called janeTech.

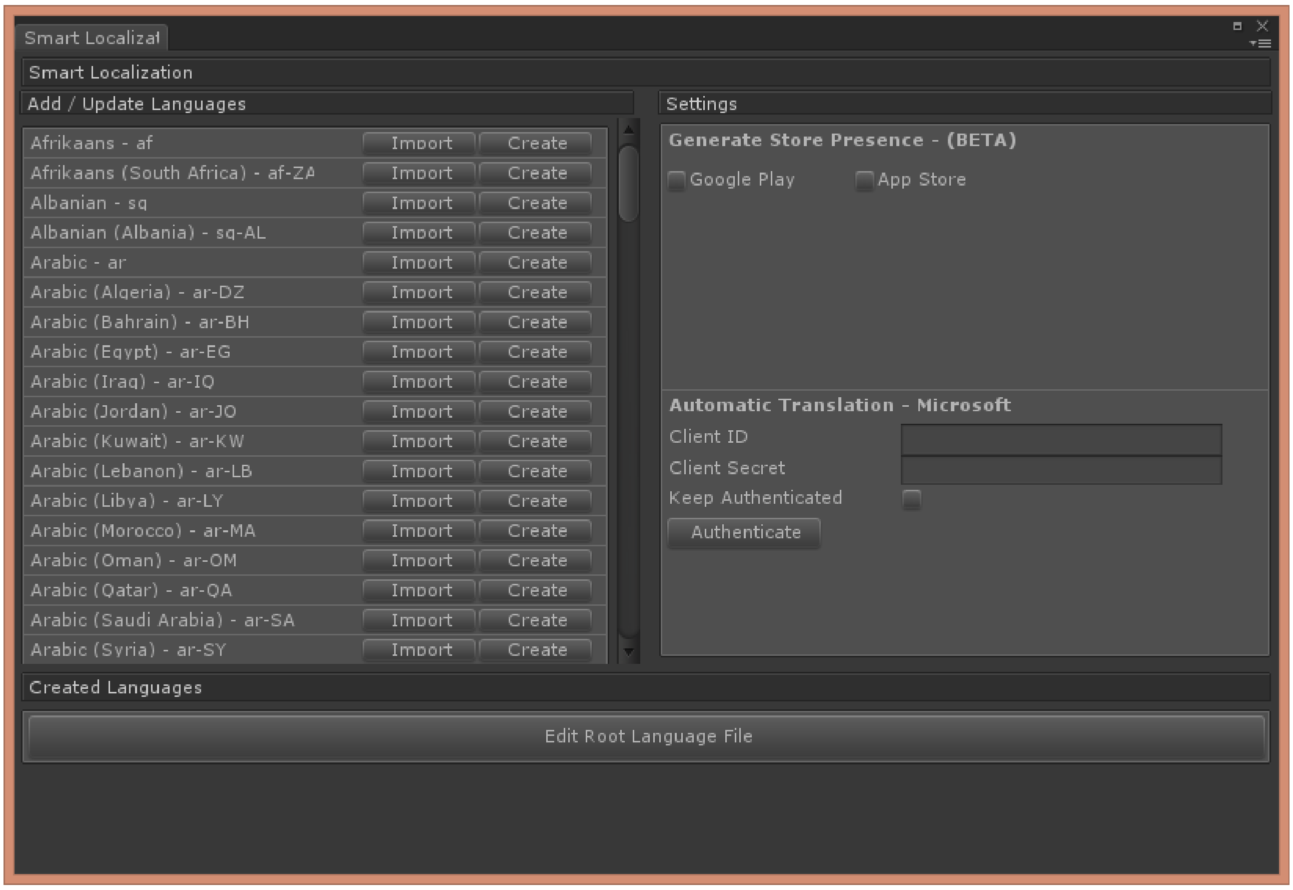
As I mentioned above, there's been some changes to the project since then. The biggest ones are:

- We are not using the System.Globalization.CultureInfo class anymore. We are using our own class called SmartCultureInfo.
- Everything is now encapsulated within the namespaces SmartLocalization and SmartLocalization.Editor
- The ability to Export/Import/Update CSV files
- Completely reworked ground for increased stability.
- Translation Window can be used at runtime (2.1 and above)
- Completely reworked GUI with features such as resizable column widths (2.1 and above)

To get started with Smart Localization you will have to start by downloading the project from the asset store and importing it to your project.
Once you've done that, navigate to Window->Smart Localization.



The window will prompt you to create a Smart Localization workspace if you haven't done so already. Follow the instructions until you see the main localization window.



Afrikaans - af	Import	Create
Afrikaans (South Africa) - af-ZA	Import	Create
Albanian - sq	Import	Create
Albanian (Albania) - sq-AL	Import	Create
Arabic - ar	Import	Create
Arabic (Algeria) - ar-DZ	Import	Create
Arabic (Bahrain) - ar-BH	Import	Create
Arabic (Egypt) - ar-EG	Import	Create
Arabic (Iraq) - ar-IQ	Import	Create
Arabic (Jordan) - ar-JO	Import	Create
Arabic (Kuwait) - ar-KW	Import	Create
Arabic (Lebanon) - ar-LB	Import	Create
Arabic (Libya) - ar-LY	Import	Create
Arabic (Morocco) - ar-MA	Import	Create
Arabic (Oman) - ar-OM	Import	Create
Arabic (Qatar) - ar-QA	Import	Create
Arabic (Saudi Arabia) - ar-SA	Import	Create
Arabic (Syria) - ar-SY	Import	Create

Generate Store Presence - (BETA)

Google Play App Store

Automatic Translation - Microsoft

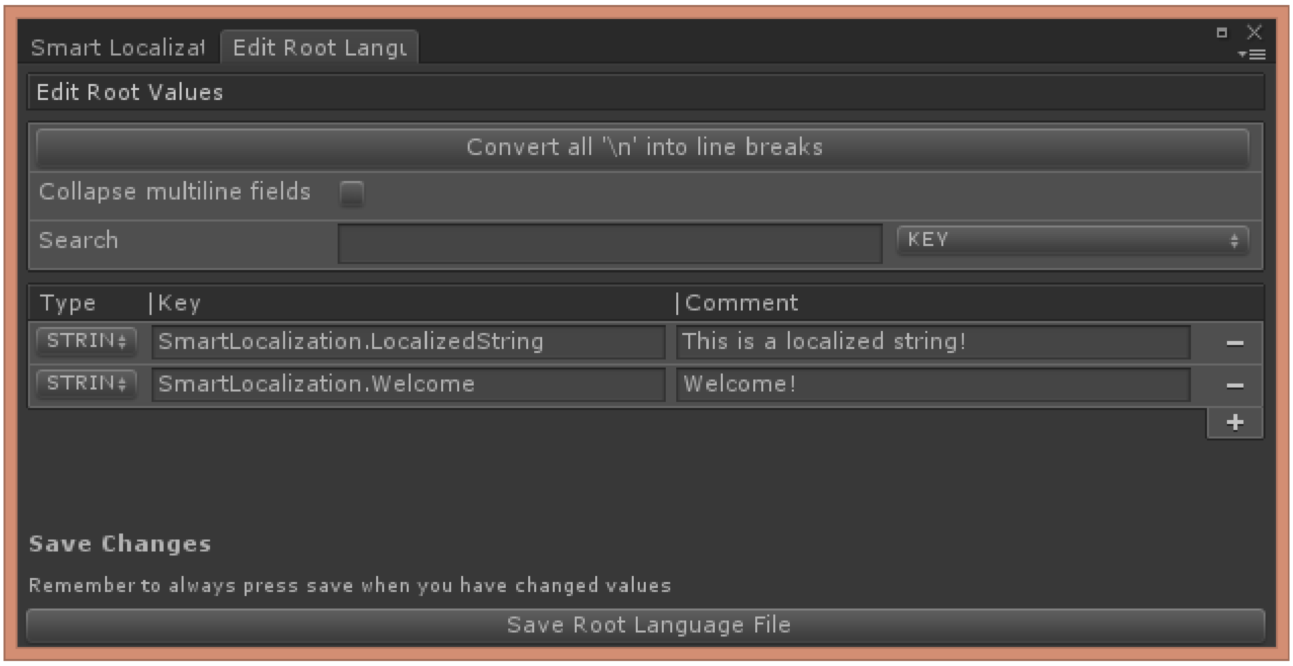
Client ID

Client Secret

Keep Authenticated

Press the Edit Root Language File button. This will bring up a new window where you create all the keys and root values along with the key types for your project. We'll start by adding two keys; SmartLocalization.Welcome and SmartLocalization.LocalizedString.

Once you've done that, press save and return to the main localization window.



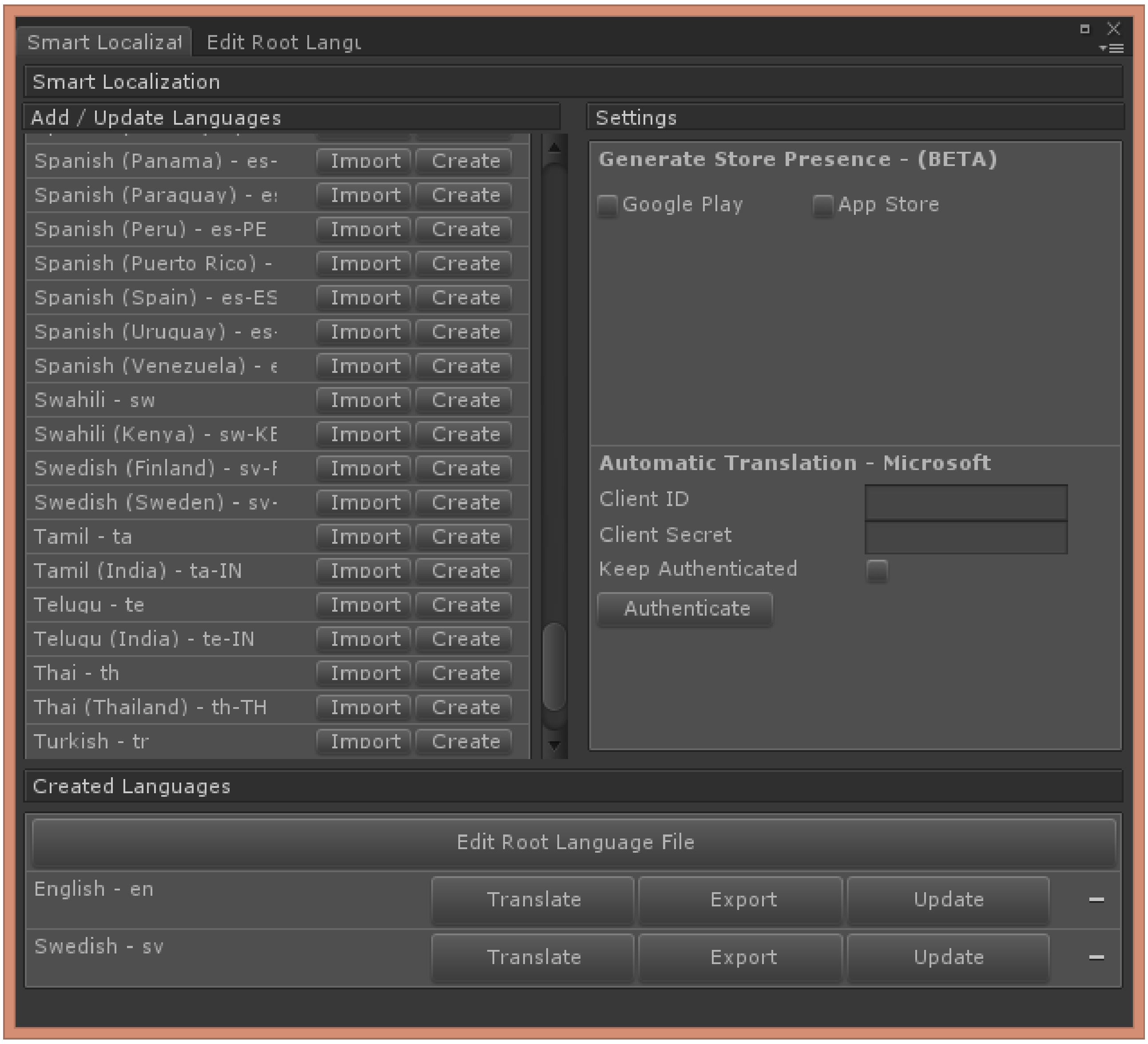
Type	Key	Comment
STRIN	SmartLocalization.LocalizedString	This is a localized string!
STRIN	SmartLocalization.Welcome	Welcome!

Save Changes

Remember to always press save when you have changed values

Now it's time to create some languages. For the purpose of this tutorial, we'll create two languages;

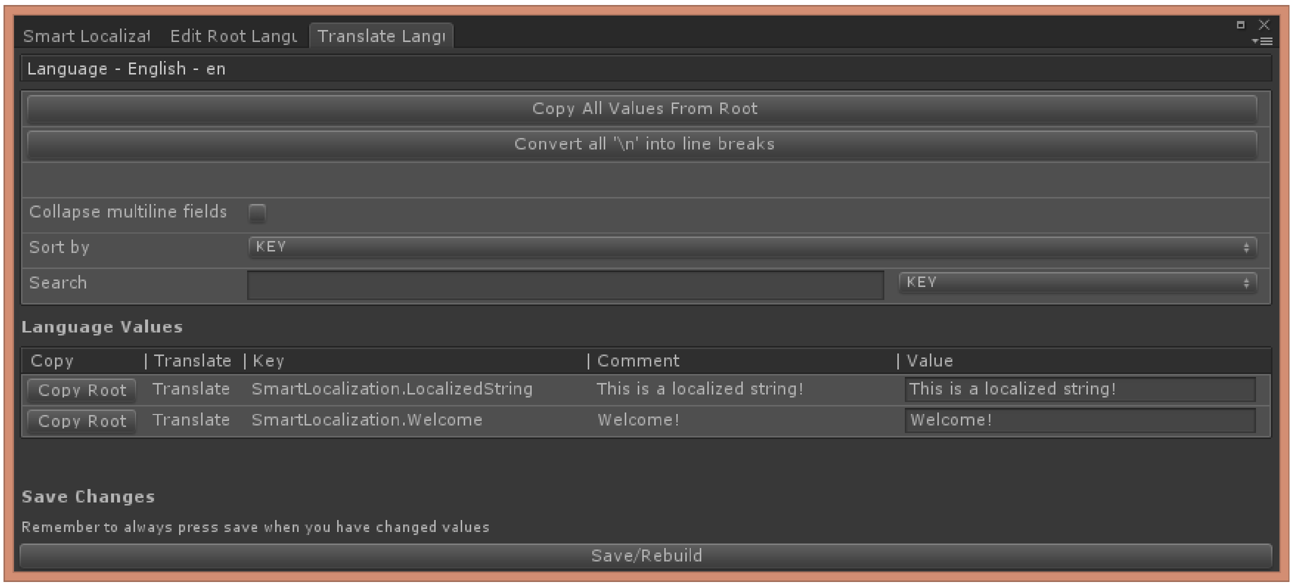
English and Swedish. Find the two cultures in the Add/Update Languages scroll list and press Create on each one.



Spanish (Panama) - es-	Import	Create
Spanish (Paraguay) - e:	Import	Create
Spanish (Peru) - es-PE	Import	Create
Spanish (Puerto Rico) -	Import	Create
Spanish (Spain) - es-ES	Import	Create
Spanish (Uruguay) - es-	Import	Create
Spanish (Venezuela) - e	Import	Create
Swahili - sw	Import	Create
Swahili (Kenya) - sw-KE	Import	Create
Swedish (Finland) - sv-F	Import	Create
Swedish (Sweden) - sv-	Import	Create
Tamil - ta	Import	Create
Tamil (India) - ta-IN	Import	Create
Telugu - te	Import	Create
Telugu (India) - te-IN	Import	Create
Thai - th	Import	Create
Thai (Thailand) - th-TH	Import	Create
Turkish - tr	Import	Create

English - en	Translate	Export	Update	–
Swedish - sv	Translate	Export	Update	–

Now press the 'Translate' button placed on the English culture row. A new window will be brought up. Since we wrote our root values in english, all we have to do here is to press the "Copy All Values From Root" – button. Don't forget to press save and return to the main window.


Smart Localizat | Edit Root Langu | Translate Langu

Language - English - en

Copy All Values From Root

Convert all '\n' into line breaks

Collapse multiline fields

Sort by KEY

Search KEY

Language Values

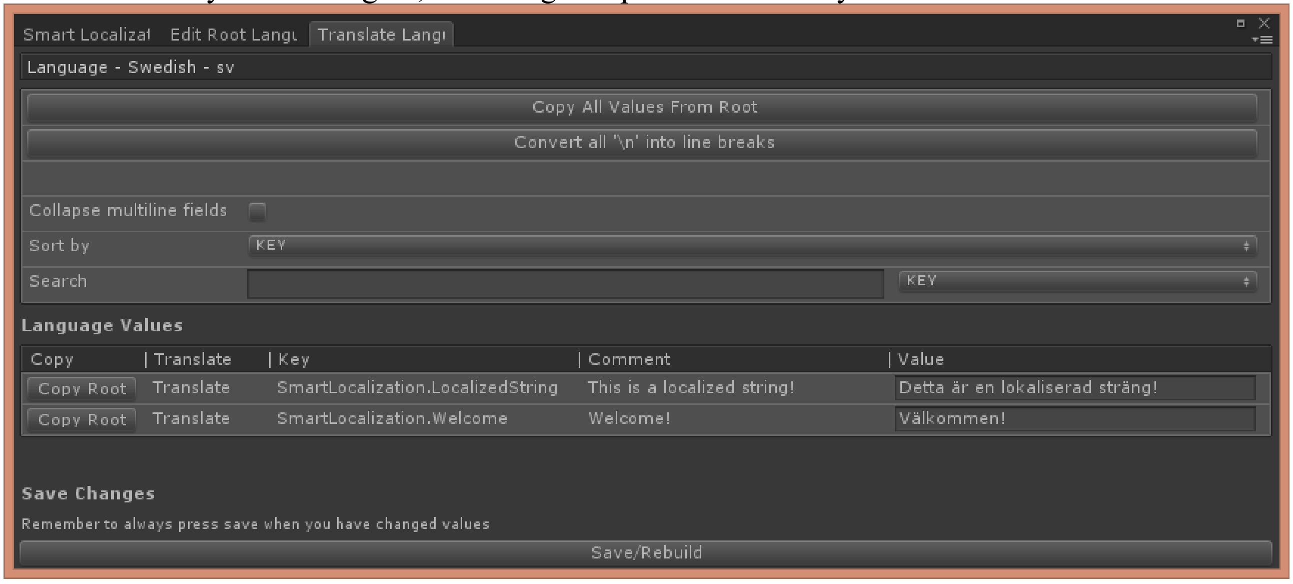
Copy	Translate	Key	Comment	Value
Copy Root	Translate	SmartLocalization.LocalizedString	This is a localized string!	This is a localized string!
Copy Root	Translate	SmartLocalization.Welcome	Welcome!	Welcome!

Save Changes

Remember to always press save when you have changed values

Save/Rebuild

Now do the same procedure with Swedish. Press Translate and write the Swedish values in the 'Value' column. If you don't know Swedish, just copy what it says in the image below or write whatever value you want. Again, don't forget to press save when you are done.


Smart Localizat | Edit Root Langu | Translate Langu

Language - Swedish - sv

Copy All Values From Root

Convert all '\n' into line breaks

Collapse multiline fields

Sort by KEY

Search KEY

Language Values

Copy	Translate	Key	Comment	Value
Copy Root	Translate	SmartLocalization.LocalizedString	This is a localized string!	Detta är en lokaliserad sträng!
Copy Root	Translate	SmartLocalization.Welcome	Welcome!	Välkommen!

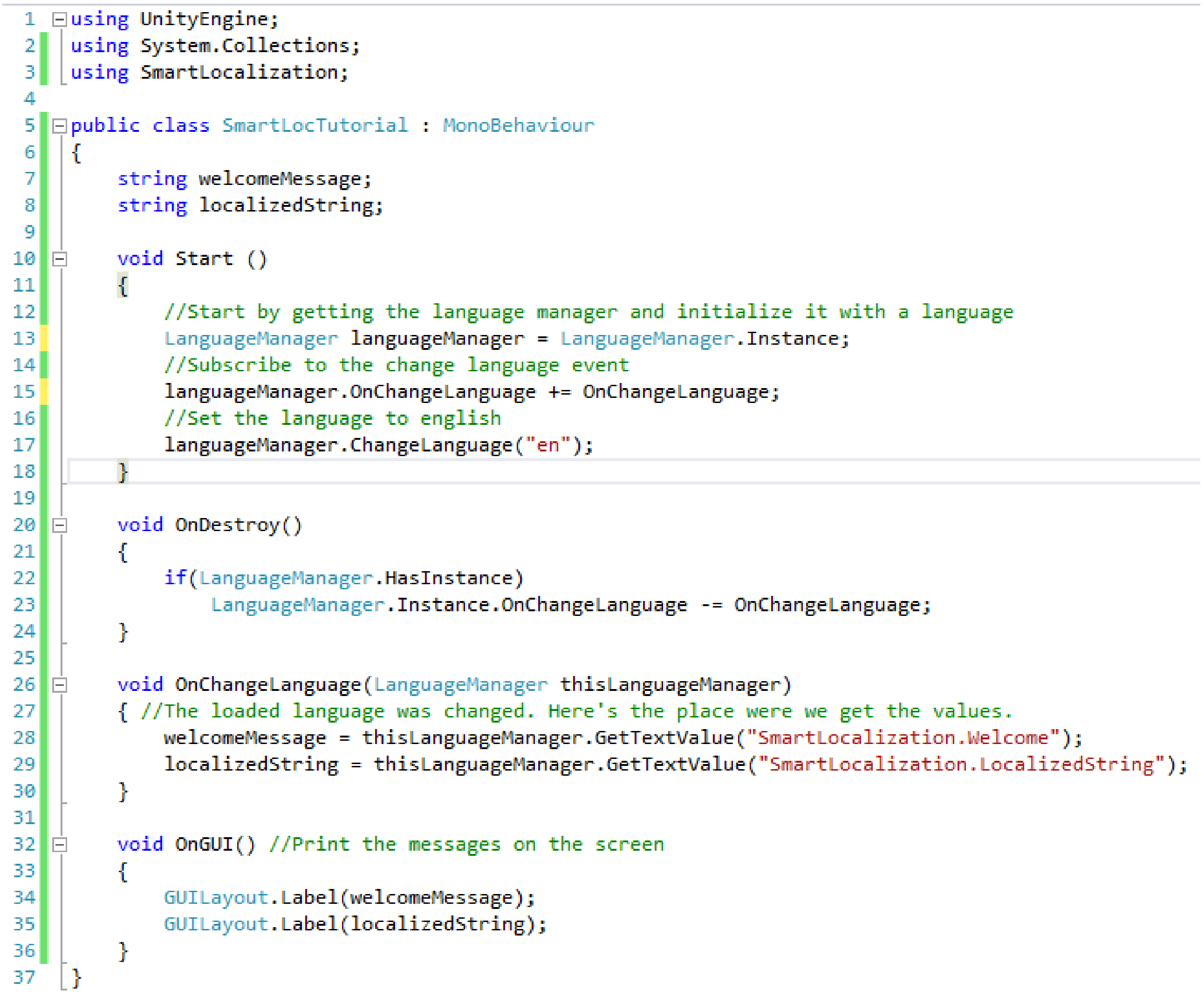
Save Changes

Remember to always press save when you have changed values

Save/Rebuild

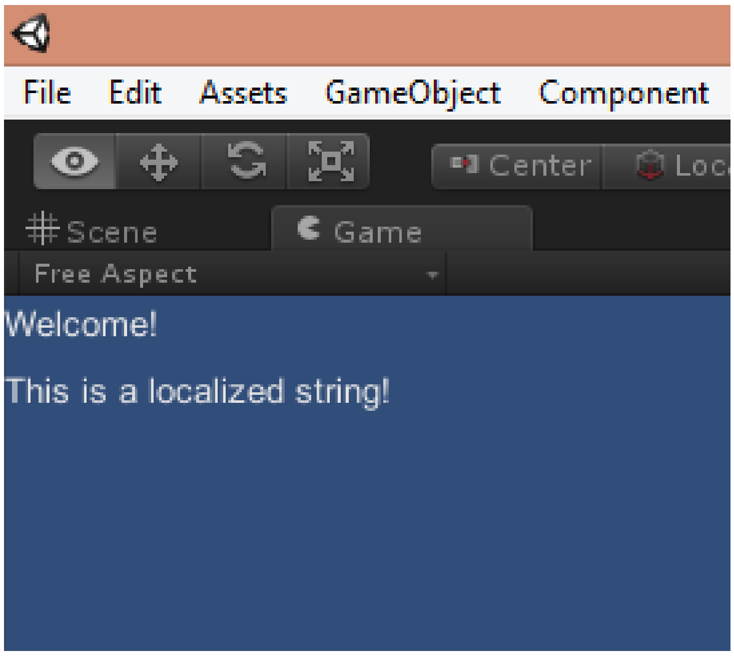
That's basically all the setup you need to get started. By now, you can open the scene LoadAllLanguages.unity that can be found in the Smart Localization package from the store and you should see the created values in the scene when you press play.

Next I'll show how a basic and easy way to get the localized values from code. I'll start by creating a new script called SmartLocTutorial.cs.
In the script we'll load the english language and print the two localized strings we created earlier.



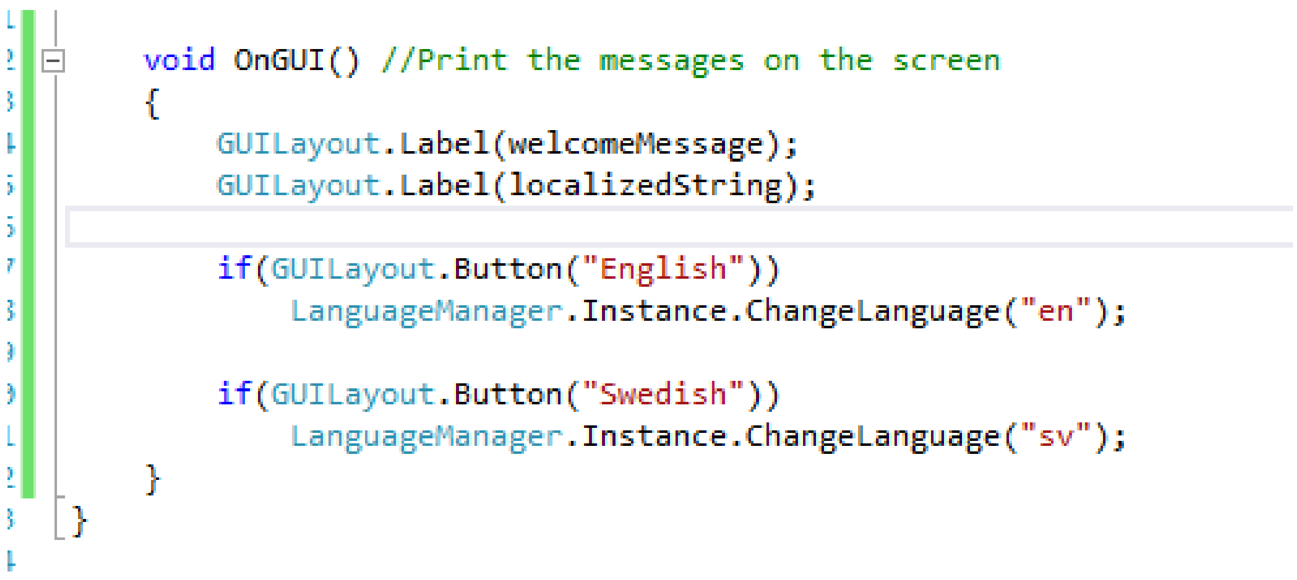
```
using UnityEngine;
using System.Collections;
using SmartLocalization;

public class SmartLocTutorial : MonoBehaviour
{
    string welcomeMessage;
    string localizedString;

    void Start ()
    {
        //Start by getting the language manager and initialize it with a language
        LanguageManager languageManager = LanguageManager.Instance;
        //Subscribe to the change language event
        languageManager.OnChangeLanguage += OnChangeLanguage;
        //Set the language to english
        languageManager.ChangeLanguage("en");
    }

    void OnDestroy()
    {
        if(LanguageManager.HasInstance)
            LanguageManager.Instance.OnChangeLanguage -= OnChangeLanguage;
    }

    void OnChangeLanguage(LanguageManager thisLanguageManager)
    { //The loaded language was changed. Here's the place were we get the values.
        welcomeMessage = thisLanguageManager.GetTextValue("SmartLocalization.Welcome");
        localizedString = thisLanguageManager.GetTextValue("SmartLocalization.LocalizedString");
    }

    void OnGUI() //Print the messages on the screen
    {
        GUILayout.Label(welcomeMessage);
        GUILayout.Label(localizedString);
    }
}
```

Adding this script to an empty game object in your scene should produce the following output.



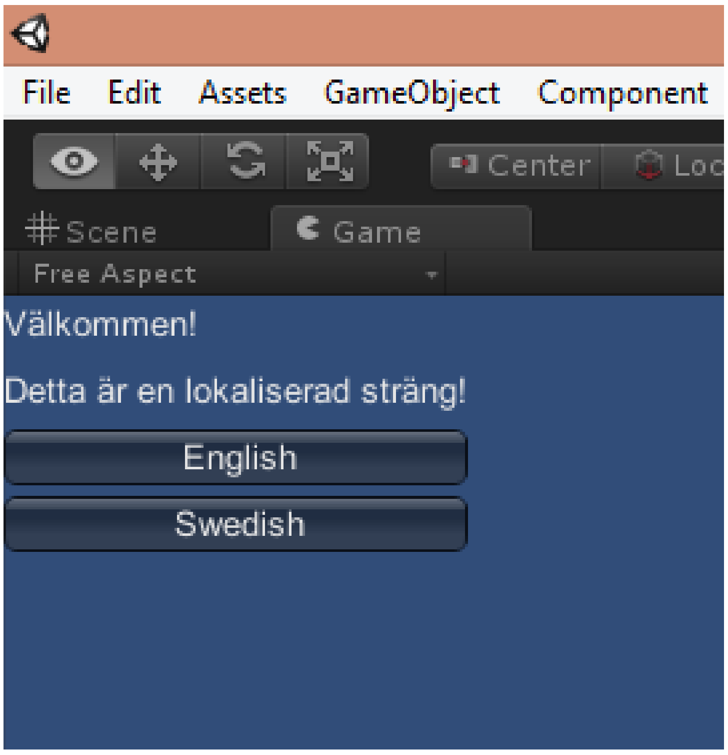
```
Welcome!
This is a localized string!
```

What if you want to switch to the swedish values at runtime? Simplest way to do that would be to expand the OnGUI method in our SmartLocTutorial script.



```
    void OnGUI() //Print the messages on the screen
    {
        GUILayout.Label(welcomeMessage);
        GUILayout.Label(localizedString);

        if(GUILayout.Button("English"))
            LanguageManager.Instance.ChangeLanguage("en");

        if(GUILayout.Button("Swedish"))
            LanguageManager.Instance.ChangeLanguage("sv");
    }
}
```

Play your scene again, press the Swedish button, and you should see the Swedish values on the screen.



Välkommen!

Detta är en lokaliserad sträng!