

Home

H. Ibrahim Penekli edited this page 38 minutes ago · 11 revisions

Welcome to the GameToolkit-Localization wiki!

With [Asset Localization](#), you can easily localize your [TextAsset](#), [AudioClip](#), [Sprite](#), [Texture](#), [Font](#) and [Prefab](#) assets. Also you can create custom localizable asset functionality for your custom assets.

- 1. [Getting Started](#)
- 2. [Localization Explorer](#)
- 3. [Localizing Components](#)
- 4. [Quick Translate Configuration](#)
- 5. [Scripting Reference](#)

▼ Pages 6

[Home](#)

[Getting Started](#)


[Localization Explorer](#)


[Localizing Components](#)

[Quick Translate Configuration](#)

[Scripting Reference](#)

Clone this wiki locally

<https://github.com/ibrahimpenekli/GameToolkit-Localization/wiki> 

 Clone in Desktop

Getting Started

H. Ibrahim Penekli edited this page 38 minutes ago · 9 revisions

1. Adding the Localization package in your project

Open or create your project in Unity.

Then import the Asset Localization package from the Asset Store window in Unity editor. You can find detailed explanation [here](#).

2. Creating localization settings

Localization settings is created automatically when package is imported. Also you can manually create via

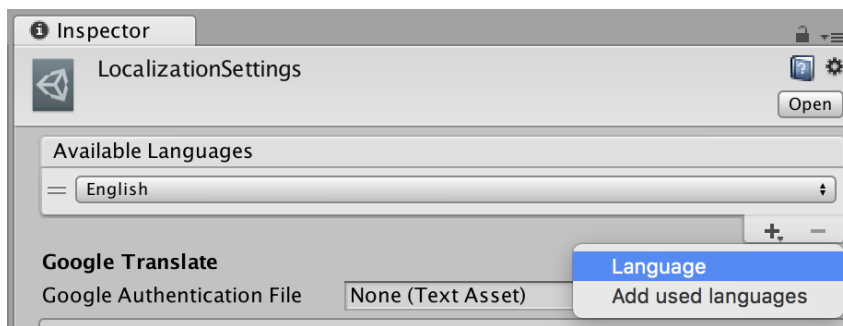
Project Window -> GameToolkit -> Localization -> Localization Settings .

Settings file must be kept under any **Resources** folder. You can read detailed explanation about [Resources](#).

3. Localization settings

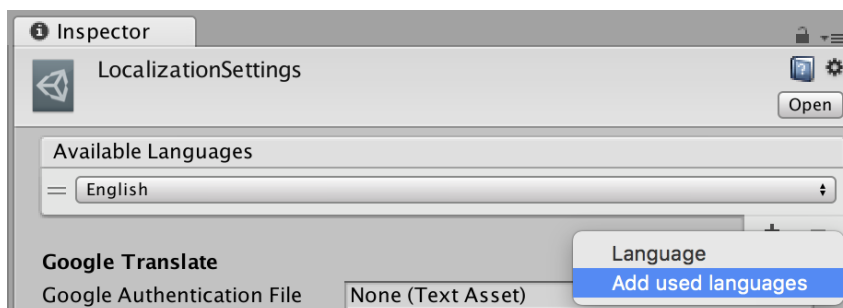
- You can add languages that must be available in you game as many as you want via

Add (+) -> Language



- Also you can add used languages in your assets via

Add (+) -> Add used languages



- If you want to use **Quick Translate** option, you should set **Google Authentication File** claimed from **Google Cloud**. For more information look at [Quick Translate Configuration](#) page.

▼ Pages 6

[Home](#)

[Getting Started](#)

[Localization Explorer](#)

[Localizing Components](#)

[Quick Translate Configuration](#)

[Scripting Reference](#)

Clone this wiki locally

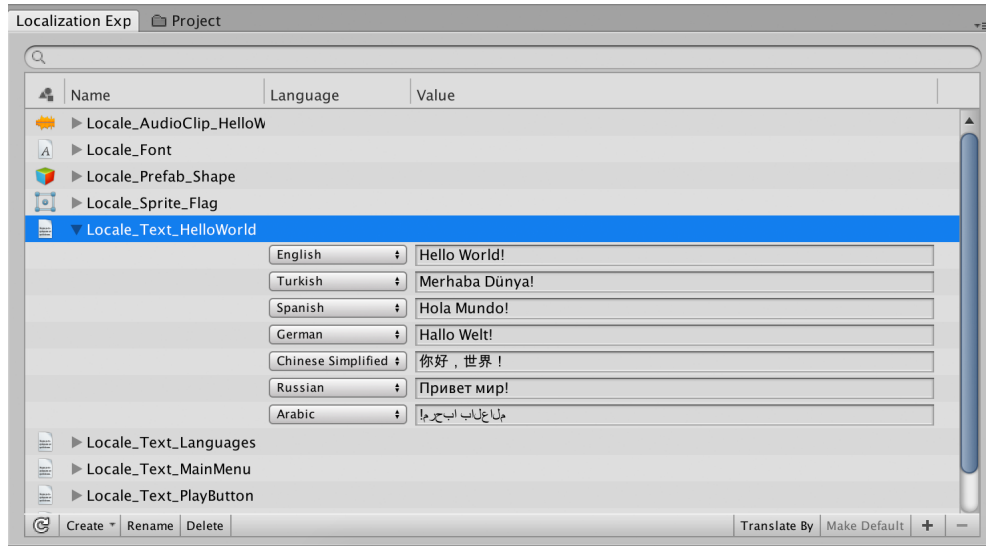
<https://github.com/ibrahimpenekli/GameToolkit-Localization/wiki/Getting-Started>



[Clone in Desktop](#)

Localization Explorer

H. Ibrahim Penekli edited this page 38 minutes ago · 10 revisions



▼ Pages 6

[Home](#)

[Getting Started](#)

[Localization Explorer](#)

[Localizing Components](#)

[Quick Translate Configuration](#)

[Scripting Reference](#)

Clone this wiki locally

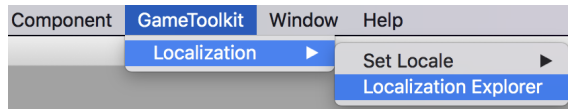
<https://github.com/ibrahimpenekli/GameToolkit-Localization/wiki/Localization-Explorer>

Clone in Desktop

1. Opening Localization Explorer

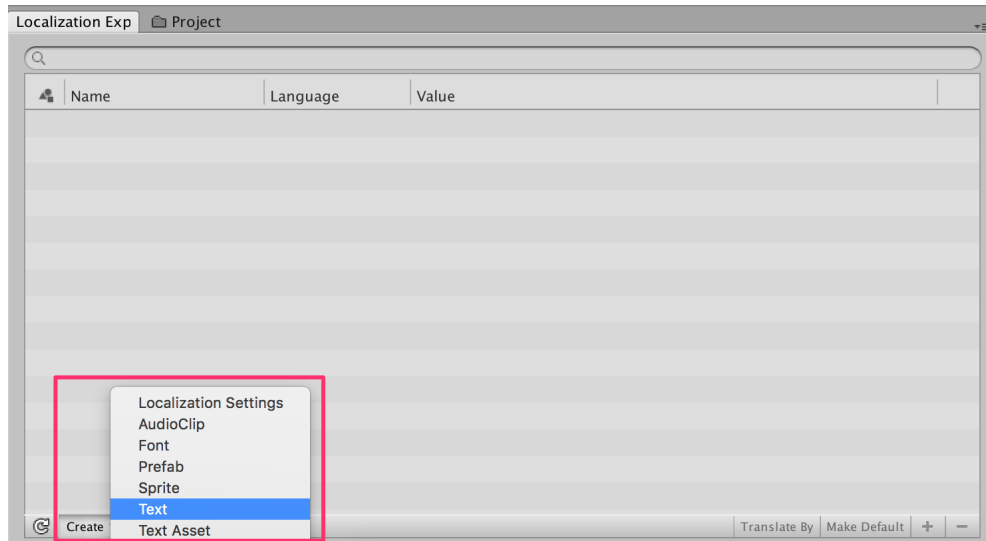
Localization explorer lets you manage your localized assets in a single window. You can access the window via

GameToolkit -> Localization -> Localization Explorer

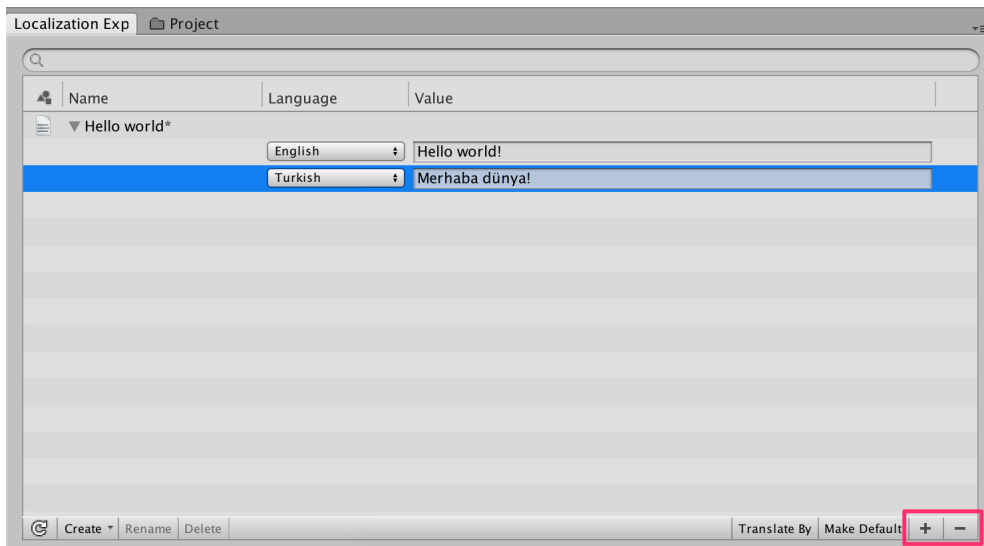


2. Add/Modify/Delete localized asset

- Add a localized asset by clicking Localization Explorer -> Create .
 - Select an asset type from dropdown menu opened.
 - Give a name for the created asset. The asset created is shown in the localization explorer.



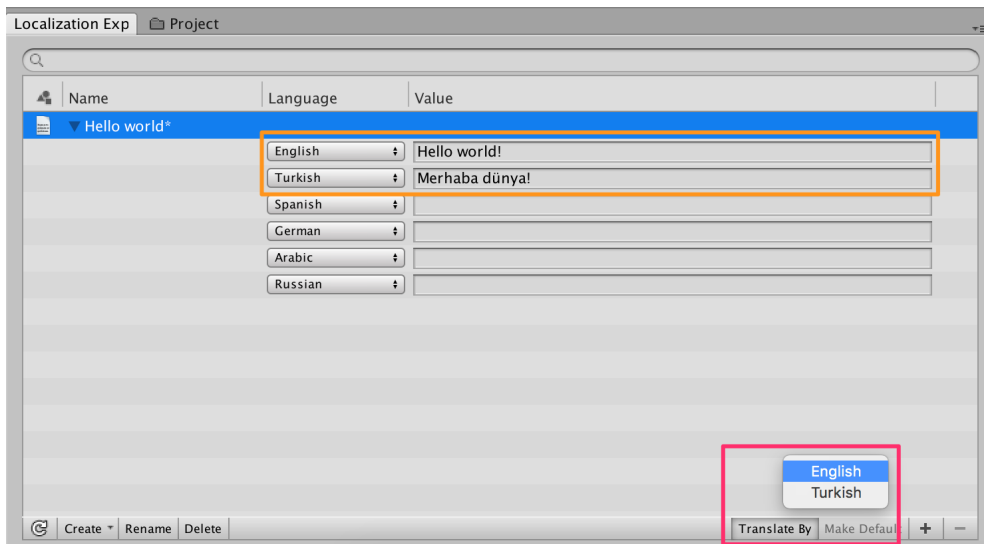
- Selected localized asset can be renamed by clicking **Rename** button and deleted by clicking **Delete** button.
- Expand the created asset and add **locale (language & value pair)** as many as you want by clicking **+** button. Also you can remove the selected locale by clicking **-** button through the explorer.
 - Enter a value with selecting corresponding language



3. Quick translate missing locale(s)

You can translate missing locale(s) automatically by clicking **Translate By** button:

- Choose a language that has already translated



Quick Translate option is available only for **Text** assets and it must be configured. See [Quick Translate Configuration](#) to learn how to configure it correctly.

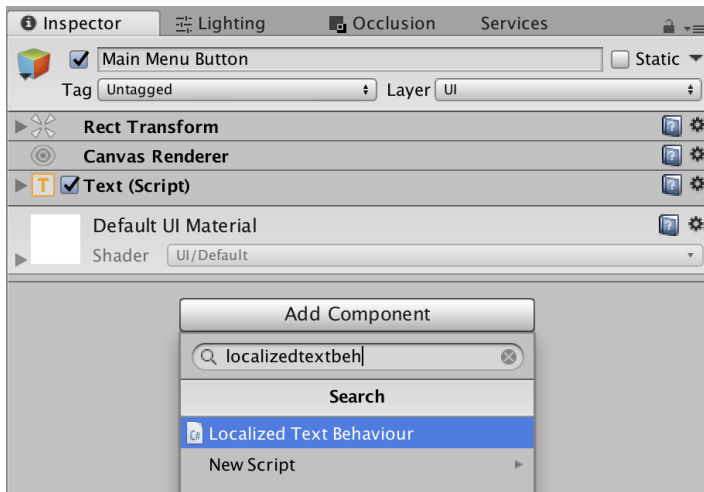
Localizing Components

H. Ibrahim Penekli edited this page 37 minutes ago · 4 revisions

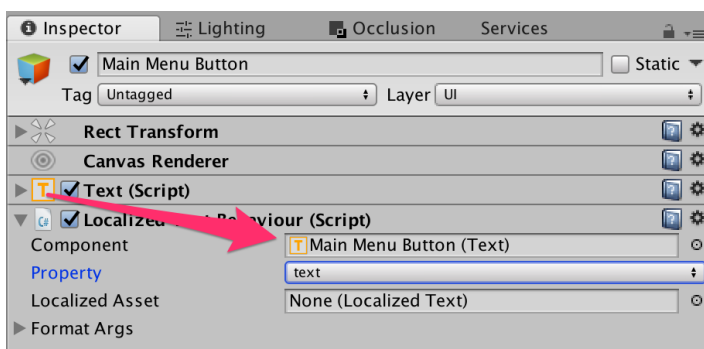
Localized assets must be attached to the actual game objects in the scene in order to set localized value to the appropriate component's attribute. The package offers various ready-to-use components.

Localizing a Text

- When the Text game object selected, click **Add Component** button in the inspector.
- Search for `LocalizedTextBehaviour` and add it.



- `LocalizedTextBehaviour` needs a component and it's property to set localized value to the property.
- Drag the `Text` component from the game object to the `LocalizedTextBehaviour`'s `Component` field.



- The localization system search for the text (string) based properties. Property dropdown lists available properties. Simply select `text` property.

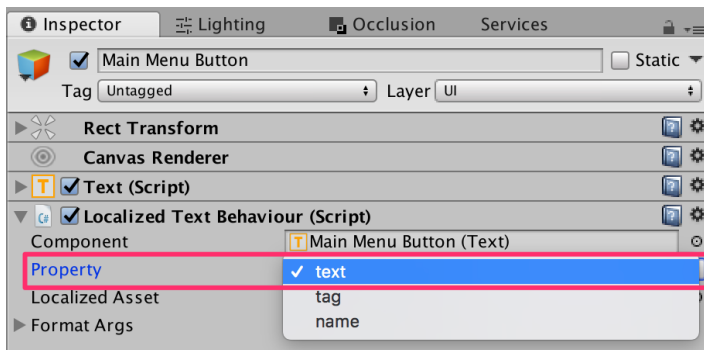
▼ Pages 6

[Home](#)[Getting Started](#)[Localization Explorer](#)[Localizing Components](#)[Quick Translate Configuration](#)[Scripting Reference](#)

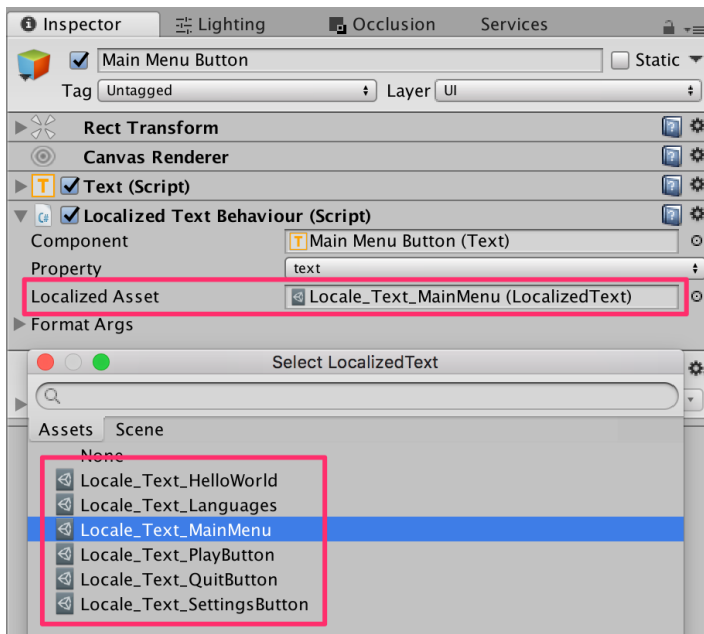
Clone this wiki locally

<https://github.com/ibrahimpenekli/GameToolkit-Localization/wiki>

Clone in Desktop



- Finally, select localized asset from the Localized Asset field.



When the game starts, the localization sets the `text` property of the selected component with selected localized asset's value. Updates the value whenever game language has changed.

This generic component & property selection provides us to use at any component that has text based property. For example; we can use same behaviour with same steps for the [Text Mesh Pro](#) component. You should follow the similar steps for all localized asset types. For instance; attach `LocalizedSpriteBehaviour` for the [Image](#) component and select [sprite](#) property and done!

If you want to go further and create your custom localization behaviour, see [here](#).

Quick Translate Configuration

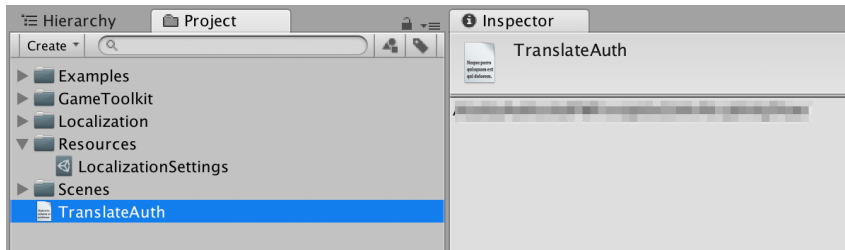
H. Ibrahim Penekli edited this page 16 hours ago · 4 revisions

1. Google Cloud Translation API

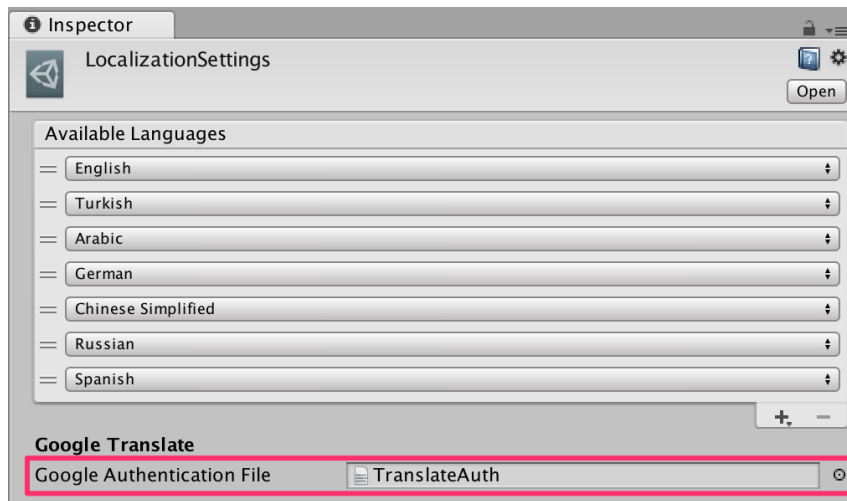
Google Cloud Translation API provides a simple programmatic interface for translating an arbitrary string into any supported language. Quick translate uses the API for translating missing locales. Google Translate API requires an authentication for pricing. More detailed information [here](#).

2. Configure localization settings for Quick Translate

- First obtain your API key from [Google Cloud Console](#). See [how-to guide](#).
- Create a text file in project assets and paste your API key into here. (Make sure that the copied key is correct one)



- Open `LocalizationSettings` from your `Resources` folder.
- Attach the created key file to `Google Authentication File` field.



Finally, you can translate missing locales automatically. See [here](#).

P.S. If you want to use quick translate only in the Editor, exclude the auth file when deploying your game for production. For more information how to exclude an asset [see here](#).

▼ Pages 6

[Home](#)

[Getting Started](#)

[Localization Explorer](#)

[Localizing Components](#)

[Quick Translate Configuration](#)

[Scripting Reference](#)

Clone this wiki locally

<https://github.com/ibrahimpenekli/GameToolkit-Localization/wiki/Quick-Translate-Configuration>

Clone in Desktop

Scripting Reference

H. Ibrahim Penekli edited this page 19 minutes ago · 4 revisions

Localization Settings

```
// To get localization settings:
LocalizationSettings localizationSettings = LocalizationSettings.Instance;

// You can get available languages as:
List<SystemLanguage> availableLanguages = localizationSettings.AvailableLanguages;

// To access Google auth file:
TextAsset authFile = localizationSettings.GoogleAuthenticationFile;
```

Localization Manager

```
// You can get current language:
SystemLanguage currentLanguage = Localization.Instance.CurrentLanguage;

// or you can set current language:
Localization.Instance.CurrentLanguage = SystemLanguage.English;

// or set by system language:
Localization.Instance.SetSystemLanguage();

// or set by default language defined in LocalizationSettings (first item is the default)
Localization.Instance.SetDefaultLanguage();

// Register application locale changed event:
Localization.Instance.LocaleChanged += (object sender, LocaleChangedEventArgs e) =>
{
    Debug.Log("Application locale has changed from " + e.PreviousLanguage + " to " + e.CurrentLanguage);
};
```

Extending Custom Localized Asset

Creating custom localized asset is involved simple steps:

- Extend class from `LocalizedAsset<T>`, enter your asset type for generic parameter:

```
[CreateAssetMenu(fileName = "MyLocalizedCustomAsset", menuName = "GameToolkit/Localization/Custom Asset")]
public class MyLocalizedCustomAsset : LocalizedAsset<MyCustomAsset>
```

- Your custom localized asset automatically registered under `Localization Explorer` -> Create menu if you set `menuName` property of `CreateAssetMenu` attribute as `"GameToolkit/Localization/<your_asset_name>"`
- Create serializable asset item by extending `LocaleItem<T>`, enter your asset type for generic parameter again (it is necessary for the Unity to serialize object):

```
[Serializable]
private class MyCustomLocaleItem : LocaleItem<MyCustomAsset> { };
```

- Define locale items array with concrete type you declared:

```
[SerializeField]
private MyCustomLocaleItem[] m_LocaleItems = new MyCustomLocaleItem[1];
```

▼ Pages 6

[Home](#)

[Getting Started](#)

[Localization Explorer](#)

[Localizing Components](#)


[Quick Translate Configuration](#)

[Scripting Reference](#)

Clone this wiki locally

<https://github.com/ibrahimpenekli/GameToolkit-Localization/wiki/Scripting-Reference>



 Clone in Desktop

- Finally, implement getter method for getting locale items with the base class type:

```
public override LocaleItemBase[] LocaleItems { get { return m_LocaleItems; } }
```

Complete code:

```
using System;
using UnityEngine;
using GameToolkit.Localization;

[CreateAssetMenu(fileName = "MyLocalizedCustomAsset", menuName = "GameToolkit/Localiz
public class MyLocalizedCustomAsset : LocalizedAsset<MyCustomAsset>
{
    [Serializable]
    private class MyCustomLocaleItem : LocaleItem<MyCustomAsset> { };

    [SerializeField]
    private MyCustomLocaleItem[] m_LocaleItems = new MyCustomLocaleItem[1];

    public override LocaleItemBase[] LocaleItems { get { return m_LocaleItems; } }
}
```

Congratulation! You have a custom localized asset that can use your game.

Extending Custom Localized Asset Behaviour

If you want to extend localized asset behaviour, you have two options:

- If you want to implement completely custom behaviour, you should extend from `LocalizedAssetBehaviour`.
- If you want to create component&property based behaviour for your custom localized asset, then you should extend from `LocalizedGenericAssetBehaviour`.

1. Extending from `LocalizedAssetBehaviour`

You must extend your class from `LocalizedAssetBehaviour` and override the `UpdateComponentValue()` appropriately. This method is invoked every-time when game starts or application language has changed. You could update the component property with your custom localized asset's value.

```
public class MyLocalizedAssetBehaviour : LocalizedAssetBehaviour
{
    public MyLocalizedCustomAsset LocalizedAsset;

    protected override void UpdateComponentValue()
    {
        // Update the specified property with current value.
        ... = LocalizedAsset.Value;
    }
}
```

2. Extending from `LocalizedGenericAssetBehaviour<TAsset, TType>`

The only step you need to take is the extend from `LocalizedGenericAssetBehaviour` and specify your custom asset type for the first generic parameter, and specify your asset value type as the second generic parameter. That's it!

```
public class MyLocalizedAssetBehaviour : LocalizedGenericAssetBehaviour<MyLocalizedC
{
}
```