

I18N.DotNet

main@ae28a8

Generated by Doxygen 1.9.5



<b>1 I18N.DotNet</b>	<b>1</b>
1.1 About	1
1.2 Installation	1
1.3 Getting Started	1
1.3.1 Writing/Adapting Source Code (I18N)	1
1.3.2 Writing Translations (L10N)	2
1.3.3 Embedding the Translations File	2
1.4 Advanced Usage (Internationalizing Applications)	3
1.4.1 Global Localizer	3
1.4.2 Local Localizers	3
1.4.3 Language Identifiers & Variants	3
1.4.4 String Format	4
1.4.5 Contexts	4
1.4.6 Loading Translations	5
1.4.7 Specifying the Translation Target Language	5
1.5 Advanced Usage (Internationalizing Libraries)	7
1.5.1 Library Localizers	7
1.6 API Documentation	7
1.6.0.1 ILocalizer Interface	7
1.6.0.2 ILoadableLocalizer Interface	8
1.6.0.3 Localizer Class	8
1.6.0.4 AutoLoadLocalizer Class	8
1.6.1 Full API Documentation	8
<b>2 Namespace Index</b>	<b>9</b>
2.1 Package List	9
<b>3 Hierarchical Index</b>	<b>11</b>
3.1 Class Hierarchy	11
<b>4 Class Index</b>	<b>13</b>
4.1 Class List	13
<b>5 Namespace Documentation</b>	<b>15</b>
5.1 I18N Namespace Reference	15
5.2 I18N.DotNet Namespace Reference	15
<b>6 Class Documentation</b>	<b>17</b>
6.1 AutoLoadLocalizer Class Reference	17
6.1.1 Detailed Description	19
6.1.2 Constructor & Destructor Documentation	19
6.1.2.1 AutoLoadLocalizer()	19
6.1.3 Member Function Documentation	19
6.1.3.1 Context() [1/2]	20

6.1.3.2 Context() [2/2]	20
6.1.3.3 Load()	20
6.1.3.4 LoadXML() [1/4]	21
6.1.3.5 LoadXML() [2/4]	21
6.1.3.6 LoadXML() [3/4]	22
6.1.3.7 LoadXML() [4/4]	22
6.1.3.8 Localize() [1/3]	23
6.1.3.9 Localize() [2/3]	23
6.1.3.10 Localize() [3/3]	23
6.1.3.11 LocalizeFormat()	24
6.1.4 Member Data Documentation	24
6.1.4.1 DEFAULT_RESOURCE_NAME	24
6.2 GlobalLocalizer Class Reference	24
6.2.1 Detailed Description	25
6.2.2 Member Function Documentation	25
6.2.2.1 Context()	25
6.2.2.2 Localize() [1/3]	25
6.2.2.3 Localize() [2/3]	26
6.2.2.4 Localize() [3/3]	26
6.2.2.5 LocalizeFormat()	27
6.2.3 Property Documentation	27
6.2.3.1 Localizer	27
6.3 ILoadableLocalizer Interface Reference	27
6.3.1 Detailed Description	28
6.3.2 Member Function Documentation	28
6.3.2.1 Context() [1/2]	28
6.3.2.2 Context() [2/2]	29
6.3.2.3 LoadXML() [1/4]	29
6.3.2.4 LoadXML() [2/4]	30
6.3.2.5 LoadXML() [3/4]	30
6.3.2.6 LoadXML() [4/4]	32
6.3.2.7 Localize() [1/3]	32
6.3.2.8 Localize() [2/3]	34
6.3.2.9 Localize() [3/3]	34
6.3.2.10 LocalizeFormat()	35
6.4 ILocalizer Interface Reference	35
6.4.1 Detailed Description	36
6.4.2 Member Function Documentation	36
6.4.2.1 Context() [1/2]	36
6.4.2.2 Context() [2/2]	37
6.4.2.3 Localize() [1/3]	37
6.4.2.4 Localize() [2/3]	38

6.4.2.5 Localize() [3/3] . . . . .	38
6.4.2.6 LocalizeFormat() . . . . .	38
6.5 Localizer Class Reference . . . . .	39
6.5.1 Detailed Description . . . . .	41
6.5.2 Constructor & Destructor Documentation . . . . .	41
6.5.2.1 Localizer() . . . . .	41
6.5.3 Member Function Documentation . . . . .	41
6.5.3.1 Context() [1/2] . . . . .	41
6.5.3.2 Context() [2/2] . . . . .	42
6.5.3.3 LoadXML() [1/4] . . . . .	42
6.5.3.4 LoadXML() [2/4] . . . . .	42
6.5.3.5 LoadXML() [3/4] . . . . .	43
6.5.3.6 LoadXML() [4/4] . . . . .	43
6.5.3.7 Localize() [1/3] . . . . .	44
6.5.3.8 Localize() [2/3] . . . . .	44
6.5.3.9 Localize() [3/3] . . . . .	44
6.5.3.10 LocalizeFormat() [1/2] . . . . .	45
6.5.3.11 LocalizeFormat() [2/2] . . . . .	45
6.6 ILoadableLocalizer.ParseException Class Reference . . . . .	45
6.6.1 Detailed Description . . . . .	46
6.6.2 Constructor & Destructor Documentation . . . . .	46
6.6.2.1 ParseException() . . . . .	46
6.7 PlainString Class Reference . . . . .	46
6.7.1 Detailed Description . . . . .	46
6.7.2 Constructor & Destructor Documentation . . . . .	47
6.7.2.1 PlainString() . . . . .	47
6.7.3 Member Function Documentation . . . . .	47
6.7.3.1 operator PlainString() [1/2] . . . . .	47
6.7.3.2 operator PlainString() [2/2] . . . . .	47
6.7.4 Property Documentation . . . . .	48
6.7.4.1 Value . . . . .	48
<b>Index</b>	<b>49</b>



# Chapter 1

## I18N.DotNet

Documentation in PDF format is available [here](#).

### 1.1 About

**I18N.DotNet** is a .NET library written in C# to enable simple internationalization (I18N) / localization (L10N) (i.e. translation to different languages) of .NET applications and libraries.

The companion utility **I18N.DotNet Tool** is provided to ease management of translation files.

### 1.2 Installation

The easiest way to install **I18N.DotNet** is using the NuGet package: <https://www.nuget.org/packages/I18N.DotNet/>

### 1.3 Getting Started

To use the **I18N.DotNet** library, three steps must be followed:

1. Write/modify the source code to internationalize strings that must be translated (see Writing/Adapting Source Code (I18N)).
2. Write translations for internationalized strings (see Writing Translations (L10N)).
3. Embed the translations file in the executable (see Embedding the Translations File).

#### 1.3.1 Writing/Adapting Source Code (I18N)

When writing internationalized source code, the strings to be translated must be wrapped with a call to `I18N.DotNet.GlobalLocalizer.Localize()`.

The easier and most convenient approach for writing internationalized software is to choose a language that will be used as the base language throughout the software development (e.g., English), and then write the software just as any non-internationalized source code, except that strings to be translated must be wrapped with calls to `Localize()`. This way the base language will act as the default language when translations are not available for the current target language.

Adapting existing non-internationalized source code is as easy as wrapping the existing strings to be translated with calls to `Localize()`.

**Example (C#)**

```
using static I18N.DotNet.GlobalLocalizer;
using System;
using System.IO;
public class Program
{
    static void Main( string[] args )
    {
        int i = 0x555;
        Console.WriteLine( Localize( "Plain string to be translated" ) );
        Console.WriteLine( Localize( $"Interpolated string to be translated with value {i:X4}" ) );
    }
}
```

### 1.3.2 Writing Translations (L10N)

String translations must be stored in an XML file (the translations file) with root element `I18N`.

For each string than has been internationalized an `Entry` element under the root must be defined, with:

- A single `Key` child element which value is the internationalized string defined in the code (replacing for interpolated strings the interpolated expressions with their positional index).
- `Value` child elements with their attribute `lang` set to the target language of the translation and which value is the translated string.

**Example**

```
<?xml version="1.0" encoding="utf-8"?>
<I18N>
  <Entry>
    <Key>Plain string to be translated</Key>
    <Value lang="es">String simple a traducir</Value>
    <Value lang="fr">String simple à traduire</Value>
  </Entry>
  <Entry>
    <Key>Interpolated string to be translated with value {0:X4}</Key>
    <Value lang="es">String interpolado a traducir con valor {0:X4}</Value>
    <Value lang="fr">String interpolé à traduire avec valeur {0:X4}</Value>
  </Entry>
</I18N>
```

**NOTE:** The companion utility `I18N.DotNet Tool` can be used to ease the creation of the translations file by scanning source files and automatically generating entries for discovered internationalized strings.

### 1.3.3 Embedding the Translations File

A very convenient way of distributing the translations for an application is to embed the translations file in the executable assembly as an embedded resource identified by `Resources.I18N.xml`.

Using Visual Studio, the easiest way to achieve this is to name the translations file `_"I18N.xml"` and deploy it in a directory named `_"Resources"` inside the VS project directory, and then configure the file in the VS project as an embedded resource (i.e., set its Build Action to "Embedded resource" in the IDE, or add `<EmbeddedResource Include="Resources\I18N.xml" />` to an `ItemGroup` in the project file).

**Example (.csproj)**

```
<Project Sdk="Microsoft.NET.Sdk">
  ...
  <ItemGroup>
    <EmbeddedResource Include="Resources\I18N.xml" />
  </ItemGroup>
  ...
</Project>
```

**NOTE:** The companion utility `I18N.DotNet Tool` can be used to generate translations files optimized for deployment from the separate translations files used during development and during the translation process.



## 1.4 Advanced Usage (Internationalizing Applications)

### 1.4.1 Global Localizer

The static class `GlobalLocalizer` has the property `Localizer` which contains the global localizer. This instance is shared and can be conveniently used by all software components. In fact all the methods exposed by the `GlobalLocalizer` class are just convenience wrappers that call the global localizer.

The property `GlobalLocalizer.Localizer` is an instance of `AutoLoadLocalizer` that on first usage (if translations have not been previously loaded) tries to load the translations from an embedded resource identified by *Resources.I18N.xml* inside the entry (application) assembly using the current UI language as the target language.

The default behavior is just right for most use cases, but if the translations file is stored in an embedded resource with a different identifier, or in a separate file (e.g., installed alongside the application executable), one of the `LoadXML` methods can be invoked on the global localizer to load it (see Loading Translations).

**Non-Default usage Example (C#)**

```
void SetupI18N( string language, string directoryPath )
{
    GlobalLocalizer.Localizer.LoadXML( directoryPath + "/I18N.xml", language );
}
```

### 1.4.2 Local Localizers

Instances of `Localizer` can be created (local localizers), loaded with string translations, and then passed to software components for being used instead of the global localizer.

For most cases using the global localizer (and optionally contexts) is just enough, but local localizers can be useful for example to implement report generation in different languages than the application UI language (see Loading Translations and Specifying the Translation Target Language).

**Example (C#)**

```
Report GenerateReport( string language )
{
    var reportLocalizer = new Localizer();
    reportLocalizer.LoadXML( Assembly.GetExecutingAssembly(), "Reports.I18N.xml", language );
    return GenerateReport( reportLocalizer, new CultureInfo( language ) );
}

Report GenerateReport( ILocalizer localizer, CultureInfo culture )
{
    var report = new Report();
    report.AddEntry( localizer.Localize( $"Date: {DateTime.Now.ToString(culture)}" ) );
    ...
    return report;
}
```

### 1.4.3 Language Identifiers & Variants

Any arbitrary string can be used for identifying languages, although it is recommended to use identifiers formed by a ISO 639-1 alpha-2 language name (2-letter language codes, e.g., `en`, `es`), additionally followed by an hyphen and a ISO 3166-1 alpha-2 country/region name (e.g., `en-US`, `es-ES`).

Language identifiers are processed as case-insensitive (i.e., `fr-FR` is equivalent to `fr-fr`).

When using language identifiers formed by a primary code and a variant code separated by an hyphen (e.g., `en-us`, `es-es`), if a localized conversion for the language variant is not found then a conversion for the primary (base) language is tried too.

For example, when loading the translations on a `Localizer` created for the `en-gb` language, for each string to be translated a translation for the language `en-gb` will be searched first, and if not found then a translation for the language `en` will be searched next.

It is therefore recommended to:

- In source code:
  - Use primary-variant code (e.g., `_"en-us"_, _"es-es"_) as target language identifiers (i.e., as arguments to the LoadXML methods).`
- In translation files:
  - Use primary code (e.g., `_"en"_, _"fr"_) as translation language identifiers (i.e., as the lang attribute values of XML I18N.Entry.Value entries) for generic (non variant-specific) translations.`
  - Use primary code-variant (e.g., `_"en-gb"_, _"es-ar"_) as translation language identifiers (i.e., as the lang attribute values of XML I18N.Entry.Value entries) for variant-specific translations.`

### 1.4.4 String Format

Calls to `String.Format()` where the format string has to be internationalized can be replaced by a call to `GlobalLocalizer.LocalizeFormat()` / `ILocalizer.LocalizeFormat()`.

**Example (C#)** `String.Format( Localize( "Format string to be translated with value {0}" ), myVar );`  
 // is equivalent to  
`LocalizeFormat( "Format string to be translated with value {0}", myVar );`

### 1.4.5 Contexts

Sometimes the same source language string has different translations in different contexts (e.g., English `_"OK"↔_` should be translated in Spanish to `_"Aceptar"_) for a button label but to _"Correcto"_) for a successful outcome indication).`

Since the source language key is the same in both cases, context partitioning must be used, which affects the source code side and the translations file side.

**1.4.5.0.1 Context Partitioning in Source Code (I18N)** In source code, the context of the key can be explicitly indicated when the string is being internationalized by calling `GlobalLocalizer.Context()` / `ILocalizer.Context()` and passing it the context identifier, and then calling the localization methods on the returned context (which is an `ILocalizer``).

Contexts can be nested. A chain of successively nested contexts can be identified by joining their identifiers using the dot character (`'.'`) as a composite context identifier.

Translations in a context are searched hierarchically: if a translation is not found for the target language in its context (neither for the language variant nor the primary language), then a translation is searched again on its parent context (if it exists).

**Example (C#)** `Button.Label = Context( "GUI.Button" ).Localize( "OK" );`  
 // ...  
`TextBox.Text = Context( "GUI" ).Context( "Status" ).Localize( "OK" );`

**1.4.5.0.2 Context Partitioning in the Translation File (L10N)** Context partitioning is performed in the translations XML file using `Context` elements as children of the root element or nested within other `Context` elements. These elements must have an `id` attribute to indicate the context identifier (which can be a composite context identifier), and are containers for the `Entry` elements that define the translations for that context.

**Example** `<?xml version="1.0" encoding="utf-8"?>`

```
<I18N>
  <Entry>
    <Key>OK</Key>
    <Value lang="fr">O.K.</Value>
  </Entry>
  <Context id="GUI">
    <Context id="Button">
      <Entry>
        <Key>OK</Key>
        <Value lang="es">Aceptar</Value>
      </Entry>
    </Context>
    <Context id="Status">
      <Entry>
        <Key>OK</Key>
        <Value lang="es">Correcto</Value>
      </Entry>
    </Context>
  </Context>
</I18N>
```

## 1.4.6 Loading Translations

The translations can be loaded into a localizer implementing `ILoadableLocalizer` by different ways:

**1.4.6.0.1 From an Embedded Resource** The easiest way of using translation files is to embed them into an executable assembly (application or library), then load them into an `ILoadableLocalizer` instance using the `LoadXML` method indicating the assembly to load the embedded resource from and its identifier.

Note: The global localizer will automatically try to load the translations file from an embedded resource identified by *Resources.I18N.xml* in the entry assembly.

**Example (C#)**

```
void SetupI18N()
{
    GlobalLocalizer.Localizer.LoadXML( Assembly.GetExecutingAssembly(), "I18N.Translations.xml" );
}
```

**1.4.6.0.2 From a Standalone File** If the translations file is stored as a separate file (e.g., installed alongside the application executable), the `LoadXML` method can be invoked on an `ILoadableLocalizer` instance passing the path to the file.

**Example (C#)**

```
void SetupI18N()
{
    var programPath = Path.GetDirectoryName( Assembly.GetExecutingAssembly().Location );
    GlobalLocalizer.Localizer.LoadXML( programPath + "/I18N.xml" );
}
```

**1.4.6.0.3 From a Stream** When the translations file are neither stored as a file or embedded resource (e.g., downloading the translations from a remote server to local memory, obtaining the translations from a database), the `LoadXML` method can be invoked on an `ILoadableLocalizer` instance passing a `System.IO.Stream` object that must provide the file contents.

## 1.4.7 Specifying the Translation Target Language

When loading translations automatically or by means of explicit calls to `LoadXML` methods, the current UI language (obtained from `System.Globalization.CultureInfo.CurrentUICulture`) is used by default as the target language.

The usage of a different target language for the global localizer or a local localizer can be specified by different ways:

**1.4.7.0.1 Change the UI Language** During application startup, before any localization method is called, set `System.Globalization.CultureInfo.CurrentCulture` to the desired target language.

This approach is simple to make the global localizer use a specific language (e.g., use a language configured by the user), and it has the advantage that resources localized by other means may probably also use the same target language.

**Example (C#)**

```
using System.Globalization;
public class Program
{
    static void Main( string[] args )
    {
        if( args.Length >= 1 )
        {
            CultureInfo.CurrentCulture = new CultureInfo( args[0] );
        }
        ...
    }
}
```

When the application is already running, changing the UI language will have no immediate effect on the localizers which translations have already been loaded.

To enforce dynamic changes of the UI language to take effect, instances of `AutoLoadLocalizer` (like the global localizer) must be manually forced to reload its translations.

**Example (C#)**

```
void SetupI18N( string language )
{
    CultureInfo.CurrentCulture = new CultureInfo( language );
    GlobalLocalizer.Localizer.Load( null );
}
```

**NOTE:** It may also be useful to set `System.Globalization.CultureInfo.CurrentCulture`, `System.Globalization.CultureInfo.DefaultThreadCurrentCulture`, and/or `System.Globalization.CultureInfo.DefaultThreadCurrentCulture`.

**1.4.7.0.2 AutoLoadLocalizer.Load Method Parameter** The `AutoLoadLocalizer` class provides a `Load` method that accepts the target language as a parameter.

The `AutoLoadLocalizer.Load` method can be called during application startup or during runtime to load/reload the translations from the embedded resource for a specific language.

**Example (C#)**

```
void SetupI18N( string language )
{
    GlobalLocalizer.Localizer.Load( language );
}
```

**1.4.7.0.3 ILoadableLocalizer.LoadXML Methods Parameter** The `LoadXML` methods of `ILoadableLocalizer` accept the target language as an optional parameter.

The `ILoadableLocalizer.LoadXML` methods can be called during application startup or during runtime to load/reload the translations for a specific language.

**Example (C#)**

```
void SetupI18N( string language )
{
    var programPath = Path.GetDirectoryName( Assembly.GetExecutingAssembly().Location );
    GlobalLocalizer.Localizer.LoadXML( programPath + "/I18N.xml", language );
}
```

## 1.5 Advanced Usage (Internationalizing Libraries)

### 1.5.1 Library Localizers

The global localizer is convenient for usage in applications (i.e., which are implemented in the entry assembly), but libraries should not use the global localizer because they would depend on the application to load the translations for its internationalized strings or risk the application discarding the translations if trying to load them automatically during library initialization.

For libraries the easiest solution is to define their own "global" localizer as a static property inside a static class, similar to the `GlobalLocalizer` class but only intended for the scope of the library.

This library localizer can be initialized using an instance of `AutoLoadLocalizer`, which is a special localizer that automatically loads the translations file from an embedded resource.

The static class can be declared with `internal` scope, or with `public` scope to allow applications to extend or replace the library localizer (e.g., to add more translations, or to change them).

Finally, the translations file for the library must be embedded in the library assembly as an embedded resource identified by *Resources.I18N.xml* (just like with an application), which the `AutoLoadLocalizer` instance will try to load by default.

**Library Localizer Implementation Example (C#)** `using I18N.DotNet;`

```
using System;
namespace ExampleLibrary
{
    public static class LibraryLocalizer
    {
        {
            public static ILocalizer Localizer { get; } = new AutoLoadLocalizer();
            internal static string Localize( PlainString text ) => Localizer.Localize( text );
            internal static string Localize( FormattableString text ) => Localizer.Localize( text );
        }
    }
}
```

**Library Localizer Usage Example (C#)** `using static ExampleLibrary.LibraryLocalizer;`

```
using System;
namespace ExampleLibrary
{
    public class ExampleClass
    {
        {
            public void SomeMethod()
            {
                Console.WriteLine( Localize( "Plain string to be translated" ) );
                Console.WriteLine( Localize( $"Interpolated string to be translated with value {i:X4}" ) );
            }
        }
    }
}
```

## 1.6 API Documentation

### 1.6.0.1 ILocalizer Interface

The `ILocalizer` interface represents classes which provide localization functionality to software components (i.e. perform string translations) for a single target language:

- `Localize` methods to translate strings, interpolated strings and collections of strings.
- `LocalizeFormat` method to format and translate strings.
- `Context` methods to access contexts and subcontexts (see `Contexts`).

### 1.6.0.2 ILoadableLocalizer Interface

The `ILoadableLocalizer` interface is an extension of `ILocalizer`` that represents localizer classes which provide functionality to load translations for a single target language from different sources:

- `LoadXML` method to load translations from a file in the filesystem.
- `LoadXML` method to load translations from a `Stream`.
- `LoadXML` method to load translations from an XML document ( `XDocument` ).
- `LoadXML` method to load translations from an embedded resource in an assembly.

### 1.6.0.3 Localizer Class

The `Localizer` class is a simple implementation of `ILoadableLocalizer`` which is capable of loading string translations for a single target language and then providing localization functionality.

### 1.6.0.4 AutoLoadLocalizer Class

The `AutoLoadLocalizer` class is an implementation of `ILoadableLocalizer`` that on first call of any of its localization methods (i.e., those specified by `ILocalizer``), loads automatically the translations from an embedded resource in an assembly using the current UI language as the target language (if translations have not been previously loaded).

The default parameters for the `AutoLoadLocalizer` constructor make the created instance load the translations file from an embedded resource identified by `Resources.I18N.xml` in the calling assembly (i.e., in the assembly that creates the instance).

A different resource identifier or assembly can be passed as parameters to the `AutoLoadLocalizer` constructor if necessary.

Additionally, this class provides:

- `Load` method to load/reload translations from the configured embedded resource for a given language.

## 1.6.1 Full API Documentation

You can browse the full API documentation for:

- [The last release \(stable\)](#)
- [Main branch \(unstable\)](#)

## Chapter 2

# Namespace Index

### 2.1 Package List

Here are the packages with brief descriptions (if available):

<a href="#">I18N</a> . . . . .	<a href="#">15</a>
<a href="#">I18N.DotNet</a> . . . . .	<a href="#">15</a>





## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Exception	
ILoadableLocalizer.ParseException . . . . .	45
GlobalLocalizer . . . . .	24
ILocalizer . . . . .	35
ILoadableLocalizer . . . . .	27
AutoLoadLocalizer . . . . .	17
Localizer . . . . .	39
PlainString . . . . .	46



## Chapter 4

# Class Index

### 4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">AutoLoadLocalizer</a>	Implementation of a localizer which configuration is automatically loaded from an embedded resource. . . . .	17
<a href="#">GlobalLocalizer</a>	Utility class for convenient access to localization functions. . . . .	24
<a href="#">ILoadableLocalizer</a>	<a href="#">Localizer</a> which translations can be loaded from different sources. . . . .	27
<a href="#">ILocalizer</a>	Converter of strings from a language-neutral value to its corresponding language-specific localization. . . . .	35
<a href="#">Localizer</a>	Converter of strings from a language-neutral value to its corresponding language-specific localization. . . . .	39
<a href="#">ILoadableLocalizer.ParseException</a>	Exception thrown when a localization file cannot be parsed properly. . . . .	45
<a href="#">PlainString</a>	Represents just a string. This class is used to allow interpolated strings to preferably be passed as FormattableString instead of string to methods that overload both types. . . . .	46



## Chapter 5

# Namespace Documentation

### 5.1 I18N Namespace Reference

#### Namespaces

- namespace [DotNet](#)

### 5.2 I18N.DotNet Namespace Reference

#### Classes

- class [AutoLoadLocalizer](#)  
*Implementation of a localizer which configuration is automatically loaded from an embedded resource.*
- class [GlobalLocalizer](#)  
*Utility class for convenient access to localization functions.*
- interface [ILoadableLocalizer](#)  
*[Localizer](#) which translations can be loaded from different sources.*
- interface [ILocalizer](#)  
*Converter of strings from a language-neutral value to its corresponding language-specific localization.*
- class [Localizer](#)  
*Converter of strings from a language-neutral value to its corresponding language-specific localization.*
- class [PlainString](#)  
*Represents just a string. This class is used to allow interpolated strings to preferably be passed as FormattableString instead of string to methods that overload both types.*



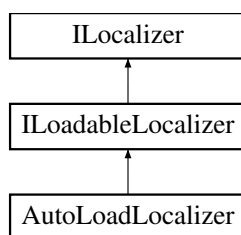
## Chapter 6

# Class Documentation

### 6.1 AutoLoadLocalizer Class Reference

Implementation of a localizer which configuration is automatically loaded from an embedded resource.

Inheritance diagram for AutoLoadLocalizer:



#### Public Member Functions

- [AutoLoadLocalizer](#) (string resourceName=[DEFAULT\\_RESOURCE\\_NAME](#), Assembly? assembly=null)  
*Constructor.*
- string [Localize](#) ([PlainString](#) text)  
*Localizes a string.*  
*Converts the language-neutral string text to its corresponding language-specific localized value.*
- string [Localize](#) ([FormattableString](#) frmtText)  
*Localizes an interpolated string.*  
*Converts the composite format string of the language-neutral formattable string frmtText ( e.g.an interpolated string) to its corresponding language-specific localized composite format value, and then generates the result by formatting the localized composite format value along with the frmtText arguments by using the formatting conventions of the current culture.*
- [IEnumerable](#)< string > [Localize](#) ([IEnumerable](#)< string > texts)  
*Localizes multiple strings.*  
*Converts the language-neutral strings in texts to their corresponding language-specific localized values.*
- string [LocalizeFormat](#) (string format, params object[] args)  
*Localizes and then formats a string.*  
*Converts the language-neutral format string format to its corresponding language-specific localized format value, and then generates the result by formatting the localized format value along with the args arguments by using the formatting conventions of the current culture.*
- [ILocalizer Context](#) (string contextId)

*Gets the localizer for a context in the current localizer.*

*Contexts are used to disambiguate the conversion of the same language-neutral string to different language-specific strings depending on the context where the conversion is performed.*

- [ILocalizer Context](#) (IEnumerable< string > splitContextIds)

*Gets the localizer for a context in the current localizer.*

*Contexts are used to disambiguate the conversion of the same language-neutral string to different language-specific strings depending on the context where the conversion is performed.*

- void [LoadXML](#) (string filepath, string? language=null, bool merge=false)

*Loads a localization configuration from a file in XML format.*

#### Parameters

filepath	<i>Path to the localization configuration file in XML format</i>
language	<i>Name, code or identifier for the target language of translations, or <code>null</code> to use the current UI language (obtained from <code>System.Globalization.CultureInfo.CurrentUICulture</code>)</i>
merge	<i>Replaces the current translations with the loaded ones when <code>&lt;c&gt;</code> false, otherwise merges both (existing translations are overridden with loaded ones).</i>

#### Exceptions

<a href="#">ParseException</a>	<i>Thrown when the input file cannot be parsed properly.</i>
--------------------------------	--

- void [LoadXML](#) (Stream stream, string? language=null, bool merge=false)

*Loads a localization configuration from a stream in XML format.*

#### Parameters

stream	<i>Stream with the localization configuration in XML format</i>
language	<i>Name, code or identifier for the target language of translations, or <code>null</code> to use the current UI language (obtained from <code>System.Globalization.CultureInfo.CurrentUICulture</code>)</i>
merge	<i>Replaces the current translations with the loaded ones when <code>&lt;c&gt;</code> false, otherwise merges both (existing translations are overridden with loaded ones).</i>

#### Exceptions

<a href="#">ParseException</a>	<i>Thrown when the stream contents cannot be parsed properly.</i>
--------------------------------	---

- void [LoadXML](#) (XDocument doc, string? language=null, bool merge=false)

*Loads a localization configuration from a XML document.*

#### Parameters

doc	<i>XML document with the localization configuration</i>
language	<i>Name, code or identifier for the target language of translations, or <code>null</code> to use the current UI language (obtained from <code>System.Globalization.CultureInfo.CurrentUICulture</code>)</i>
merge	<i>Replaces the current translations with the loaded ones when <code>&lt;c&gt;</code> false, otherwise merges both (existing translations are overridden with loaded ones).</i>

#### Exceptions

<a href="#">ParseException</a>	<i>Thrown when the input document cannot be parsed properly.</i>
--------------------------------	--

- void [LoadXML](#) (Assembly assembly, string resourceName, string? language=null, bool merge=false)



*Loads a localization configuration from an XML text embedded as a resource in the given assembly.*

#### Parameters

assembly	Assembly that contains the embedded XML text
resourceName	Name of the embedded resource for the XML text
language	Name, code or identifier for the target language of translations, or <code>null</code> to use the current UI language (obtained from <code>System.Globalization.CultureInfo.CurrentCulture</code> )
merge	Replaces the current translations with the loaded ones when <code>&lt;c&gt; false</code> , otherwise merges both (existing translations are overridden with loaded ones).

#### Exceptions

<a href="#">ParseException</a>	Thrown when the embedded resource contents cannot be parsed properly.
<code>InvalidOperationException</code>	Thrown when the embedded resource could not be found in the given assembly

- void [Load](#) (string? language, bool merge=false)

*Loads the localization configuration from the embedded resource using the given language.*

## Static Public Attributes

- const string [DEFAULT\\_RESOURCE\\_NAME](#) = "Resources.l18N.xml"

### 6.1.1 Detailed Description

Implementation of a localizer which configuration is automatically loaded from an embedded resource.

### 6.1.2 Constructor & Destructor Documentation

#### 6.1.2.1 AutoLoadLocalizer()

```
AutoLoadLocalizer (
    string resourceName = DEFAULT\_RESOURCE\_NAME,
    Assembly? assembly = null )
```

Constructor.

#### Parameters

resourceName	Name of the embedded resource for the XML text
assembly	Assembly that contains the embedded XML text (the calling assembly will be used if <code>null</code> )

### 6.1.3 Member Function Documentation

### 6.1.3.1 Context() [1/2]

```
ILocalizer Context (
    IEnumerable< string > splitContextIds )
```

Gets the localizer for a context in the current localizer.

Contexts are used to disambiguate the conversion of the same language-neutral string to different language-specific strings depending on the context where the conversion is performed.

Implements [ILocalizer](#).

### 6.1.3.2 Context() [2/2]

```
ILocalizer Context (
    string contextId )
```

Gets the localizer for a context in the current localizer.

Contexts are used to disambiguate the conversion of the same language-neutral string to different language-specific strings depending on the context where the conversion is performed.

Implements [ILocalizer](#).

### 6.1.3.3 Load()

```
void Load (
    string? language,
    bool merge = false )
```

Loads the localization configuration from the embedded resource using the given language.

If this method is not called explicitly, the translations are automatically loaded from the embedded resource using the current UI language when a localization method is called for the first time.

#### Parameters

<i>language</i>	Name, code or identifier for the target language of translations, or <code>null</code> to use the current UI language (obtained from <code>System.Globalization.CultureInfo.CurrentUICulture</code> )
<i>merge</i>	Replaces the current translations with the loaded ones when <code>&lt;c&gt;false</code> , otherwise merges both (existing translations are overridden with loaded ones).

#### Exceptions

<a href="#">ILoadableLocalizer.ParseException</a>	Thrown when the embedded resource contents cannot be parsed properly.
<a href="#">InvalidOperationException</a>	Thrown when the embedded resource could not be found

**6.1.3.4 LoadXML() [1/4]**

```
void LoadXML (
    Assembly assembly,
    string resourceName,
    string? language = null,
    bool merge = false )
```

Loads a localization configuration from an XML text embedded as a resource in the given assembly.

**Parameters**

<i>assembly</i>	Assembly that contains the embedded XML text
<i>resourceName</i>	Name of the embedded resource for the XML text
<i>language</i>	Name, code or identifier for the target language of translations, or <code>null</code> to use the current UI language (obtained from <code>System.Globalization.CultureInfo.CurrentUICulture</code> )
<i>merge</i>	Replaces the current translations with the loaded ones when <code>&lt;c&gt;false</code> , otherwise merges both (existing translations are overridden with loaded ones).

**Exceptions**

<a href="#"><i>ParseException</i></a>	Thrown when the embedded resource contents cannot be parsed properly.
<a href="#"><i>InvalidOperationException</i></a>	Thrown when the embedded resource could not be found in the given assembly

Implements [ILoadableLocalizer](#).

**6.1.3.5 LoadXML() [2/4]**

```
void LoadXML (
    Stream stream,
    string? language = null,
    bool merge = false )
```

Loads a localization configuration from a stream in XML format.

**Parameters**

<i>stream</i>	Stream with the localization configuration in XML format
<i>language</i>	Name, code or identifier for the target language of translations, or <code>null</code> to use the current UI language (obtained from <code>System.Globalization.CultureInfo.CurrentUICulture</code> )
<i>merge</i>	Replaces the current translations with the loaded ones when <code>&lt;c&gt;false</code> , otherwise merges both (existing translations are overridden with loaded ones).

## Exceptions

<a href="#">ParseException</a>	Thrown when the stream contents cannot be parsed properly.
--------------------------------	--

Implements [ILoadableLocalizer](#).

**6.1.3.6 LoadXML()** [3/4]

```
void LoadXML (
    string filepath,
    string? language = null,
    bool merge = false )
```

Loads a localization configuration from a file in XML format.

## Parameters

<i>filepath</i>	Path to the localization configuration file in XML format
<i>language</i>	Name, code or identifier for the target language of translations, or <code>null</code> to use the current UI language (obtained from <code>System.Globalization.CultureInfo.CurrentUICulture</code> )
<i>merge</i>	Replaces the current translations with the loaded ones when <code>&lt;c&gt;&gt;false</code> , otherwise merges both (existing translations are overridden with loaded ones).

## Exceptions

<a href="#">ParseException</a>	Thrown when the input file cannot be parsed properly.
--------------------------------	---

Implements [ILoadableLocalizer](#).

**6.1.3.7 LoadXML()** [4/4]

```
void LoadXML (
    XDocument doc,
    string? language = null,
    bool merge = false )
```

Loads a localization configuration from a XML document.

## Parameters

<i>doc</i>	XML document with the localization configuration
<i>language</i>	Name, code or identifier for the target language of translations, or <code>null</code> to use the current UI language (obtained from <code>System.Globalization.CultureInfo.CurrentUICulture</code> )
<i>merge</i>	Replaces the current translations with the loaded ones when <code>&lt;c&gt;&gt;false</code> , otherwise merges both (existing translations are overridden with loaded ones).

## Exceptions

<a href="#">ParseException</a>	Thrown when the input document cannot be parsed properly.
--------------------------------	---

Implements [ILoadableLocalizer](#).

### 6.1.3.8 Localize() [1/3]

```
string Localize (
    FormattableString fmtText )
```

Localizes an interpolated string.

Converts the composite format string of the language-neutral formattable string *fmtText* ( e.g.an interpolated string) to its corresponding language-specific localized composite format value, and then generates the result by formatting the localized composite format value along with the *fmtText* arguments by using the formatting conventions of the current culture.

Implements [ILocalizer](#).

### 6.1.3.9 Localize() [2/3]

```
IEnumerable< string > Localize (
    IEnumerable< string > texts )
```

Localizes multiple strings.

Converts the language-neutral strings in *texts* to their corresponding language-specific localized values.

Implements [ILocalizer](#).

### 6.1.3.10 Localize() [3/3]

```
string Localize (
    PlainString text )
```

Localizes a string.

Converts the language-neutral string *text* to its corresponding language-specific localized value.

Implements [ILocalizer](#).

### 6.1.3.11 LocalizeFormat()

```
string LocalizeFormat (
    string format,
    params object[] args )
```

Localizes and then formats a string.

Converts the language-neutral format string *format* to its corresponding language-specific localized format value, and then generates the result by formatting the localized format value along with the *args* arguments by using the formatting conventions of the current culture.

Implements [ILocalizer](#).

## 6.1.4 Member Data Documentation

### 6.1.4.1 DEFAULT\_RESOURCE\_NAME

```
const string DEFAULT_RESOURCE_NAME = "Resources.I18N.xml" [static]
```

Default identifier for the embedded resource containing the translations.

## 6.2 GlobalLocalizer Class Reference

Utility class for convenient access to localization functions.

### Static Public Member Functions

- static string [Localize](#) ([PlainString](#) text)  
*Localizes a string using the global localizer.*
- static string [Localize](#) ([FormattableString](#) fmtText)  
*Localizes an interpolated string using the global localizer.*
- static IEnumerable< string > [Localize](#) (IEnumerable< string > texts)  
*Localizes multiple strings.*
- static string [LocalizeFormat](#) (string format, params object[] args)  
*Localizes and then formats a string using the global localizer.*
- static [ILocalizer Context](#) (string contextId)  
*Gets a context in the global localizer.*

### Properties

- static [ILoadableLocalizer Localizer](#) = new [AutoLoadLocalizer](#)() [get]

### 6.2.1 Detailed Description

Utility class for convenient access to localization functions.

### 6.2.2 Member Function Documentation

#### 6.2.2.1 Context()

```
static ILocalizer Context (  
    string contextId ) [static]
```

Gets a context in the global localizer.

See also

[ILocalizer.Context\(string\)](#)

Parameters

<i>contextId</i>	Identifier of the context
------------------	---------------------------

Returns

[Localizer](#) for the given context

#### 6.2.2.2 Localize() [1/3]

```
static string Localize (  
    FormattableString frmtText ) [static]
```

Localizes an interpolated string using the global localizer.

See also

[ILocalizer.Localize\(FormattableString\)](#)

Parameters

<i>frmtText</i>	Language-neutral formattable string
-----------------	-------------------------------------

**Returns**

Formatted string generated from the language-specific localized format string if found, or generated from *fmtText* otherwise

**6.2.2.3 Localize()** [2/3]

```
static IEnumerable< string > Localize (
    IEnumerable< string > texts ) [static]
```

Localizes multiple strings.

**See also**

[ILocalizer.Localize\(IEnumerable<string>\)](#)

**Parameters**

<i>texts</i>	Array of language-neutral strings
--------------	-----------------------------------

**Returns**

Array with the language-specific localized strings if found, or the language-neutral string otherwise

**6.2.2.4 Localize()** [3/3]

```
static string Localize (
    PlainString text ) [static]
```

Localizes a string using the global localizer.

**See also**

[ILocalizer.Localize\(PlainString\)](#)

**Parameters**

<i>text</i>	Language-neutral string
-------------	-------------------------

**Returns**

Language-specific localized string if found, or *text* otherwise



### 6.2.2.5 LocalizeFormat()

```
static string LocalizeFormat (
    string format,
    params object[] args ) [static]
```

Localizes and then formats a string using the global localizer.

#### See also

`ILocalizer.LocalizeFormat(string, object[])`

#### Parameters

<i>format</i>	Language-neutral format string
<i>args</i>	Arguments for the format string

#### Returns

Formatted string generated from the language-specific localized format string if found, or generated from *format* otherwise

## 6.2.3 Property Documentation

### 6.2.3.1 Localizer

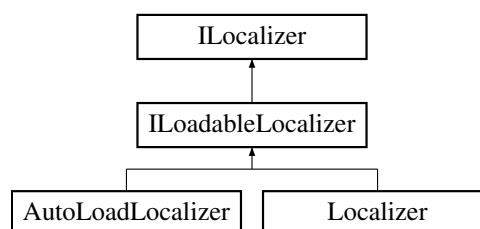
```
ILoadableLocalizer Localizer = new AutoLoadLocalizer() [static], [get]
```

Global localizer.

## 6.3 ILoadableLocalizer Interface Reference

[Localizer](#) which translations can be loaded from different sources.

Inheritance diagram for ILoadableLocalizer:



## Classes

- class [ParseException](#)

*Exception thrown when a localization file cannot be parsed properly.*

## Public Member Functions

- void [LoadXML](#) (string filepath, string? language=null, bool merge=false)  
*Loads a localization configuration from a file in XML format.*
- void [LoadXML](#) (Stream stream, string? language=null, bool merge=false)  
*Loads a localization configuration from a stream in XML format.*
- void [LoadXML](#) (XDocument doc, string? language=null, bool merge=false)  
*Loads a localization configuration from a XML document.*
- void [LoadXML](#) (Assembly assembly, string resourceName, string? language=null, bool merge=false)  
*Loads a localization configuration from an XML text embedded as a resource in the given assembly.*
- string [Localize](#) ([PlainString](#) text)  
*Localizes a string.*
- string [Localize](#) (FormattableString frmtText)  
*Localizes an interpolated string.*
- IEnumerable< string > [Localize](#) (IEnumerable< string > texts)  
*Localizes multiple strings.*
- string [LocalizeFormat](#) (string format, params object[] args)  
*Localizes and then formats a string.*
- [ILocalizer Context](#) (string contextId)  
*Gets the localizer for a context in the current localizer.*
- [ILocalizer Context](#) (IEnumerable< string > splitContextIds)  
*Gets the localizer for a context in the current localizer.*

### 6.3.1 Detailed Description

[Localizer](#) which translations can be loaded from different sources.

### 6.3.2 Member Function Documentation

#### 6.3.2.1 Context() [1/2]

```
ILocalizer Context (
    IEnumerable< string > splitContextIds ) [inherited]
```

Gets the localizer for a context in the current localizer.

Contexts are used to disambiguate the conversion of the same language-neutral string to different language-specific strings depending on the context where the conversion is performed.

## Parameters

<i>splitContextIds</i>	Chain of context identifiers in split form
------------------------	--

## Returns

[Localizer](#) for the given context

Implemented in [AutoLoadLocalizer](#), and [Localizer](#).

**6.3.2.2 Context()** [2/2]

```
ILocalizer Context (
    string contextId ) [inherited]
```

Gets the localizer for a context in the current localizer.

Contexts are used to disambiguate the conversion of the same language-neutral string to different language-specific strings depending on the context where the conversion is performed.

Contexts can be nested. The context identifier can identify a chain of nested contexts by separating their identifiers with the '.' character (left = outermost / right = innermost).

## Parameters

<i>contextId</i>	Identifier of the context
------------------	---------------------------

## Returns

[Localizer](#) for the given context

Implemented in [AutoLoadLocalizer](#), and [Localizer](#).

**6.3.2.3 LoadXML()** [1/4]

```
void LoadXML (
    Assembly assembly,
    string resourceName,
    string? language = null,
    bool merge = false )
```

Loads a localization configuration from an XML text embedded as a resource in the given assembly.

## Parameters

<i>assembly</i>	Assembly that contains the embedded XML text
<i>resourceName</i>	Name of the embedded resource for the XML text
<i>language</i>	Name, code or identifier for the target language of translations, or <code>null</code> to use the current UI language (obtained from <code>System.Globalization.CultureInfo.CurrentUICulture</code> )
<i>merge</i>	Replaces the current translations with the loaded ones when <code>&lt;c&gt;false</code> , otherwise merges both (existing translations are overridden with loaded ones).

## Exceptions

<a href="#"><i>ParseException</i></a>	Thrown when the embedded resource contents cannot be parsed properly.
<i>InvalidOperationException</i>	Thrown when the embedded resource could not be found in the given assembly

Implemented in [AutoLoadLocalizer](#), and [Localizer](#).

**6.3.2.4 LoadXML() [2/4]**

```
void LoadXML (
    Stream stream,
    string? language = null,
    bool merge = false )
```

Loads a localization configuration from a stream in XML format.

## Parameters

<i>stream</i>	Stream with the localization configuration in XML format
<i>language</i>	Name, code or identifier for the target language of translations, or <code>null</code> to use the current UI language (obtained from <code>System.Globalization.CultureInfo.CurrentUICulture</code> )
<i>merge</i>	Replaces the current translations with the loaded ones when <code>&lt;c&gt;false</code> , otherwise merges both (existing translations are overridden with loaded ones).

## Exceptions

<a href="#"><i>ParseException</i></a>	Thrown when the stream contents cannot be parsed properly.
---------------------------------------	--

Implemented in [AutoLoadLocalizer](#), and [Localizer](#).

**6.3.2.5 LoadXML() [3/4]**

```
void LoadXML (
    string filepath,
```

```
string? language = null,  
bool merge = false )
```

Loads a localization configuration from a file in XML format.

## Parameters

<i>filepath</i>	Path to the localization configuration file in XML format
<i>language</i>	Name, code or identifier for the target language of translations, or <code>null</code> to use the current UI language (obtained from <code>System.Globalization.CultureInfo.CurrentUICulture</code> )
<i>merge</i>	Replaces the current translations with the loaded ones when <code>&lt;c&gt;false</code> , otherwise merges both (existing translations are overridden with loaded ones).

## Exceptions

<a href="#">ParseException</a>	Thrown when the input file cannot be parsed properly.
--------------------------------	---

Implemented in [AutoLoadLocalizer](#), and [Localizer](#).

**6.3.2.6 LoadXML()** [4/4]

```
void LoadXML (
    XDocument doc,
    string? language = null,
    bool merge = false )
```

Loads a localization configuration from a XML document.

## Parameters

<i>doc</i>	XML document with the localization configuration
<i>language</i>	Name, code or identifier for the target language of translations, or <code>null</code> to use the current UI language (obtained from <code>System.Globalization.CultureInfo.CurrentUICulture</code> )
<i>merge</i>	Replaces the current translations with the loaded ones when <code>&lt;c&gt;false</code> , otherwise merges both (existing translations are overridden with loaded ones).

## Exceptions

<a href="#">ParseException</a>	Thrown when the input document cannot be parsed properly.
--------------------------------	---

Implemented in [AutoLoadLocalizer](#), and [Localizer](#).

**6.3.2.7 Localize()** [1/3]

```
string Localize (
    FormattableString fmtText ) [inherited]
```

Localizes an interpolated string.

Converts the composite format string of the language-neutral formattable string *fmtText* ( e.g.an interpolated string) to its corresponding language-specific localized composite format value, and then generates the result by formatting the localized composite format value along with the *fmtText* arguments by using the formatting conventions of the current culture.

## Parameters

<i>fmtText</i>	Language-neutral formattable string
----------------	-------------------------------------

## Returns

Formatted string generated from the language-specific localized format string if found, or generated from *fmtText* otherwise

Implemented in [AutoLoadLocalizer](#), and [Localizer](#).

**6.3.2.8 Localize()** [2/3]

```
IEnumerable< string > Localize (
    IEnumerable< string > texts ) [inherited]
```

Localizes multiple strings.

Converts the language-neutral strings in *texts* to their corresponding language-specific localized values.

## Parameters

<i>texts</i>	Language-neutral strings
--------------	--------------------------

## Returns

Implemented in [AutoLoadLocalizer](#), and [Localizer](#).

**6.3.2.9 Localize()** [3/3]

```
string Localize (
    PlainString text ) [inherited]
```

Localizes a string.

Converts the language-neutral string *text* to its corresponding language-specific localized value.

## Parameters

<i>text</i>	Language-neutral string
-------------	-------------------------



**Returns**

Language-specific localized string if found, or *text* otherwise

Implemented in [AutoLoadLocalization](#), and [Localization](#).

**6.3.2.10 LocalizeFormat()**

```
string LocalizeFormat (
    string format,
    params object[] args ) [inherited]
```

Localizes and then formats a string.

Converts the language-neutral format string *format* to its corresponding language-specific localized format value, and then generates the result by formatting the localized format value along with the *args* arguments by using the formatting conventions of the current culture.

**Parameters**

<i>format</i>	Language-neutral format string
<i>args</i>	Arguments for the format string

**Returns**

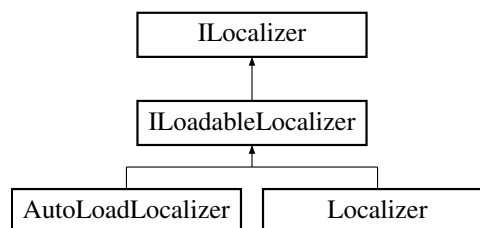
Formatted string generated from the language-specific localized format string if found, or generated from *format* otherwise

Implemented in [AutoLoadLocalization](#).

## 6.4 ILocalization Interface Reference

Converter of strings from a language-neutral value to its corresponding language-specific localization.

Inheritance diagram for ILocalization:



## Public Member Functions

- string [Localize](#) ([PlainString](#) text)  
*Localizes a string.*
- string [Localize](#) ([FormattableString](#) frmtText)  
*Localizes an interpolated string.*
- string [LocalizeFormat](#) (string format, params object[] args)  
*Localizes and then formats a string.*
- [IEnumerable< string > Localize](#) ([IEnumerable< string > texts](#))  
*Localizes multiple strings.*
- [ILocalizer Context](#) (string contextId)  
*Gets the localizer for a context in the current localizer.*
- [ILocalizer Context](#) ([IEnumerable< string > splitContextIds](#))  
*Gets the localizer for a context in the current localizer.*

### 6.4.1 Detailed Description

Converter of strings from a language-neutral value to its corresponding language-specific localization.

### 6.4.2 Member Function Documentation

#### 6.4.2.1 Context() [1/2]

```
ILocalizer Context (
    IEnumerable< string > splitContextIds )
```

Gets the localizer for a context in the current localizer.

Contexts are used to disambiguate the conversion of the same language-neutral string to different language-specific strings depending on the context where the conversion is performed.

#### Parameters

<i>splitContextIds</i>	Chain of context identifiers in split form
------------------------	--

#### Returns

[Localizer](#) for the given context

Implemented in [AutoLoadLocalizer](#), and [Localizer](#).

### 6.4.2.2 Context() [2/2]

```
ILocalization Context (
    string contextId )
```

Gets the localizer for a context in the current localizer.

Contexts are used to disambiguate the conversion of the same language-neutral string to different language-specific strings depending on the context where the conversion is performed.

Contexts can be nested. The context identifier can identify a chain of nested contexts by separating their identifiers with the '.' character (left = outermost / right = innermost).

#### Parameters

<i>contextId</i>	Identifier of the context
------------------	---------------------------

#### Returns

[Localizer](#) for the given context

Implemented in [AutoLoadLocalization](#), and [Localization](#).

### 6.4.2.3 Localize() [1/3]

```
string Localize (
    FormattableString fmtText )
```

Localizes an interpolated string.

Converts the composite format string of the language-neutral formattable string *fmtText* ( e.g.an interpolated string) to its corresponding language-specific localized composite format value, and then generates the result by formatting the localized composite format value along with the *fmtText* arguments by using the formatting conventions of the current culture.

#### Parameters

<i>fmtText</i>	Language-neutral formattable string
----------------	-------------------------------------

#### Returns

Formatted string generated from the language-specific localized format string if found, or generated from *fmtText* otherwise

Implemented in [AutoLoadLocalization](#), and [Localization](#).

#### 6.4.2.4 Localize() [2/3]

```
IEnumerable< string > Localize (
    IEnumerable< string > texts )
```

Localizes multiple strings.

Converts the language-neutral strings in *texts* to their corresponding language-specific localized values.

##### Parameters

<i>texts</i>	Language-neutral strings
--------------	--------------------------

##### Returns

Implemented in [AutoLoadLocalizer](#), and [Localizer](#).

#### 6.4.2.5 Localize() [3/3]

```
string Localize (
    PlainString text )
```

Localizes a string.

Converts the language-neutral string *text* to its corresponding language-specific localized value.

##### Parameters

<i>text</i>	Language-neutral string
-------------	-------------------------

##### Returns

Language-specific localized string if found, or *text* otherwise

Implemented in [AutoLoadLocalizer](#), and [Localizer](#).

#### 6.4.2.6 LocalizeFormat()

```
string LocalizeFormat (
    string format,
    params object[] args )
```

Localizes and then formats a string.

Converts the language-neutral format string *format* to its corresponding language-specific localized format value, and then generates the result by formatting the localized format value along with the *args* arguments by using the formatting conventions of the current culture.

## Parameters

<i>format</i>	Language-neutral format string
<i>args</i>	Arguments for the format string

## Returns

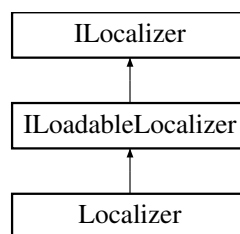
Formatted string generated from the language-specific localized format string if found, or generated from *format* otherwise

Implemented in [AutoLoadLocalizer](#).

## 6.5 Localizer Class Reference

Converter of strings from a language-neutral value to its corresponding language-specific localization.

Inheritance diagram for Localizer:



### Public Member Functions

- [Localizer](#) ()  
*Default constructor.*
- string [Localize](#) (PlainString text)  
*Localizes a string.*  
*Converts the language-neutral string text to its corresponding language-specific localized value.*
- string [Localize](#) (FormattableString frmtText)  
*Localizes an interpolated string.*  
*Converts the composite format string of the language-neutral formattable string frmtText ( e.g.an interpolated string) to its corresponding language-specific localized composite format value, and then generates the result by formatting the localized composite format value along with the frmtText arguments by using the formatting conventions of the current culture.*
- string [LocalizeFormat](#) (string format, params object?[] args)
- IEnumerable< string > [Localize](#) (IEnumerable< string > texts)  
*Localizes multiple strings.*  
*Converts the language-neutral strings in texts to their corresponding language-specific localized values.*
- [Localizer Context](#) (string contextId)  
*Gets the localizer for a context in the current localizer.*  
*Contexts are used to disambiguate the conversion of the same language-neutral string to different language-specific strings depending on the context where the conversion is performed.*
- [Localizer Context](#) (IEnumerable< string > splitContextIds)  
*Gets the localizer for a context in the current localizer.*  
*Contexts are used to disambiguate the conversion of the same language-neutral string to different language-specific strings depending on the context where the conversion is performed.*
- void [LoadXML](#) (string filepath, string? language=null, bool merge=false)  
*Loads a localization configuration from a file in XML format.*

*Parameters*

filepath	Path to the localization configuration file in XML format
language	Name, code or identifier for the target language of translations, or <i>null</i> to use the current UI language (obtained from <i>System.Globalization.CultureInfo.CurrentUICulture</i> )
merge	Replaces the current translations with the loaded ones when <i>&lt;c&gt;false</i> , otherwise merges both (existing translations are overridden with loaded ones).

*Exceptions*

<a href="#">ParseException</a>	Thrown when the input file cannot be parsed properly.
--------------------------------	---

- void [LoadXML](#) (Stream stream, string? language=null, bool merge=false)

Loads a localization configuration from a stream in XML format.

*Parameters*

stream	Stream with the localization configuration in XML format
language	Name, code or identifier for the target language of translations, or <i>null</i> to use the current UI language (obtained from <i>System.Globalization.CultureInfo.CurrentUICulture</i> )
merge	Replaces the current translations with the loaded ones when <i>&lt;c&gt;false</i> , otherwise merges both (existing translations are overridden with loaded ones).

*Exceptions*

<a href="#">ParseException</a>	Thrown when the stream contents cannot be parsed properly.
--------------------------------	--

- void [LoadXML](#) (XDocument doc, string? language=null, bool merge=false)

Loads a localization configuration from a XML document.

*Parameters*

doc	XML document with the localization configuration
language	Name, code or identifier for the target language of translations, or <i>null</i> to use the current UI language (obtained from <i>System.Globalization.CultureInfo.CurrentUICulture</i> )
merge	Replaces the current translations with the loaded ones when <i>&lt;c&gt;false</i> , otherwise merges both (existing translations are overridden with loaded ones).

*Exceptions*

<a href="#">ParseException</a>	Thrown when the input document cannot be parsed properly.
--------------------------------	---

- void [LoadXML](#) (Assembly assembly, string resourceName, string? language=null, bool merge=false)

Loads a localization configuration from an XML text embedded as a resource in the given assembly.

*Parameters*

assembly	Assembly that contains the embedded XML text
resourceName	Name of the embedded resource for the XML text
language	Name, code or identifier for the target language of translations, or <i>null</i> to use the current UI language (obtained from <i>System.Globalization.CultureInfo.CurrentUICulture</i> )
merge	Replaces the current translations with the loaded ones when <i>&lt;c&gt;false</i> , otherwise merges both (existing translations are overridden with loaded ones).

*Exceptions*

<a href="#">ParseException</a>	<i>Thrown when the embedded resource contents cannot be parsed properly.</i>
<code>InvalidOperationException</code>	<i>Thrown when the embedded resource could not be found in the given assembly</i>

- string [LocalizeFormat](#) (string format, params object[] args)  
*Localizes and then formats a string.*

### 6.5.1 Detailed Description

Converter of strings from a language-neutral value to its corresponding language-specific localization.

### 6.5.2 Constructor & Destructor Documentation

#### 6.5.2.1 Localizer()

```
Localizer ( )
```

Default constructor.

The target language of translations is set to the current UI language (obtained from `CultureInfo.CurrentUICulture`).

### 6.5.3 Member Function Documentation

#### 6.5.3.1 Context() [1/2]

```
Localizer Context (
    IEnumerable< string > splitContextIds )
```

Gets the localizer for a context in the current localizer.

Contexts are used to disambiguate the conversion of the same language-neutral string to different language-specific strings depending on the context where the conversion is performed.

Implements [ILocalizer](#).

### 6.5.3.2 Context() [2/2]

```

Localizer Context (
    string contextId )

```

Gets the localizer for a context in the current localizer.

Contexts are used to disambiguate the conversion of the same language-neutral string to different language-specific strings depending on the context where the conversion is performed.

Implements [ILocalizer](#).

### 6.5.3.3 LoadXML() [1/4]

```

void LoadXML (
    Assembly assembly,
    string resourceName,
    string? language = null,
    bool merge = false )

```

Loads a localization configuration from an XML text embedded as a resource in the given assembly.

#### Parameters

<i>assembly</i>	Assembly that contains the embedded XML text
<i>resourceName</i>	Name of the embedded resource for the XML text
<i>language</i>	Name, code or identifier for the target language of translations, or <code>null</code> to use the current UI language (obtained from <code>System.Globalization.CultureInfo.CurrentUICulture</code> )
<i>merge</i>	Replaces the current translations with the loaded ones when <code>&lt;c&gt;false</code> , otherwise merges both (existing translations are overridden with loaded ones).

#### Exceptions

<a href="#">ParseException</a>	Thrown when the embedded resource contents cannot be parsed properly.
<a href="#">InvalidOperationException</a>	Thrown when the embedded resource could not be found in the given assembly

Implements [ILoadableLocalizer](#).

### 6.5.3.4 LoadXML() [2/4]

```

void LoadXML (
    Stream stream,
    string? language = null,
    bool merge = false )

```

Loads a localization configuration from a stream in XML format.



## Parameters

<i>stream</i>	Stream with the localization configuration in XML format
<i>language</i>	Name, code or identifier for the target language of translations, or <code>null</code> to use the current UI language (obtained from <code>System.Globalization.CultureInfo.CurrentUICulture</code> )
<i>merge</i>	Replaces the current translations with the loaded ones when <code>&lt;c&gt;false</code> , otherwise merges both (existing translations are overridden with loaded ones).

## Exceptions

<a href="#">ParseException</a>	Thrown when the stream contents cannot be parsed properly.
--------------------------------	--

Implements [ILoadableLocalizer](#).

### 6.5.3.5 LoadXML() [3/4]

```
void LoadXML (
    string filepath,
    string? language = null,
    bool merge = false )
```

Loads a localization configuration from a file in XML format.

## Parameters

<i>filepath</i>	Path to the localization configuration file in XML format
<i>language</i>	Name, code or identifier for the target language of translations, or <code>null</code> to use the current UI language (obtained from <code>System.Globalization.CultureInfo.CurrentUICulture</code> )
<i>merge</i>	Replaces the current translations with the loaded ones when <code>&lt;c&gt;false</code> , otherwise merges both (existing translations are overridden with loaded ones).

## Exceptions

<a href="#">ParseException</a>	Thrown when the input file cannot be parsed properly.
--------------------------------	---

Implements [ILoadableLocalizer](#).

### 6.5.3.6 LoadXML() [4/4]

```
void LoadXML (
    XDocument doc,
    string? language = null,
    bool merge = false )
```

Loads a localization configuration from a XML document.

## Parameters

<i>doc</i>	XML document with the localization configuration
<i>language</i>	Name, code or identifier for the target language of translations, or <code>null</code> to use the current UI language (obtained from <code>System.Globalization.CultureInfo.CurrentUICulture</code> )
<i>merge</i>	Replaces the current translations with the loaded ones when <code>&lt;c&gt;false</code> , otherwise merges both (existing translations are overridden with loaded ones).

## Exceptions

<a href="#">ParseException</a>	Thrown when the input document cannot be parsed properly.
--------------------------------	---

Implements [ILoadableLocalizer](#).

#### 6.5.3.7 Localize() [1/3]

```
string Localize (
    FormattableString frmtText )
```

Localizes an interpolated string.

Converts the composite format string of the language-neutral formattable string *frmtText* ( e.g.an interpolated string) to its corresponding language-specific localized composite format value, and then generates the result by formatting the localized composite format value along with the *frmtText* arguments by using the formatting conventions of the current culture.

Implements [ILocalizer](#).

#### 6.5.3.8 Localize() [2/3]

```
IEnumerable< string > Localize (
    IEnumerable< string > texts )
```

Localizes multiple strings.

Converts the language-neutral strings in *texts* to their corresponding language-specific localized values.

Implements [ILocalizer](#).

#### 6.5.3.9 Localize() [3/3]

```
string Localize (
    PlainString text )
```

Localizes a string.

Converts the language-neutral string *text* to its corresponding language-specific localized value.

Implements [ILocalizer](#).

**6.5.3.10 LocalizeFormat() [1/2]**

```
string LocalizeFormat (
    string format,
    params object?[] args )
```

**6.5.3.11 LocalizeFormat() [2/2]**

```
string LocalizeFormat (
    string format,
    params object[] args ) [inherited]
```

Localizes and then formats a string.

Converts the language-neutral format string *format* to its corresponding language-specific localized format value, and then generates the result by formatting the localized format value along with the *args* arguments by using the formatting conventions of the current culture.

**Parameters**

<i>format</i>	Language-neutral format string
<i>args</i>	Arguments for the format string

**Returns**

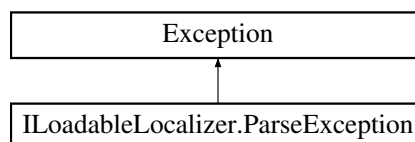
Formatted string generated from the language-specific localized format string if found, or generated from *format* otherwise

Implemented in [AutoLoadLocalizer](#).

**6.6 ILoadableLocalizer.ParseException Class Reference**

Exception thrown when a localization file cannot be parsed properly.

Inheritance diagram for ILoadableLocalizer.ParseException:

**Public Member Functions**

- [ParseException](#) (string message)  
*Constructor.*

### 6.6.1 Detailed Description

Exception thrown when a localization file cannot be parsed properly.

### 6.6.2 Constructor & Destructor Documentation

#### 6.6.2.1 ParseException()

```
ParseException (
    string message )
```

Constructor.

Parameters

<i>message</i>	A message that describes the error.
----------------	-------------------------------------

## 6.7 PlainString Class Reference

Represents just a string. This class is used to allow interpolated strings to preferably be passed as Formattable↵String instead of string to methods that overload both types.

### Public Member Functions

- [PlainString](#) (string value)  
*Default constructor.*

### Static Public Member Functions

- static implicit [operator PlainString](#) (string value)  
*Converts a string value to a [PlainString](#).*
- static implicit [operator PlainString](#) (FormattableString arg)  
*Converts a FormattableString value to a [PlainString](#).*

### Properties

- string [Value](#) [get]

#### 6.7.1 Detailed Description

Represents just a string. This class is used to allow interpolated strings to preferably be passed as Formattable↵String instead of string to methods that overload both types.

## 6.7.2 Constructor & Destructor Documentation

### 6.7.2.1 PlainString()

```
PlainText (
    string value )
```

Default constructor.

## 6.7.3 Member Function Documentation

### 6.7.3.1 operator PlainString() [1/2]

```
static implicit operator PlainString (
    FormattableString arg ) [static]
```

Converts a FormattableString value to a [PlainText](#).

This implicit operator is needed to avoid FormattableString values to be automatically converted to string and then to [PlainText](#) when resolving parameter overloads.

Value

Exceptions

<i>InvalidOperationException</i>	Always thrown
----------------------------------	---------------

### 6.7.3.2 operator PlainString() [2/2]

```
static implicit operator PlainString (
    string value ) [static]
```

Converts a string value to a [PlainText](#).

Parameters

<i>value</i>	Value
--------------	-------

## 6.7.4 Property Documentation

### 6.7.4.1 Value

`string Value [get]`

Value of the string.

# Index

- AutoLoadLocalizer, [17](#)
  - AutoLoadLocalizer, [19](#)
  - Context, [19](#), [20](#)
  - DEFAULT\_RESOURCE\_NAME, [24](#)
  - Load, [20](#)
  - LoadXML, [21](#), [22](#)
  - Localize, [23](#)
  - LocalizeFormat, [23](#)

- Context
  - AutoLoadLocalizer, [19](#), [20](#)
  - GlobalLocalizer, [25](#)
  - ILoadableLocalizer, [28](#), [29](#)
  - ILocalizer, [36](#)
  - Localizer, [41](#)

- DEFAULT\_RESOURCE\_NAME
  - AutoLoadLocalizer, [24](#)

- GlobalLocalizer, [24](#)
  - Context, [25](#)
  - Localize, [25](#), [26](#)
  - LocalizeFormat, [26](#)
  - Localizer, [27](#)

- I18N, [15](#)
- I18N.DotNet, [15](#)
- ILoadableLocalizer, [27](#)
  - Context, [28](#), [29](#)
  - LoadXML, [29](#), [30](#), [32](#)
  - Localize, [32](#), [34](#)
  - LocalizeFormat, [35](#)
- ILoadableLocalizer.ParseException, [45](#)
  - ParseException, [46](#)
- ILocalizer, [35](#)
  - Context, [36](#)
  - Localize, [37](#), [38](#)
  - LocalizeFormat, [38](#)

- Load
  - AutoLoadLocalizer, [20](#)

- LoadXML
  - AutoLoadLocalizer, [21](#), [22](#)
  - ILoadableLocalizer, [29](#), [30](#), [32](#)
  - Localizer, [42](#), [43](#)

- Localize
  - AutoLoadLocalizer, [23](#)
  - GlobalLocalizer, [25](#), [26](#)
  - ILoadableLocalizer, [32](#), [34](#)
  - ILocalizer, [37](#), [38](#)
  - Localizer, [44](#)

- LocalizeFormat
  - AutoLoadLocalizer, [23](#)
  - GlobalLocalizer, [26](#)
  - ILoadableLocalizer, [35](#)
  - ILocalizer, [38](#)
  - Localizer, [44](#), [45](#)
- Localizer, [39](#)
  - Context, [41](#)
  - GlobalLocalizer, [27](#)
  - LoadXML, [42](#), [43](#)
  - Localize, [44](#)
  - LocalizeFormat, [44](#), [45](#)
  - Localizer, [41](#)

- operator PlainString
  - PlainString, [47](#)

- ParseException
  - ILoadableLocalizer.ParseException, [46](#)
- PlainString, [46](#)
  - operator PlainString, [47](#)
  - PlainString, [47](#)
  - Value, [48](#)

- Value
  - PlainString, [48](#)