I18N.DotNet

main@96e92d

Generated by Doxygen 1.9.5

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-%28i18n%29 "Writing/Adapting Source Code (I18N)").
2. Write translations for internationalized strings (see writing-translations-%28l10n%29 "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.Global.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 exising 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.Global;
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.

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.

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>
```

1.3.3 Embedding the Translations File

A very convenient way of distributing the translations for an application is to embedded 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 store 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>
```

1.4 Advanced Usage

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 `Global` class are just convenience wrappers that call the global localizer.

The instance of `Global.Localizer` is created automatically on first usage (if not having been set previously) with a default `Localizer` (see Localizer Class). This default localizer has the target language set to the current UI language, and tries to load the translations file from an embedded resource identified by *Resources.I18N.xml* inside the entry (application) assembly.

The default behavior is just right for most use cases, but `Global.Localizer` can be set with a different localizer during application startup or dynamically during execution (e.g., to use a different language than the current UI language, see Specifying the Translation Target Language).

Additionally, 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), the `LoadXML` method can be invoked on the global localizer to load it.

Non-Default usage Example (C#)
```
void SetupI18N( string language, string directoryPath )
{
  Global.Localizer = new Localizer( language );
  Global.Localizer.LoadXML( directoryPath + "/I18N.xml" );
}
```

1.4.2 Local Localizers

If necessary, additional instances of the `Localizer` class 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 Specifying the Translation Target Language).

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:

- Use primary-variant code (e.g., _"en-us"_, _"es-es"_) as target language identifiers (i.e., as arguments to the `Localizer` constructor).
- 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 `Global.LocalizeFormat()` or `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 `Global.Context()` or `ILocalizer.↩
Context()` and passing it the context identifier, and then calling the localization methods on the returned context `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 is 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 ILocalizer Interface

The `ILocalizer` interface represents classes which are responsible for providing localization functionality (i.e. perform string translations) for software components.

1.4.7 Localizer Class

The `Localizer` class is a simple implementation of the `ILocalizer` interface, and is capable of loading string translations from a translations file for a single target language and then providing localization functionality.

1.4.7.1 Specifying the Translation Target Language

The default `Localizer` constructor sets the target language for translations to the current UI language (obtained from `System.Globalization.CultureInfo.CurrentUICulture`).

To specify a different target language use the `Localizer` constructor that accepts a language identifier as a parameter.

Example (C#)
```
void SetupI18N( string language )
{
  Global.Localizer = new Localizer( language );
}
```

1.4.7.2 Loading the Translations File

The translations file can be loaded into a `Localizer` by different ways:

1.4.7.2.1 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 a `Localizer` 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()
{
  Global.Localizer.LoadXML( Assembly.GetExecutingAssembly(), "I18N.Translations.xml" );
}
```

1.4.7.2.2 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 a `Localizer` instance passing the path to the file.

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

1.4.7.2.3 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 a `Localizer` instance passing a `System.IO.Stream` object which must provide the file contents.

1.5 Advanced Usage (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.

For libraries the easiest solution is to define their own "global" localizer as a static property inside a static class, similar to the `Global` 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 which automatically loads the translations file from an embedded resource (see AutoLoadLocalizer Class).

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; set; } = 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}" ) );
    }
  }
}
```

1.5.2 AutoLoadLocalizer Class

The `AutoLoadLocalizer` class is an implementation of the `ILocalizer` interface that loads automatically the translations file from an embedded resource in an assembly.

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 used as parameters to the `AutoLoadLocalizer` constructor if necessary.

Chapter 2

Namespace Index

2.1 Package List

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

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

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.

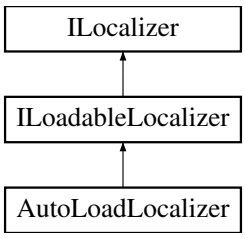
Chapter 6

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 thefrmtText 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 theargs 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=true)

  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 `null` to use the current UI language (obtained from System.Globalization.CultureInfo.CurrentUICulture)*
merge	*Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).*

Exceptions

ParseException	*Thrown when the input file cannot be parsed properly.*

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

  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 `null` to use the current UI language (obtained from System.Globalization.CultureInfo.CurrentUICulture)*
merge	*Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).*

Exceptions

ParseException	*Thrown when the stream contents cannot be parsed properly.*

- void LoadXML (XDocument doc, string? language=null, bool merge=true)

  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 `null` to use the current UI language (obtained from System.Globalization.CultureInfo.CurrentUICulture)*
merge	*Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).*

Exceptions

ParseException	*Thrown when the input document cannot be parsed properly.*

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

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* `null` *to use the current UI language (obtained from System.Globalization.CultureInfo.CurrentUICulture)*
merge	*Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).*

Exceptions

ParseException	*Thrown when the embedded resource contents cannot be parsed properly.*
InvalidOperationException	*Thrown when the embedded resource could not be found in the given assembly*

- void Load (string? language, bool merge=true)

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

Static Public Attributes

- const string DEFAULT_RESOURCE_NAME = "Resources.I18N.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 `null`)

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 = true )
```

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

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

Exceptions

ILoadableLocalizer.ParseException	Thrown when the embedded resource contents cannot be parsed properly.
InvalidOperationException	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 = true )
```

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 `null` to use the current UI language (obtained from System.Globalization.CultureInfo.CurrentUICulture)
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	Thrown when the embedded resource contents cannot be parsed properly.
InvalidOperationException	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 = true )
```

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 `null` to use the current UI language (obtained from System.Globalization.CultureInfo.CurrentUICulture)
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	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 = true )
```

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 `null` to use the current UI language (obtained from System.Globalization.CultureInfo.CurrentUICulture)
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	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 = true )
```

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 `null` to use the current UI language (obtained from System.Globalization.CultureInfo.CurrentUICulture)
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	Thrown when the input document cannot be parsed properly.

Implements ILoadableLocalizer.

6.1.3.8 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.

6.1.3.9 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 frmtText)

  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

contextId	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

frmtText	Language-neutral formattable string

Returns

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

texts	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

text	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

format	Language-neutral format string
args	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

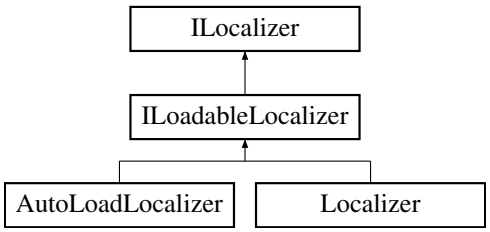
```
ILoadableLocalizer Localizer = new AutoLoadLocalizer()  [static], [get]
```

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=true)

  Loads a localization configuration from a file in XML format.
- void LoadXML (Stream stream, string? language=null, bool merge=true)

  Loads a localization configuration from a stream in XML format.
- void LoadXML (XDocument doc, string? language=null, bool merge=true)

  Loads a localization configuration from a XML document.
- void LoadXML (Assembly assembly, string resourceName, string? language=null, bool merge=true)

  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

splitContextIds	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

contextId	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 = true )
```

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 `null` to use the current UI language (obtained from System.Globalization.CultureInfo.CurrentUICulture)
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	Thrown when the embedded resource contents cannot be parsed properly.
InvalidOperationException	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 = true )
```

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 `null` to use the current UI language (obtained from System.Globalization.CultureInfo.CurrentUICulture)
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	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 = true )
```

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 `null` to use the current UI language (obtained from System.Globalization.CultureInfo.CurrentUICulture)
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	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 = true )
```

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 `null` to use the current UI language (obtained from System.Globalization.CultureInfo.CurrentUICulture)
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	Thrown when the input document cannot be parsed properly.

Implemented in AutoLoadLocalizer, and Localizer.

6.3.2.7 Localize() [1/3]

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

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.

Parameters

frmtText	Language-neutral formattable string

Returns

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

texts	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

text	Language-neutral string

Returns

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

Implemented in AutoLoadLocalizer, and Localizer.

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

format	Language-neutral format string
args	Arguments for the format string

Returns

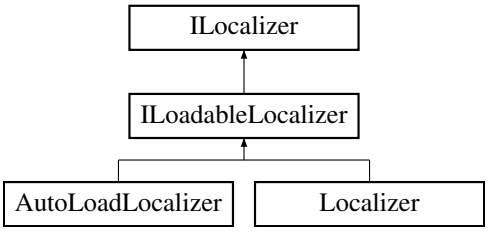
Formatted string generated from the language-specific localized format string if found, or generated from*format* otherwise

Implemented in AutoLoadLocalizer.

6.4 ILocalizer Interface Reference

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

Inheritance diagram for ILocalizer:



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

splitContextIds	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]

```
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.

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

contextId	Identifier of the context

Returns

Localizer for the given context

Implemented in AutoLoadLocalizer, and Localizer.

6.4.2.3 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.

Parameters

frmtText	Language-neutral formattable string

Returns

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

Implemented in AutoLoadLocalizer, and Localizer.

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

texts	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

text	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

format	Language-neutral format string
args	Arguments for the format string

Returns

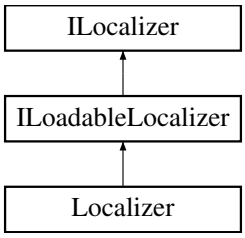
Formatted string generated from the language-specific localized format string if found, or generated from *format* otherwise

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 thefrmtText 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=true)

  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* `null` *to use the current UI language (obtained from System.Globalization.CultureInfo.CurrentUICulture)*
merge	*Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).*

Exceptions

ParseException	*Thrown when the input file cannot be parsed properly.*

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

  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* `null` *to use the current UI language (obtained from System.Globalization.CultureInfo.CurrentUICulture)*
merge	*Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).*

Exceptions

ParseException	*Thrown when the stream contents cannot be parsed properly.*

- void LoadXML (XDocument doc, string? language=null, bool merge=true)

  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* `null` *to use the current UI language (obtained from System.Globalization.CultureInfo.CurrentUICulture)*
merge	*Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).*

Exceptions

ParseException	*Thrown when the input document cannot be parsed properly.*

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

  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* `null` *to use the current UI language (obtained from System.Globalization.CultureInfo.CurrentUICulture)*
merge	*Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).*

Exceptions

ParseException	*Thrown when the embedded resource contents cannot be parsed properly.*
InvalidOperationException	*Thrown when the embedded resource could not be found in the given assembly*

- 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 languange 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 = true )
```

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 `null` to use the current UI language (obtained from System.Globalization.CultureInfo.CurrentUICulture)
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	Thrown when the embedded resource contents cannot be parsed properly.
InvalidOperationException	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 = true )
```

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 `null` to use the current UI language (obtained from System.Globalization.CultureInfo.CurrentUICulture)
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	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 = true )
```

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 `null` to use the current UI language (obtained from System.Globalization.CultureInfo.CurrentUICulture)
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	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 = true )
```

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 `null` to use the current UI language (obtained from System.Globalization.CultureInfo.CurrentUICulture)
merge	Replaces the current translations with the loaded ones when<c>false, otherwise merges both (existing translations are overridden with loaded ones).

Exceptions

ParseException	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

format	Language-neutral format string
args	Arguments for the format string

Returns

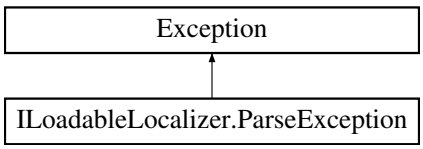
Formatted string generated from the language-specific localized format string if found, or generated from*format* otherwise

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

message	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()

```
PlainString (
            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 PlainString.

This implicit operator is needed to avoid FormattableString values to be automatically converted to string and then to PlainString when resolving parameter overloads.

Value

Exceptions

InvalidOperationException	Always thrown

6.7.3.2 operator PlainString() [2/2]

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

Converts a string value to a PlainString.

Parameters

value	Value

6.7.4 Property Documentation

6.7.4.1 Value

```
string Value  [get]
```

Value of the string.

Index