I18N.DotNet

main@be0cbf

Generated by Doxygen 1.9.5

1 I18N.DotNet 1
1.1 About . . . 1
1.2 Installation . . . 1
1.3 Getting Started . . . 1
1.3.1 Adapting Source Code (I18N) . . . 1
1.3.2 Writing Translations (L10N) . . . 2
1.4 Advanced Usage . . . 2
1.4.1 Language Identifiers & Variants . . . 2
1.4.2 String Format . . . 3
1.4.3 Global and Local Localizers . . . 3
1.4.4 Contexts . . . 3
1.4.5 Embedding the Translations File . . . 4

2 Namespace Index 5
2.1 Package List . . . 5

3 Hierarchical Index 7
3.1 Class Hierarchy . . . 7

4 Class Index 9
4.1 Class List . . . 9

5 Namespace Documentation 11
5.1 I18N Namespace Reference . . . 11
5.2 I18N.DotNet Namespace Reference . . . 11

6 Class Documentation 13
6.1 Global Class Reference . . . 13
6.1.1 Detailed Description . . . 13
6.1.2 Member Function Documentation . . . 13
6.1.2.1 Context() . . . 13
6.1.2.2 Localize() [1/3] . . . 14
6.1.2.3 Localize() [2/3] . . . 14
6.1.2.4 Localize() [3/3] . . . 15
6.1.2.5 LocalizeFormat() . . . 15
6.1.3 Property Documentation . . . 16
6.1.3.1 Localizer . . . 16
6.2 ILocalizer Interface Reference . . . 16
6.2.1 Detailed Description . . . 16
6.2.2 Member Function Documentation . . . 16
6.2.2.1 Localize() [1/3] . . . 16
6.2.2.2 Localize() [2/3] . . . 17
6.2.2.3 Localize() [3/3] . . . 17
6.2.2.4 LocalizeFormat() . . . 18

6.3 Localizer Class Reference . . . 18
6.3.1 Detailed Description . . . 19
6.3.2 Constructor & Destructor Documentation . . . 19
6.3.2.1 Localizer() . . . 19
6.3.3 Member Function Documentation . . . 20
6.3.3.1 Context() [1/2] . . . 20
6.3.3.2 Context() [2/2] . . . 20
6.3.3.3 LoadXML() [1/3] . . . 21
6.3.3.4 LoadXML() [2/3] . . . 21
6.3.3.5 LoadXML() [3/3] . . . 21
6.3.3.6 Localize() [1/3] . . . 22
6.3.3.7 Localize() [2/3] . . . 22
6.3.3.8 Localize() [3/3] . . . 23
6.3.3.9 LocalizeFormat() . . . 23
6.3.3.10 SetTargetLanguage() . . . 23
6.4 Localizer.ParseException Class Reference . . . 24
6.4.1 Detailed Description . . . 24
6.4.2 Constructor & Destructor Documentation . . . 24
6.4.2.1 ParseException() . . . 24
6.5 PlainString Class Reference . . . 24
6.5.1 Detailed Description . . . 25
6.5.2 Constructor & Destructor Documentation . . . 25
6.5.2.1 PlainString() . . . 25
6.5.3 Member Function Documentation . . . 25
6.5.3.1 operator PlainString() [1/2] . . . 25
6.5.3.2 operator PlainString() [2/2] . . . 26
6.5.4 Property Documentation . . . 26
6.5.4.1 Value . . . 26

Index **27**

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.

A 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

1.3.1 Adapting Source Code (I18N)

Source code must be adapted following two simple steps:

- The first step consists in adding a couple of calls during initialization of the program (before any translated string is used):
  - Call `I18N.DotNet.Global.Localizer.SetTargetLanguage()` to set the language to which strings will be translated.
  - Call `I18N.DotNet.Global.Localizer.LoadXML()` to load the file that contains the translations.
- The second step consists in adapting the source code in order to wrap the strings to be translated with a call to `I18N.DotNet.Global.Localize()`.

Example
```
using static I18N.DotNet.Global;
using System;
using System.IO;
using System.Reflection;
public class Program
{
  static void Main( string[] args )
  {
    var programPath = Path.GetDirectoryName( Assembly.GetExecutingAssembly().Location );
    int i = 0x555;
    Localizer.SetTargetLanguage( CultureInfo.CurrentUICulture.Name ).LoadXML( programPath + "/I18N.xml" );
    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 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>
```

1.4 Advanced Usage

1.4.1 Language Identifiers & Variants

Any arbitrary string can be used for identifying languages, and they are processed as case-insensitive.

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, if `"en-gb"` is passed to `Localizer.SetTargetLanguage()`, then 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 `Localizer.SetTargetLanguage()`).
- 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.

1.4.2 String Format

Calls to `String.Format()` where the format string has to be internationalized can be replaced by a call to `I18N.DotNet.Global.LocalizeFormat()` (or `Localizer.LocalizeFormat()`, see Global and Local Localizers).

Example
```
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 );
```

1.4.3 Global and Local Localizers

Instances of the `Localizer` class are responsible for loading string translations and then providing localization functionality (i.e. perform string translations) for software components.

The static class `I18N.DotNet.Global` 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 `I18N.DotNet.Global` class are just convenience wrappers that call the global localizer.

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. Nevertheless, for most cases using the global localizer is just enough.

1.4.4 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.

1.4.4.0.1 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 `I18N.DotNet.Global.Context()` (or `Localizer.Context()`, see Global and Local Localizers) and passing it the context identifier, and then calling the localization methods on the returned context `Localizer`.

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

1.4.4.0.2 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="Button">
      <Entry>
        <Key>OK</Key>
        <Value lang="es">Correcto</Value>
      </Entry>
    </Context>
  </Context>
</I18N>
```

1.4.5 Embedding the Translations File

Instead of using translation files installed on the filesystem during the installation procedure for the application, these files can be embedded inside an executable assembly. Embedded resource files can then be accessed as `Stream` objects which are passed to `Localizer.LoadXML`.

Chapter 2

Namespace Index

2.1 Package List

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

I18N 11
I18N.DotNet 11

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

ApplicationException
    Localizer.ParseException 24
Global 13
ILocalizer 16
    Localizer 18
PlainString 24

Chapter 4

Class Index

4.1 Class List

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

Global
: Utility class for convenient access to localization functions. 13

ILocalizer
: Converter of strings from a language-neutral value to its corresponding language-specific localization. 16

Localizer
: Converter of strings from a language-neutral value to its corresponding language-specific localization. 18

Localizer.ParseException
: Exception thrown when a localization file cannot be parsed properly. 24

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. 24

Chapter 5

Namespace Documentation

5.1 I18N Namespace Reference

Namespaces

- namespace DotNet

5.2 I18N.DotNet Namespace Reference

Classes

- class Global

  Utility class for convenient access to localization functions.
- 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 Global 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 Localizer Context (string contextId)

  Gets a context in the global localizer.

Properties

- static Localizer Localizer = new Localizer() `[get]`

6.1.1 Detailed Description

Utility class for convenient access to localization functions.

6.1.2 Member Function Documentation

6.1.2.1 Context()

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

Gets a context in the global localizer.

See also

Localizer.Context(string)

Parameters

contextId	Identifier of the context

Returns

Localizer for the given context

6.1.2.2 Localize() [1/3]

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

Localizes an interpolated string using the global localizer.

See also

Localizer.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

6.1.2.3 Localize() [2/3]

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

Localizes multiple strings.

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

Parameters

texts	Array of language-neutral strings

Returns

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

6.1.2.4 Localize() [3/3]

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

Localizes a string using the global localizer.

See also

Localizer.Localize(PlainString)

Parameters

text	Language-neutral string

Returns

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

6.1.2.5 LocalizeFormat()

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

Localizes and then formats a string using the global localizer.

See also

Localizer.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

6.1.3 Property Documentation

6.1.3.1 Localizer

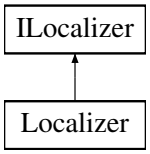
```
Localizer Localizer = new Localizer()  [static], [get]
```

Global localizer.

6.2 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.

6.2.1 Detailed Description

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

6.2.2 Member Function Documentation

6.2.2.1 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 Localizer.

6.2.2.2 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 Localizer.

6.2.2.3 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 Localizer.

6.2.2.4 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

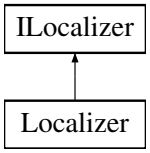
Formatted string generated from the language-specific localized format string if found, or generated from*format* otherwise

Implemented in Localizer.

6.3 Localizer Class Reference

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

Inheritance diagram for Localizer:



Classes

- class ParseException

  Exception thrown when a localization file cannot be parsed properly.

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)

  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.
- 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.
- Localizer Context (IEnumerator< string > splitContextIds)

  Gets the localizer for a context in the current localizer.
- Localizer SetTargetLanguage (string language)

  Sets the localized language to which conversion will be performed.
- void LoadXML (string filepath, bool merge=true)

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

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

  Loads a localization configuration from a XML document.

6.3.1 Detailed Description

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

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Localizer()

```
Localizer ( )
```

Default constructor.

6.3.3 Member Function Documentation

6.3.3.1 Context() [1/2]

```
Localizer Context (
            IEnumerator< 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

6.3.3.2 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.

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

6.3.3.3 LoadXML() [1/3]

```
void LoadXML (
            Stream stream,
            bool merge = true )
```

Loads a localization configuration from a stream in XML format.

Precondition: The language must be set before calling this method.

Parameters

stream	Stream with the localization configuration in XML format
merge	Replaces the current localization mapping with the loaded one when<c>false, otherwise merges both (existing mappings are overridden with loaded ones).

Exceptions

InvalidOperationException	Thrown when the language is not set.
ParseException	Thrown when the input file cannot be parsed properly.

6.3.3.4 LoadXML() [2/3]

```
void LoadXML (
            string filepath,
            bool merge = true )
```

Loads a localization configuration from a file in XML format.

Precondition: The language must be set before calling this method.

Parameters

filepath	Path to the localization configuration file in XML format
merge	Replaces the current localization mapping with the loaded one when<c>false, otherwise merges both (existing mappings are overridden with loaded ones).

Exceptions

InvalidOperationException	Thrown when the language is not set.
ParseException	Thrown when the input file cannot be parsed properly.

6.3.3.5 LoadXML() [3/3]

```
void LoadXML (
```

```
            XDocument doc,
            bool merge = true )
```

Loads a localization configuration from a XML document.

Precondition: The language must be set before calling this method.

Parameters

doc	XML document with the localization configuration
merge	Replaces the current localization mapping with the loaded one when<c>false, otherwise merges both (existing mappings are overridden with loaded ones).

Exceptions

InvalidOperationException	Thrown when the language is not set.
ParseException	Thrown when the input file cannot be parsed properly.

6.3.3.6 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.

6.3.3.7 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.

6.3.3.8 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.

6.3.3.9 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.

6.3.3.10 SetTargetLanguage()

```
Localizer SetTargetLanguage (
            string language )
```

Sets the localized language to which conversion will be performed.

Language matching is case-insensitive.

Any arbitrary string can be used for identifying languages, but when using language identifiers formed by a primary code and a variant code separated by an hyphen(e.g., "en-us") if a localized conversion for the "full" language is not found then a conversion for the primary(base) language is tried too.

Parameters

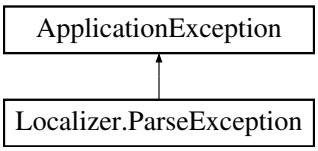
language	Name, code or identifier for the language

Returns

6.4 Localizer.ParseException Class Reference

Exception thrown when a localization file cannot be parsed properly.

Inheritance diagram for Localizer.ParseException:



Public Member Functions

- ParseException (string message)
  Constructor.

6.4.1 Detailed Description

Exception thrown when a localization file cannot be parsed properly.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 ParseException()

```
ParseException (
            string message )
```

Constructor.

Parameters

message	A message that describes the error.

6.5 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]`

6.5.1 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.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 PlainString()

```
PlainString (
            string value )
```

Default constructor.

6.5.3 Member Function Documentation

6.5.3.1 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

6.5.3.2 operator PlainString() [2/2]

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

Converts a string value to a PlainString.

Parameters

value	Value

6.5.4 Property Documentation

6.5.4.1 Value

```
string Value  [get]
```

Value of the string.

Index

Context
    Global, 13
    Localizer, 20

Global, 13
    Context, 13
    Localize, 14, 15
    LocalizeFormat, 15
    Localizer, 16

I18N, 11
I18N.DotNet, 11
ILocalizer, 16
    Localize, 16, 17
    LocalizeFormat, 18

LoadXML
    Localizer, 20, 21
Localize
    Global, 14, 15
    ILocalizer, 16, 17
    Localizer, 22
LocalizeFormat
    Global, 15
    ILocalizer, 18
    Localizer, 23
Localizer, 18
    Context, 20
    Global, 16
    LoadXML, 20, 21
    Localize, 22
    LocalizeFormat, 23
    Localizer, 19
    SetTargetLanguage, 23
Localizer.ParseException, 24
    ParseException, 24

operator PlainString
    PlainString, 25, 26

ParseException
    Localizer.ParseException, 24
PlainString, 24
    operator PlainString, 25, 26
    PlainString, 25
    Value, 26

SetTargetLanguage
    Localizer, 23

Value
    PlainString, 26