

CAPLIN

Caplin Trader 2.2

How To Localize Your Application

April 2011

CONFIDENTIAL

Contents

1	Preface.....	1
1.1	What this document contains.....	1
	About Caplin document formats	1
1.2	Who should read this document.....	1
1.3	Related documents.....	2
1.4	Typographical conventions.....	2
1.5	Feedback.....	3
1.6	Acknowledgments.....	3
2	Assumptions and restrictions.....	4
3	Setting up a new locale.....	5
3.1	Adding the locale identifier.....	6
3.2	Partitioning localization properties across locales.....	7
4	Translating to the new locale.....	8
4.1	A. Creating properties files for the new locale.....	10
	Creating properties files for your application and blades	11
	Creating properties files for the Caplin Trader framework	13
4.2	B. Defining formatted items in the new locale.....	15
4.3	C. Editing localization properties for formatted items.....	16
	Adding number multiplier properties	19
4.4	D. Preparing a common terms translation dictionary.....	20
4.5	E. Translating common terms.....	21
4.6	F. Translating the text.....	21
	Guidelines for translators	22
	After Translation	25
4.7	G. Localizing CSS.....	25
5	Building and testing Caplin Trader for the new locale.....	26
6	Reusing a localized Caplin-supplied blade.....	27
7	Tools to help with localization.....	28
7.1	Report of Localization Keys Details by key name.....	29
7.2	Report of Missing Localization Keys by locale.....	30
7.3	Tabular Report of Missing Localization Keys by key name.....	31
7.4	Localization Key Values Report (csv format).....	32

8 Troubleshooting..... 33

 8.1 ??? <key.name> ??? displayed in text fields..... 33

 8.2 Translations missing in variant locale..... 33

9 Appendix A: Date and time format specifiers..... 34

10 Glossary of terms and acronyms..... 37

1 Preface

1.1 What this document contains

This document explains how to localize a Caplin Trader application to display text in a different language and display dates, times, numeric data, and other items, in formats according to local preferences. The combination of language and display preferences is called a **locale**.

Note: Caplin Trader applications can only be localized if they have been coded to support this; the code is then said to contain **localization support**. For details of how to do this, see the document **Caplin Trader: How To Add Localization Support**.

About Caplin document formats

This document is supplied in three formats:

- ◆ Portable document format (*.PDF* file), which you can read on-line using a suitable PDF reader such as Adobe Reader®. This version of the document is formatted as a printable manual; you can print it from the PDF reader.
- ◆ Web pages (*.HTML* files), which you can read on-line using a web browser. To read the web version of the document navigate to the *HTMLDoc_m_n* folder and open the file *index.html*.
- ◆ Microsoft HTML Help (*.CHM* file), which is an HTML format contained in a single file. To read a *.CHM* file just open it – no web browser is needed.

For the best reading experience

On the machine where your browser or PDF reader runs, install the following Microsoft Windows® fonts: Arial, Courier New, Times New Roman, Tahoma. You must have a suitable Microsoft license to use these fonts.

Restrictions on viewing *.CHM* files

You can only read *.CHM* files from Microsoft Windows.

Microsoft Windows security restrictions may prevent you from viewing the content of *.CHM* files that are located on network drives. To fix this either copy the file to a local hard drive on your PC (for example the Desktop), or ask your System Administrator to grant access to the file across the network. For more information see the Microsoft knowledge base article at <http://support.microsoft.com/kb/896054/>.

1.2 Who should read this document

This document is intended for Technical Managers and Software Developers who need to localize a Caplin Trader application. There is also a section giving [guidelines for translators](#)²².

1.3 Related documents

◆ Caplin Trader: Localization Overview And Concepts

Gives an overview of localization within Caplin Trader. It covers localization concepts, how support for localization is built in to Caplin Trader, and how Caplin Trader applications are localized.

◆ Caplin Trader: How To Add Localization Support

Explains how to code a Caplin Trader application so that it can subsequently be localized to other languages and regional preferences.

◆ Caplin Trader API Reference

Documents the JavaScript libraries that allow developers to implement Caplin Trader applications by writing custom JavaScript code.

1.4 Typographical conventions

The following typographical conventions are used to identify particular elements within the text.

<i>Type</i>	<i>Uses</i>
aMethod	Function or method name
<i>aParameter</i>	Parameter or variable name
<i>/AFolder/Afile.txt</i>	File names, folders and directories
<div>Some code;</div>	Program output and code examples
The value=10 attribute is...	Code fragment in line with normal text
Some text in a dialog box	Dialog box output
Something typed in	User input – things you type at the computer keyboard
Glossary term	Items that appear in the “Glossary of terms and acronyms”
XYZ Product Overview	Document name
◆	Information bullet point
■	Action bullet point – an action you should perform

Note: Important Notes are enclosed within a box like this.
Please pay particular attention to these points to ensure proper configuration and operation of the solution.

Tip: Useful information is enclosed within a box like this.
Use these points to find out where to get more help on a topic.

Information about the applicability of a section is enclosed in a box like this.
For example: “This section only applies to version 1.3 of the product.”

1.5 Feedback

Customer feedback can only improve the quality of our product documentation, and we would welcome any comments, criticisms or suggestions you may have regarding this document.

Visit our feedback web page at <https://support.caplin.com/documentfeedback/>.

1.6 Acknowledgments

Adobe® Reader is a registered trademark of Adobe Systems Incorporated in the United States and/or other countries.

Windows and *Excel* are registered trademarks of Microsoft Corporation in the United States and other countries.

Ext JS is a Cross-Browser, Rich Internet Application Framework produced by Sencha Inc.

2 Assumptions and restrictions

Names used in examples

This guide refers to Windows folder trees where the files used for localization are located within Caplin Trader, blades, and application code. Some of the folder names in the tree are specific to your organization, so for simplicity, the examples in the guide assume the following settings for these names:

Organization specific folder name	Used for	Name used in examples
<org-name>	The name of your organization.	<i>novobank</i>
<version>	The version of your Caplin Trader application.	<i>1.0</i>
<app-name>	The name of your Caplin Trader application.	<i>novotrader</i>
<JavaScript_namespace>	The namespace for the JavaScript code of the application.	<i>novox</i>

Code repository and code location

This guide assumes that you keep your Caplin Trader software and your Caplin Trader application code under software configuration management (SCM), and the configuration management system (“the code repository”) maps the software files to a tree of folders on a Windows PC, from where they can be accessed.

In the rest of the guide, the Windows folder tree where the Caplin Trader framework and application code are located is referred to for convenience as **<app-root>**

<app-root> is defined as

X:\<scm-root>\<org-name>\<version>

where:

- ◆ *X:* is a suitable Windows drive mapping.
- ◆ *<scm-root>* is the root folder to which the configuration management system maps files (*scm* means “software configuration management”).
- ◆ *<org-name>* and *<version>* are as defined above.

For example:

<app-root> = *X:\scm_root\novobank\1.0*

Localization properties files must be UTF-8 encoded

Localization involves creating and editing a set of **localization properties files** (see **Caplin Trader: Localization Overview And Concepts**). These files must be in the **UTF-8** character set *without* a **Byte-Order Mark** at the start of the file.

Note: When editing localization properties files, make sure that the text editor supports UTF-8, and has been configured to use the UTF-8 character set and to save the files with no Byte-Order Mark.

3 Setting up a new locale

In Caplin Trader, each locale is defined by a **locale identifier**, which allows the application to select the correct locale for each end-user. You must first decide on the locale identifier(s) for the locale that you want to set up.

Locale identifiers are of the form:

`<language-code>_<country>`

- ◆ `<language-code>` follows the [ISO 639.1](#) standard for 2 letter language codes.
- ◆ `<country>` follows the [ISO 3166](#) standard for 2 letter country codes.

Examples

- ◆ The default locale for the shipped Caplin Trader software is American English, for which the locale identifier is **en_US**.
- ◆ Caplin Trader also supports British English, for which the identifier is **en_GB**.
- ◆ If your application is to be localized to support standard French, the locale identifier for this would be **fr_FR**.
- ◆ Alternatively, if the localization is to support Canadian French, the locale identifier would be **fr_CA**.
- If the locale is already supported in Caplin Trader, use the locale identifier that is already defined for it.
- If the locale is not already supported in Caplin Trader, define a locale identifier for it, following the standards defined above.

Tip: There is a list of the locales supported in the Caplin Trader product, and their corresponding locale identifiers, in the document **Caplin Trader: Localization Overview And Concepts**.

Base locale and variants

You may wish to localize the application to support a **base locale** and one or more **variant locales**. For example, a French localization could support the following:

Localization to:	Locale identifier
Standard French (base locale)	fr_FR
Canadian French (variant locale of Standard French)	fr_CA
Belgian French (variant locale of Standard French)	fr_BE
Swiss French (variant locale of Standard French)	fr_CH

Note: The base locale for English is **en_US** (American English)

3.1 Adding the locale identifier

- Add your new locale identifier(s) to the configurable list of supported locales.

This list controls the locales that are available for the end-user to choose from on the application's login screen. It is located in:

`<app-root>\novotrader\webapp\conf\applicationProperties.xml`

Edit the property `CAPLIN.LOCALE.LIST`:

```
<property name="CAPLIN.LOCALE.LIST" value="en_US,en_GB,fr_FR" />
```

In this example the locale `fr_FR` (standard French) has been added to the list.

Note: The locale identifiers in the locale configuration file *must* have a country code. For example, if standard French is supported through the base locale properties files *fr.properties*, the locale identifier in the configuration file must be `fr_FR`, *not* `fr`.

- Update the list of language names.

This list defines the name of each language *in the language itself*, and is used to present the available languages to end-users when they log in to the application. It is located in the file

`<app-root>\novotrader\webapp\modules\i18n\caplin\i18n\en\en.properties`

Add a new property of the form:

`ct.i18n.language.name.<new-locale>=<language-name>`

For example:

`ct.i18n.language.name.fr_FR=Français (France)`

`ct.i18n.language.name.fr_CA=Français (Canada)`

3.2 Partitioning localization properties across locales

The localized items for a particular locale (translated text, and display preferences such as formats for dates, times, and numbers), are defined in **localization properties**. These properties are defined in a set of **localization properties files**. If you are implementing a base locale and variants, the localization properties files must be named as shown in the following example for French:

Localization to:	Locale identifier:	Localization properties files are called:
Standard French (base locale)	fr_FR	fr.properties (Note there is no country code in the file name, because this is a base locale.)
Canadian French (variant locale)	fr_CA	fr_CA.properties
Belgian French (variant locale)	fr_BE	fr_BE.properties
Swiss French (variant locale)	fr_CH	fr_CH.properties

Each **variant localization properties file** should only contain localization properties that differ from the base locale's equivalent properties. For example, an *en.properties* file (US English) might contain the following properties:

Example *en.properties*

```
novobank.novotrader.app.season.winter=winter
novobank.novotrader.app.season.spring=spring
novobank.novotrader.app.season.summer=summer
novobank.novotrader.app.season.fall=fall
```

The equivalent *en_GB.properties* file (British English variant locale) would contain the localization property that differs, but not the other three properties:

Example *en_GB.properties*

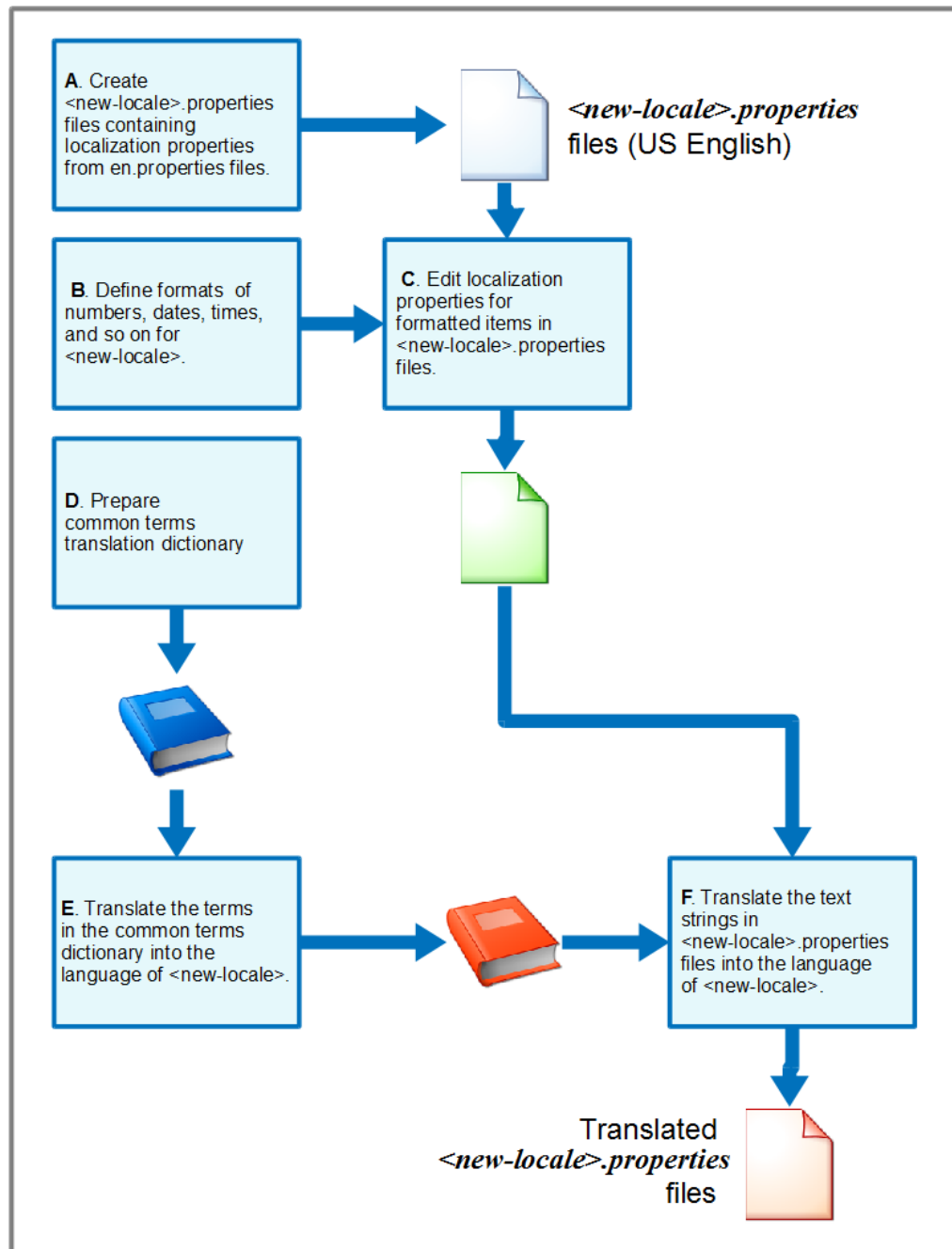
```
novobank.novotrader.app.season.fall=autumn
```

At run time, when the application has the locale *en_GB*, the text for the season is obtained from the definitions in *en.properties*, except for the **localization key** *novobank.novotrader.app.season.fall*, where the text is obtained from the variant definition in *en_GB.properties*.

Tip: For more information about localization properties, localization keys, and localization files, see the document **Caplin Trader: Localization Overview And Concepts**.

4 Translating to the new locale

Having decided on the locale identifier for a new locale, translate the localization properties defined for the `en` (US English) locale into the new locale. The following diagram summarizes this process:



The translation process

In practice, there may need to be several iterations of translation to produce text of the required quality for the new locale. The translation dictionary (steps D and E) can be produced in parallel with the other translation preparation steps (steps A – C). There is a further step G, not shown on the diagram, which involves localizing CSS files; this step can also be performed in parallel.

The following sections explain the steps of this process in more detail.

Note: If the new locale is a variant locale, and the base locale is not supported in Caplin Trader, you should translate the Caplin Trader framework to the base locale (along with your application and any blades it uses) before translating to the variant locale.

- [A. Creating properties files for the new locale](#) ¹⁰
- [B. Defining formatted items in the new locale](#) ¹⁵
- [C. Editing localization properties for formatted items](#) ¹⁶
- [D. Preparing a common terms translation dictionary](#) ²⁰
- [E. Translating common terms](#) ²¹
- [F. Translating the text](#) ²¹
- [G. Localizing CSS](#) ²⁵

4.1 A. Creating properties files for the new locale

You will need to create one or more **localization properties files** for the new locale, containing the properties for the strings that are to be translated. You do this by copying the contents of existing properties files. When the new locale is a base locale (for example, standard French, `fr_FR`), you copy the properties for the base English locale (`en_US`). When the new locale is a variant (for example, Canadian French, `fr_CA`), you copy the properties for the base locale of the language (for example, the properties for `fr_FR`). The property values in the files for the new locale are then translated.

- Assuming that the translators will directly edit the localization properties files, it is recommended that you supply them with a Zip file containing the files in a copy of the code repository's folder structure. The translated files should be returned to you in a Zip file with the same folder structure. You can then unzip the file into the code repository, and the properties files for the new locale will appear in the correct locations.
- If the translators cannot be provided with a Zip file, make sure you record against each of the `<new-locale>.properties` files its location in the repository, so that translated versions can subsequently be placed back in the same location.

Note: The content of the `<new-locale>.properties` files must be in the **UTF-8** character set *without* a **Byte-Order Mark** at the start of the file (see [Assumptions and Restrictions](#) ^[4]).

Note: Where the new locale is a variant of an existing base locale, Caplin strongly recommend that the localization properties files for the new locale only contain properties for which the translation or format differs from the equivalents in the base locale. This makes it easier to maintain the localizations.
See [Partitioning localization properties across locales](#) ^[7].

Tip: For a list of the locales that are supported in Caplin Trader, refer to the document **Caplin Trader: Localization Overview And Concepts**.

Detailed instructions for creating new properties files

- If the new locale is already supported within Caplin Trader:
 - Follow the steps in [Creating properties files for your application and blades](#) ^[11].
 - Then go to section [B. Defining formatted items in the new locale](#) ^[15].
- If the new locale is *not* already supported within Caplin Trader:
 - Follow the steps in [Creating properties files for your application and blades](#) ^[11].
 - Then follow the steps in [Creating properties files for the Caplin Trader framework](#) ^[13].
 - Then go to section [B. Defining formatted items in the new locale](#) ^[15].

Creating properties files for your application and blades

You will need to support the new locale in:

- ◆ your Caplin Trader application,
- ◆ any blades that your organization has designed and implemented,
- ◆ any Caplin-supplied blades used by your application that do not already support the new locale.

The following steps explain how to create the localization properties files whose contents will later be translated.

Where a step refers to files called *<base_locale>.properties*, the *<base_locale>* part of the name is the locale identifier of the base locale, without the country code. For example, if the base locale's identifier is *fr_FR* (Standard French), the file is *fr.properties*.

1. Create properties files for the application code

Your application should have a localization properties file for the base English locale. This is typically in:

```
<app-root>\novotrader\novox\i18n\en\en.properties
```

Base locales

- If the new locale is a base locale, copy *<app-root>\novotrader\novox\i18n\en\en.properties* to

```
<app-root>\novotrader\novox\i18n\<base-locale>\<base-locale>.properties
```

For example, if the base locale is *fr_FR* (Standard French), copy the *en.properties* file to

```
<app-root>\novotrader\novox\i18n\fr\fr.properties
```

Variant locales

- If the new locale is a variant locale, copy

```
<app-root>\novotrader\novox\i18n\<base-locale>\<base-locale>.properties
```

to

```
<app-root>\novotrader\novox\i18n\<base-locale>\<variant-locale>.properties
```

For example, if the base locale is *fr_FR* (Standard French), and the variant locale is *fr_CA* (Canadian French), copy

```
<app-root>\novotrader\novox\i18n\fr\fr.properties
```

to

```
<app-root>\novotrader\novox\i18n\fr\fr_CA.properties
```

2. Create properties files for the blades

Locate the *en* localization properties files for any blades that the application uses. These files are in:

```
<app-root>\blades\<blade-name>\i18n\en\en.properties
```

For example:

```
<app-root>\blades\novo-blade-A\i18n\en\en.properties
```

```
<app-root>\blades\novo-blade-B\i18n\en\en.properties
```

Base locales

- If the new locale is a base locale, for each blade, copy
`<app-root>\blades\<blade-name>\i18n\en\en.properties` to
`<app-root>\blades\<blade-name>\i18n\<base-locale>\<base-locale>.properties`
For example, if the blade is `novo-blade-A` and the base locale is `fr_FR` (Standard French), copy
`<app-root>\blades\novo-blade-A\i18n\en\en.properties`
to
`<app-root>\blades\novo-blade-A\i18n\fr\fr.properties`

Variant locales

- If the new locale is a variant locale, for each blade, copy its `<base-locale>.properties` file to
`<variant-locale>.properties`
For example, if the blade is `novo-blade-A`, the base locale is `fr_FR` (Standard French), and the variant locale is `fr_CA` (Canadian French), copy
`<app-root>\blades\novo-blade-A\i18n\fr\fr.properties`
to
`<app-root>\blades\novo-blade-A\i18n\fr\fr_CA.properties`

Tip: Make sure you do this for the `blades-common` properties in
`<app-root>\blades\blades-common\i18n\en\en.properties`

3. Next steps

- If Caplin Trader does not support the new locale, follow the steps in [Creating properties files for the Caplin Trader framework](#)^[13].
- If Caplin Trader does support the new locale, go to [B. Defining formatted items in the new locale](#)^[15].

Creating properties files for the Caplin Trader framework

If the Caplin Trader framework supports the new locale, ignore this section.

If the Caplin Trader framework *does not* support the new locale, you need to localize the framework code yourself. The following steps explain how to create the localization properties files whose contents will be translated.

Where a step refers to files called `<base_locale>.properties`, the `<base_locale>` part of the name is the locale identifier of the base locale, without the country code. For example, if the base locale's identifier is `fr_FR` (Standard French), the file is `fr.properties`.

Caplin Trader framework properties files

- Locate the English base localization properties files for the Caplin Trader framework.

You can obtain these files from the built version of your application in:

```
<app-root>\novotrader\webapp\modules\i18n\caplin\
```

There is a properties file for each Caplin Trader module, in a sub-folder:

```
<module-name>\i18n\en\en.properties
```

For example:

```
core\i18n\en\en.properties
```

```
grid\i18n\en\en.properties
```

For each Caplin Trader module:

Base Locales

- If the new locale is a base locale, append the contents of the module's `en.properties` file in

```
<app-root>\novotrader\webapp\modules\i18n\caplin\<module-name>\i18n\en\en.properties
```

to

```
<app-root>\novotrader\novox\i18n\<base-locale>\<base-locale>.properties
```

(the application's localization properties file for the new base locale).

For example, if the Caplin Trader module name is `core` and the new base locale is `fr_FR` (Standard French), append the contents of

```
<app-root>\novotrader\webapp\modules\i18n\caplin\core\i18n\en\en.properties
```

to

```
<app-root>\novotrader\novox\i18n\fr\fr.properties
```

Variant locales

- If the new locale is a variant locale, and the base locale is already supported in Caplin Trader, append the contents of the module's `<base-locale>.properties` file in

```
<app-root>\novotrader\webapp\modules\i18n\caplin\<module-name>\i18n\
```

```
<base-locale>\<base-locale>.properties
```

to

```
<app-root>\novotrader\novox\i18n\<base-locale>\<variant-locale>.properties
```

(the application's localization properties file for the variant locale).

For example, if the Caplin Trader module name is `core`, the base locale is `fr_FR` (Standard French), and the variant locale is `fr_CA` (Canadian French), append the contents of

```
<app-root>\novotrader\webapp\modules\i18n\caplin\core\i18n\fr\fr.properties
```

to

```
<app-root>\novotrader\novox\i18n\fr\fr_CA.properties
```

- If the new locale is a variant locale, and the base locale is *not* supported in Caplin Trader, then assuming that you have already translated to the base locale, you do not need to copy any files.

This is because the relevant Caplin Trader module localization properties for the base locale in

```
<app-root>\novotrader\novox\i18n\<base-locale>\<base-locale>.properties
```

(the application's localization properties file for the base locale).

were copied to the variant locale's properties file

```
<app-root>\novotrader\novox\i18n\<base-locale>\<variant-locale>.properties
```

in step 1 of [Creating properties files for your application and blades](#) ¹¹ ("If the new locale is a variant locale, copy...").

4.2 B. Defining formatted items in the new locale

The formats of some types of displayed items are determined by the application's locale. For Caplin Trader applications you can define the format of:

- ◆ Dates.
 - ◆ Times.
 - ◆ The digit separator for numbers above 999.
 - ◆ The **decimal radix character**.
 - ◆ The amount suffixes displayed against numbers.
- Decide on the formats of these items for the new locale.

For example:

Formats for the British English locale `en_GB`

Format type	Required format
Dates	31/12/2011
Times	10:33 pm
Digit separator	123,456,789
Decimal radix character	123.45678
Amount suffix for thousands	10 K

Formats for the French locale `fr_FR`

Format type	Required format
Dates	31-12-2011
Times	22:33
Digit separator	123 456 789
Decimal radix character	123,45678
Amount suffix for thousands	10 K

Formats for the German locale `de_DE`

Format type	Required format
Dates	31.12.2011
Times	22:33
Digit separator	123.456.789
Decimal radix character	123,45678
Amount short suffix for thousands	10 K

- Now edit the localization properties to define your chosen formats – see [C. Editing localization properties for formatted items](#) ¹⁶.

4.3 C. Editing localization properties for formatted items

The following steps explain how to edit the localization properties to define your chosen formats.

Where a step refers to files called `<base_locale>.properties`, the `<base_locale>` part of the name is the locale identifier of the base locale, without the country code. For example, if the base locale's identifier is `fr_FR` (Standard French), the file is `fr.properties`.

The formats for dates, times, and numbers are defined in the localization properties files in:

- ◆ `<app-root>\novotrader\webapp\modules\i18n\caplin\i18n\i18n\...`
(contains the main definitions)
- ◆ `<app-root>\novotrader\webapp\modules\i18n\caplin\element\i18n\...`
(contains the amount suffixes displayed against numbers)

The formats for the base English locale (`en_US`) are in `en\en.properties` files.

For example:

```
<app-root>\novotrader\webapp\modules\i18n\caplin\i18n\i18n\en\en.properties
```

Base locales

- If the new locale is a base locale:
 - Append all the properties in the `caplin\i18n\i18n\en\en.properties` file that start with `ct.i18n.date.`, `ct.i18n.time.`, `ct.i18n.number.`, and `ct.i18n.decimal.` to

`<app-root>\novotrader\novox\i18n\<base-locale>\<base-locale>.properties`
(the application's localization properties file for the new base locale).

For example, if the base locale is `fr_FR` (Standard French), append the relevant properties to

`<app-root>\novotrader\novox\i18n\fr\fr.properties`
 - Append all the properties in the `caplin\element\i18n\en\en.properties` file to the same file as above.

Variant locales

- If the new locale is a variant locale, and the base locale is already supported within Caplin Trader:
 - Append all the properties in the `caplin\i18n\i18n\<base-locale>\<base-locale>.properties` file that start with `ct.i18n.date.`, `ct.i18n.time.`, `ct.i18n.number.`, and `ct.i18n.decimal.` to

`<app-root>\novotrader\novox\i18n\<base-locale>\<new-locale>.properties`
(the application's localization properties file for the variant locale).

For example, if the base locale is `fr_FR` (Standard French), and the variant locale is `fr_CA` (Canadian French), append the relevant properties in

`<app-root>\novotrader\novox\i18n\fr\fr.properties`
to

`<app-root>\novotrader\novox\i18n\fr\fr_CA.properties`

- Append all the properties in the `caplin\element\i18n\<base-locale>\<base-locale>.properties` file to the same file as above.
- If the new locale is a variant locale, and the base locale is *not* supported in Caplin Trader, then assuming that you have already defined the formatted items for the base locale, you do not need to copy any properties.

This is because the relevant Caplin Trader localization properties for the base locale in

`<app-root>\novotrader\novox\i18n\<base-locale>\<base-locale>.properties`
(the application's localization properties file for the base locale).

were copied to the variant locale's properties file

`<app-root>\novotrader\novox\i18n\<base-locale>\<variant-locale>.properties`

in step 1 of [Creating properties files for your application and blades](#)¹¹ (“If the new locale is a variant locale, copy...”).

Editing the formatting properties

- Edit the following formatting properties in the file to which you copied them in the previous steps.
(The property values shown are those for the `en_US` locale.)

Formatting Property	Description
<code>ct.i18n.time.format.separator=:</code>	Separator between the hours, minutes and seconds parts of a time. For example: 17:23:31
<code>ct.i18n.date.format=m-d-Y</code>	Format of dates. For example: 07-31-2011 See notes 1, 2.
<code>ct.i18n.date.format.long=D, M d, Y, h:i:s A</code>	Format of dates with times (“long” dates). For example: Sun, Jul 31, 2011, 05:23:07 PM See notes 1, 2.
<code>ct.i18n.decimal.radix.character=.</code>	The decimal radix character . For example: 1234.56789
<code>ct.i18n.number.grouping.separator=,</code>	The digit separator for numbers above 999. For example: 123,456,789 See note 3.
<code>ct.element.number.formatting.amount.suffix.short.thousands=K</code>	Amount short suffix for thousands. For example: 10.4K
<code>ct.element.number.formatting.amount.suffix.short.millions=M</code>	Amount short suffix for millions. For example: 21.3M
<code>ct.element.number.formatting.amount.suffix.short.billions=B</code>	Amount short suffix for billions. For example: 4.87B
<code>ct.element.number.formatting.amount.suffix.thousands=K</code>	Amount standard suffix for thousands. For example: 10.4K

Formatting Property	Description
<code>ct.element.number.formatting.amount.suffix.millions=MIL</code>	Amount standard suffix for millions. For example: 21.3MIL
<code>ct.element.number.formatting.amount.suffix.billions=BIL</code>	Amount standard suffix for billions. For example: 4.87BIL

Notes

1. The format specifiers that can be used in the localization properties `ct.i18n.date.format` and `ct.i18n.date.format.long` are those defined for the **Date** class in the Ext JS JavaScript library. For convenience, these format specifiers are listed in [Appendix A: Date and time format specifiers](#)^[34].
The definitive reference information for the formats can be found on the Ext JS API web site at <http://dev.sencha.com/deploy/dev/docs/?class=Date>
2. The separators between parts of a date and/or time defined by `ct.i18n.date.format` and `ct.i18n.date.format.long` can be defined as any legal UTF-8 characters. For example, the time part of `ct.i18n.date.format.long` could be defined as `h:i:s` (05:23.07)
3. To set the number grouping separator to a space, use the backslash character followed by a space:
`ct.i18n.number.grouping.separator=\<space char>`
Example of number grouping when the separator is defined as a space:
123 456 789

Tip: The formatting properties copied include text strings for day names, month names, and the abbreviations for “before midday” and “after midday”. These strings should be translated along with the other localization properties to be translated (see [Translating the text](#)^[21]).

- Add any extra number multiplier properties required for the new locale.
See [Adding number multiplier properties](#)^[19].

Adding number multiplier properties

In trading applications, end-users often need to enter large quantities in a shorthand way using a multiplier suffix. For example, the value 5,000 may be entered as 5K, where the suffix K is a thousands multiplier. Caplin Trader provides a **parse()** method on the JavaScript class **caplin.element.parser.LocalisedAmountParser** that can parse a string, such as “5K” or “5k”, and produce its actual numeric value (5000) for subsequent use by the application code.

This class uses a set of localization properties that specify which number multipliers can be used in screen input for each supported locale. At the time of publication, the multipliers supported for the base English (en_US) locale are as follows:

Number multiplier suffixes for locale en_US

Localization key and English value	Description
<code>ct.element.number.formatting.multiplier.k=1000</code>	The multiplier when k or K is the multiplier suffix (for example 5k = 5 x 1000).
<code>ct.element.number.formatting.multiplier.m=1000000</code>	The multiplier when m or M is the multiplier suffix (for example 5m = 5 x 1000000).
<code>ct.element.number.formatting.multiplier.b=1000000000</code>	The multiplier when b or B is the multiplier suffix (for example 5b = 5 x 1000000000).

These properties are defined in

`<app-root>\<my-app-name>\webapp\modules\i18n\caplin\element\i18n\en\en.properties`

If the new locale requires additional multiplier suffixes, you can add new properties to define them.

- Add the properties to the `<new-locale>.properties` file.
- Each new property has a name of the form:

`ct.element.number.formatting.multiplier.<multiplier-suffix>`

As a hypothetical example, assume the new locale is Malay (ms_MY). The Malay word for billion is “juta”, so the multiplier suffix for billions could be ‘j’ rather than ‘b’. In that case, you would define a new localization key:

```
ct.element.number.formatting.multiplier.j=1000000000
```

For further information about the **caplin.element.parser.LocalisedAmountParser** class, see the **Caplin Trader API Reference** document.

4.4 D. Preparing a common terms translation dictionary

This step is not mandatory, but is highly recommended if the screens and dialogs that your Caplin Trader application presents to the end-user contain many financial terms, abbreviations, and acronyms.

Translators can have difficulty porting specialist terms into the target language, for a number of reasons:

- ◆ A translator may not be familiar with the business or technical area to which the terms relate.
- ◆ The term as it appears in a localization properties file may lack context, so it is not clear to the translator what the exact meaning is, and therefore what its translation should be.

For example, in Fixed Income trading there is a type of bond trade called a “butterfly”. In some other language this type of trade may have a different name that is *not* the word for “butterfly” in this language. Lacking the appropriate context and /or business familiarity, a translator might translate the term literally, so at run time end-users see a meaningless message on the screen.

- ◆ There may be several translators working on the software and they may have different interpretations of the same term used in different places, so the resulting translations are inconsistent.

To overcome these issues, it is recommended that you prepare a dictionary of common terms for your application. For example:

Common terms translation dictionary

Term	Area	Definition	Translation (locale p1_PL)	Translation notes
...				
Butterfly	Bond Trading	A butterfly trade is	<To be completed – see E. Translating common terms 21>	<To be completed – see E. Translating common terms 21>
...				

Where a particular term can have different meanings depending on context, it should have an entry in the dictionary for each such context, since the translation of the term may be different for each entry. The “Area” and “Definition” columns should be used to clearly define the context of each dictionary entry.

- Translate the terms in the dictionary to the language of the new locale (see [E. Translating common terms](#) | 21>).

This would normally be the first task of the translators working on the application. They should then use the dictionary to help ensure that the application itself is correctly translated.

Tip: If you previously prepared a common terms translation dictionary when implementing another locale, use that for the new locale.

4.5 E. Translating common terms

- Once you have prepared the English version of the translation dictionary, take a copy of it for the new locale, and translate the terms in this copy to the language of the new locale. For example (this is a hypothetical example):

Common terms translation dictionary with Polish translations

Term	Area	Definition	Translation (locale pl_PL)	Translation notes
...				
Butterfly	Bond Trading	A butterfly trade is	Ptak	Equivalent term in Polish is "bird".
...				

The "Translation notes" column is filled in by the translators, and is useful aid to subsequent users of the dictionary who do not understand the language of the new locale.

- Make the translated dictionary available to the translators who are to work on the software translation (see [F. Translating the text](#)^[21]).

4.6 F. Translating the text

- Supply the translators with:
 - The translation dictionary, containing the definitions of common terms and their translation to the language of the new locale.
 - The *en.properties* files containing the (American) English text strings, so that the translators have a reference source.
 - The *<new-locale>.properties* files containing the (American) English text strings that are to be overwritten with their equivalents in the language of *<new-locale>*.
- If the translation is to a variant locale for which there is already a translation to a base locale, supply the localization properties files for the base locale as well, so the translators can refer to them.

For example, if the translation is to Canadian French (locale *fr_CA*), supply the base French localization properties files *fr.properties*.

Note: The content of the *<new-locale>.properties* files must be in the **UTF-8** character set *without* a **Byte-Order Mark** at the start of the file (see [Assumptions and Restrictions](#)^[4b]).

Using translation memory or terminology management software

The process of translating the Caplin Trader application may be aided by software tools. These tools often have a **translation memory** feature (a database of previous translations that helps the translators to ensure consistency and accuracy when translating more material), or **terminology management software**.

- If the translators are using a translation tool, they should load the translation dictionary into its translation memory or terminology database.

Guidelines for translators

This section contains some guidelines for translators who need to work with localization properties files.

- Edit the `<new-locale>.properties` files. For example if you are translating into standard French, edit the files called `fr_FR.properties`.
- The translated texts must be in the **UTF-8** character set.
The text in the supplied `.properties` files is in UTF-8 format. Make sure that your text editor supports UTF-8, and has been configured to read from and write to the `.properties` files in this format, with no **Byte-Order Mark** at the start of the file.

Each item to be translated has the form:

```
<localization-key>=<text-to-translate>
```

There can be spaces either side of the = sign.

`<localization-key>` is a localization key that uniquely identifies the text to be translated.

For example:

```
ct.core.download.message.download.complete=Your download should have completed.
```

In this case the `<localization-key>` is `ct.core.download.message.download.complete`.

Tip: The English version of the texts is also available in files called `en.properties`. You can use these files for reference, but do not edit them.

- Replace the `<text-to-translate>` with the equivalent text in the language of the new locale.
For example, if the locale is `fr_FR`, the edited properties file (`fr.properties`) would contain:

```
ct.core.download.message.download.complete=Le téléchargement devrait avoir terminé.
```

Note: Do not change the `<localization-key>` to the left of the '=' sign.

- Do not translate or otherwise edit lines beginning with a hash character '#'; they are internal comments.

```
# This is a comment
```

- You can split the translated text across multiple lines in the properties file by terminating each line, except the last one, with a space followed by a backslash character: ' \'. White space at the beginning and end of a line is ignored. The line splits only affect how the text is presented in the properties file, not how it is displayed at run time.

For example:

```
ct.core.download.message.download.complete=Lorem ipsum dolor sit amet, \
sed diam nonummy nibh.
```

This results in the following text being displayed for this localization key when the application is run:

Lorem ipsum dolor sit amet, sed diam nonummy nibh.

- The text to be translated may contain named variables inside square brackets []. When the application runs, these items are replaced with text or other information that is determined by the application.

For example:

```
user.welcome.message=Hello [username].
```

When the application displays the above message, [username] is replaced with the name of the logged in user:

Hello John.

- Do not change the names of variables (the contents of the square brackets), or alter/remove the square brackets surrounding them, otherwise the application may not work correctly.

However, you may need to change the *position* of a variable within the translated text so that the translation makes sense according to the grammar of the target language.

For example:

English (*en.properties*) – as supplied

```
ct.core.book.open=Open the [color] book.
```

Running the application in the English locale (*en_US*) would display a message like this:

Open the red book.

French (*fr.properties*) – after translation

```
ct.core.book.open=Ouvrez le livre [color].
```

In the French translation the variable must *follow* the word for book. Running the application in the French locale (*fr_FR*) would display a message like this:

Ouvrez le livre rouge.

Translating for a variant locale

The required translation may be for a locale that is a variation of an existing base locale. For example, there may already be localization properties files called *fr.properties* that contain the translations for the Standard French base locale, `fr_FR`. An example of a variant locale is Canadian French, `fr_CA`.

- To translate into a variant locale, edit the supplied *<new-variant-locale>.properties* files, using the *<base-locale>.properties* files as a reference.

For example, to translate into Canadian French, you would edit the supplied *fr_CA.properties* files, using the *fr.properties* files as a reference:

English (*en.properties*) – as supplied

```
ct.core.battery.dialog=Change the battery.
```

French (*fr.properties*) – as supplied

```
ct.core.battery.dialog=Changer la pile.
```

Canadian French (*fr_CA.properties*) – after translation

```
ct.core.battery.dialog=Changer la batterie.
```

Translation dictionary and translation memory

- Use the preferred translations in the **translation dictionary** (if one is supplied).

Access to the translation dictionary may be through the **translation memory** or **terminology management software** built in to the translation tool you are using.

After Translation

Tidying up the translated properties files for variant locales

- Ensure that the properties files for a variant locale only contain translations that are different to those in the base local. You may need to compare the contents of each translated localized properties file against the file(s) for the base locale, and remove the localization properties that do not differ.

For example:

The base locale is `fr_FR`, and the properties file `fr.properties` contains:

```
ct.core.download.message.download.complete=Le téléchargement devrait avoir terminé.  
ct.core.battery.dialog=Changer la pile.
```


The variant locale is `fr_CA`, and the translated properties file `fr_CA.properties` contains:

```
ct.core.download.message.download.complete=Le téléchargement devrait avoir terminé.  
ct.core.battery.dialog=ct.core.battery.dialog=Changer la batterie.
```

Remove from `fr_CA.properties` the localization property with key `ct.core.download.message.download.complete`, because its value has not changed:

```
ct.core.battery.dialog=ct.core.battery.dialog=Changer la batterie.
```

4.7 G. Localizing CSS

(This step is not shown in the diagram in [Translating to the new locale](#) )

You may need to modify some of the CSS files that define the styling of display components in your application. For example, styling that defines the pixel width of a text area may need to be changed to accommodate a larger amount of text in the translation.

To create a localized CSS file:

- Copy the file to a new file called `<css-name>-<new-locale>.css`
For example, if the original CSS file is called `ticket-style.css`, and the new locale is `fr_FR` (standard French), copy the file to `ticket-style.css-fr_FR.css`.
 - If the original CSS file is in one of your own blades, or in your application, put the new file in the same folder as the original.
 - If the original CSS file is in a Caplin-supplied blade, put the new file in the folder where the application level CSS resides. This ensures that any subsequent updates to the Caplin-supplied blade will not overwrite your changes.
- Change the styles in the new file as required.

5 Building and testing Caplin Trader for the new locale

Checking the translated files

- Check the translated localization properties files for completeness and any obvious mistakes.

You may need to do additional translation work to fix obviously missing and/or incorrect translations.

Tip: Caplin supplies some reporting tools that can help you to do this; see [Tools to help with localization](#).

Building the application

- Put the translated localization properties files for the new locale into your code repository.
- Build the Caplin Trader application.

Testing the application

Run the application in the new locale and check the quality of the localization.

- You should comprehensively test by invoking as many of the screens, menus, and dialogs as you can.
- A speaker of the native language of the new locale (who was not involved in the original translation) should check:
 - the translated texts,
 - the formats of dates, numbers, and so on.
- Check for any layout problems resulting from the translated text occupying more space than the English version.
- If the language script of the new locale is read from right to left, check for any alignment problems with text and with columns of numbers.

6 Reusing a localized Caplin-supplied blade

You may have localized a Caplin-supplied blade for use in application A (because Caplin does not supply the blade with support for the particular locale). If you wish to use the same blade in another application B, supporting the same locale:

- Find all the localization properties for the blade in application A.

- The properties are in the application level properties file:

`<app-root>\novotrader\novox\i18n\<locale>\<locale>.properties`

For example:

`<app-root>\novotraderA\novoxA\i18n\fr\fr.properties`

- Look in this file for localization properties with names of the form:

`blade.<blade-name>.<property-description>`

For example, the FX Tile blade could have a property called:

`blade.fxtile.tier.limit.warning`

- Copy these properties across to the application level localization properties file for application B.

For example, copy them to:

`<app-root>\novotraderB\novoxB\i18n\fr\fr.properties`

7 Tools to help with localization

Caplin supplies a number of reporting tools that you can use to help ensure your application is fully localized:

- To generate the reports, run the following ant build command from your Caplin Trader development environment:

```
ant generate-il8n-reports
```

The reports are generated in the folder:

```
<app-root>\reports\i18n\
```

The following sections explain, with examples, what each report contains.

- ◆ [Report of Localization Keys Details](#) ²⁹
- ◆ [Report of Missing Localization Keys](#) ³⁰
- ◆ [Tabular Report of Missing Localization Keys](#) ³¹
- ◆ [Localization Token Values Report \(spreadsheet\)](#) ³²

7.1 Report of Localization Keys Details by key name

Description

Lists all the localization keys that are used in source code (XML, HTML, and JavaScript), covering the Caplin Trader framework, all the blades used in your Caplin Trader application, and the Caplin Trader application itself. The report is ordered by key name.

Purpose

Helps you identify the source code files that contain each localization key, and the position(s) of the key in each file.

Example Localization Key Details report (fragment)

```
Report of Localization Key Details
=====

This report gives detailed information about each Localization Key in the code base.
This includes:
    The files that contain the Localization Key.
    The line number where the Localization Key is located in the file.

1).  ct.core.theme.alert.template.caption
    Localization Key Details (3):
    File: D:\Development\CT2\main\caplintrader\build\themes\blank\
        layout-themes\blank.xml
        Line Number: 215
    File: D:\Development\CT2\main\caplintrader\build\themes\noir\
        layout-themes\noir.xml
        Line Number: 220
    File: D:\Development\CT2\main\caplintrader\build\themes\pastel\
        layout-themes\pastel.xml
        Line Number: 237

2).  ct.core.griddefinitions.metals
    Localization Key Details (2):
    File: D:\Development\CT2\main\caplintrader\webapp\conf\gridDefinitions.xml
        Line Number: 553
    File: D:\Development\CT2\main\caplintrader\webapp\conf\gridDefinitions.xml
        Line Number: 692

...

***** End of Report *****
```


7.2 Report of Missing Localization Keys by locale

Description

Lists the localization keys that are used in source code (XML, HTML, and JavaScript), but are not present in any localization properties files for a locale. The report is ordered by locale.

Purpose

Helps you identify the localization keys that have no property definitions in a locale.

Example Missing Localization Keys report (fragment)

```
Report of Missing Localization Keys
=====

This report shows all missing Localization Keys by locale.
A Key is missing if it is not found in any Localization Properties file.

1).      en_US      [ 0 Missing Localization Keys ]
2).      en_GB      [ 0 Missing Localization Keys ]
3).      fr_FR      [ 8 Missing Localization Keys ]
        blade.fxtile.buy ,
        blade.fxtile.sell ,
        blade.fxtile.status.pending ,
        blade.fxtile.status.prices.stale ,
        ct.core.command.cancel ,
        ct.core.command.ok ,
        ct.framework.connection.state.connecting ,
        ct.grid.add.column ,
        ct.grid.data.unavailable ,

        ***** End of Report *****
```

Variant locales

Tip: The report will not tell you if a required variant translation for a property is missing.

This is because the report only lists those keys that are not found in any localization properties files for both the variant locale and the base locale. For example, assume you have defined the locales `fr_FR` (standard French base locale) and `fr_CA` (Canadian French variant locale). The localization property `blade.fxtile.buy` would only appear in this report if it was not defined in any `fr_CA.properties` files or `fr.properties` files.

7.3 Tabular Report of Missing Localization Keys by key name

Description

Lists the localization keys that are used in source code (XML, HTML, and JavaScript), but are not defined in any localization properties files for a locale. The report is ordered by key name.

Purpose

Helps you identify the localization keys that have no property definitions in a locale.

Example Tabular Report of Missing Localization Keys (fragment)

Tabular Report of Missing Localization Keys				
=====				
This report shows each Localization Key, and for each locale, whether or not the key is present in at least one Localization Properties file.				
Explanation of Terms:				
"+" : The Localization Key is present for the locale (in at least one Localization Properties file).				
Missing" : The Localization Key is missing for the locale (not found in any Localization Properties files).				
=====				
No.	Localization Key	en_GB	en_US	fr_FR

1	blade.fxtile.buy	+	+	Missing
2	blade.fxtile.sell	+	+	Missing
3	blade.fxtile.status.pending	+	+	Missing
4	blade.fxtile.status.prices.stale	+	+	Missing
5	blade.fxtile.status.trading.unavailable	+	+	+
6	blade.fxtile.tier.limit.warning	+	+	+
7	blade.fxtile.trade.result.confirmed	+	+	+
8	blade.fxtile.trade.result.expired	+	+	+
9	blade.fxtile.trade.result.passed	+	+	+
10	blade.fxtile.trade.result.unknown	+	+	+
11	caption.fi_watchlist	+	+	+
12	caption.fx_watchlist	+	+	+
13	caption.select_date	+	+	+
14	chart.date	+	+	+
15	chart.maturity_year	+	+	+
16	chart.yield	+	+	+
17	chart.yield_pc	+	+	+
18	colorpickerdialog.caption.colours	+	+	+
19	consolelogger.caption.log_console	+	+	+
20	ct.core.command.cancel	+	+	Missing
21	ct.core.command.ok	+	+	Missing
22	ct.core.dialog.header.alert	+	+	+
23	ct.core.dialog.header.confirm	+	+	+
...				
45	ct.framework.connection.state.connected	+	+	+
46	ct.framework.connection.state.connecting	+	+	Missing
...				
68	ct.grid.add.column	+	+	Missing
69	ct.grid.data.unavailable	+	+	Missing
70	ct.grid.deleting.message	+	+	+
...				

Total	825	0	0	154
=====				
***** End of Report *****				

7.4 Localization Key Values Report (csv format)

Description

This report lists:

- ◆ The localization keys that are used in source code (XML, HTML, and JavaScript).
- ◆ The source code files that contain each key.
- ◆ The value of each key in each supported locale, or “missing” if the key is not defined in any localization properties files for the locale.

The report is ordered by key name.

Purpose

Helps you identify the value of each localization key in supported locales, and the source code files that contain the key. The report is produced in **CSV** format, and can be loaded into a spreadsheet.

The following picture shows an example report that has been loaded into a Microsoft Excel® spreadsheet.

1	Tabular Report of Missing Keys				
2					
3	This report shows details of each Localization Key for each supported locale.				
4					
5	Explanation of Terms:				
6	**** Missing ****: This means that the Localization Key is missing for the locale (not found in any Localization Properties files).				
7					
8					
9	No.	Keys	Key Details	fr_FR	en_GB
				en_US	
	1	blade.fxtile.buy	File: D:\Development\CT2\main\blades\FxTile\src\caplin\FxTile\FxTilePresentationModel.js Line Number: 88 File: D:\Development\CT2\main\blades\FxTile\src\caplin\FxTile\FxTilePresentationModel.js Line Number: 95	*** Missing ***	Buy
				Buy	
10	2	blade.fxtile.sell	File: D:\Development\CT2\main\blades\FxTile\src\caplin\FxTile\FxTilePresentationModel.js Line Number: 87 File: D:\Development\CT2\main\blades\FxTile\src\caplin\FxTile\FxTilePresentationModel.js Line Number: 94	*** Missing ***	Sell
				Sell	
11	...				
12	18	colorpickerdialog.caption	File: D:\Development\CT2\main\caplintrader\build\xml\caplintrader.xml Line Number: 54	couleurs	colours
13	...				
14	20	ct.core.command.cancel	File: D:\Development\CT2\main\libraries\dom\src\caplin\dom>alert\StandardAlertDispatcher.js Line Number: 38 File: D:\Development\CT2\main\libraries\dom\src\caplin\dom>alert\WebcentricAlertDispatcher.js Line Number: 30	Annuler	Cancel
				Cancel	
15	21	ct.core.command.ok	File: D:\Development\CT2\main\libraries\dom\src\caplin\dom>alert\StandardAlertDispatcher.js Line Number: 37 File: D:\Development\CT2\main\libraries\dom\src\caplin\dom>alert\WebcentricAlertDispatcher.js Line Number: 29	*** Missing ***	OK
				OK	
16	...				
17	Total	825		154	0
18				0	0

Localization Token Values Report (spreadsheet)

8 Troubleshooting

This document describes some typical problems that you might encounter when localizing a Caplin Trader application. It describes the symptoms of each problem, lists the possible diagnoses, and gives guidance on possible solutions.

- ◆ [The application displays “??? <key.name> ???” in some text fields](#) ^[33]
- ◆ [Some translations are missing in the variant locale](#) ^[33]

8.1 ??? <key.name> ??? displayed in text fields

Problem

I have translated my Caplin Trader application to a new locale and now the application displays

??? <key.name> ???

in some fields in place of text (<key.name> is the name of a localization key).

Diagnosis

??? <key.name> ??? indicates that the named localization key occurs somewhere in the code of the application, but does not have a value defined for the new locale (it does not appear in any localization properties file for the locale).

Solution

Define the affected localization keys in the appropriate <new-locale>.properties files, together with their values (text) in the language of the locale, and rebuild the application.

Tip: The [Tabular Report of Missing Localization Keys](#) ^[31] shows all the localization keys that do not have values. The [Localization Key Values Report \(csv format\)](#) ^[32] gives the details of where localization keys are used in code.

8.2 Translations missing in variant locale

Problem

I have implemented a variant locale, but the translations for some items are not in the variant language when they should be.

Diagnosis

The affected items were probably missed in translation. The localization properties files for a variant locale only contain those properties that differ from their equivalents in the base locale. If an item is not translated to the variant locale, the base locale equivalent is displayed at run time.

Solution

Define the localization keys in the appropriate <variant-locale>.properties file, together with their values (text) in the language of the variant locale, and rebuild the application.

9 Appendix A: Date and time format specifiers

The following table lists the format specifiers that can be used in the localization properties `ct.i18n.date.format` and `ct.i18n.date.format.long` (see [C. Editing localization properties for formatted items](#)^[16]). They are the formats defined for the **Date** class in the Ext JS JavaScript library. The definitive reference information for the formats can be found on the Ext JS API web site at <http://dev.sencha.com/deploy/dev/docs/?class=Date>.

Format Specifier	Description	Values / Examples
d	Day of the month, 2 digits with leading zeros.	01 to 31
D	A short textual representation of the day of the week.	Mon to Sun
j	Day of the month without leading zeros.	1 to 31
l	A full textual representation of the day of the week.	Examples: Monday Tuesday ... Saturday Sunday
N	ISO-8601 numeric representation of the day of the week.	1 (for Monday) through 7 (for Sunday)
S	English ordinal suffix for the day of the month, 2 characters. Note: This format specifier can only be used in English locales.	st, nd, rd or th. For use with format j Example: 2nd
w	Numeric representation of the day of the week.	0 (for Sunday) to 6 (for Saturday)
z	The day of the year (starting from 0).	0 to 364 (365 in leap years)
W	ISO-8601 week number of year. Weeks start on a Monday.	01 to 53
F	A full textual representation of a month, such as January or March.	January to December
m	Numeric representation of a month, with leading zeros.	01 to 12
M	A short textual representation of a month.	Jan to Dec
n	Numeric representation of a month, without leading zeros.	1 to 12
t	Number of days in the given month.	28 to 31
o	ISO-8601 year number (identical to Y, but if the ISO week number w belongs to the previous or next year, that year is used instead).	1998 2004
Y	A full numeric representation of a year (4 digits).	1999 2003

Format Specifier	Description	Values / Examples
y	A two digit representation of a year.	99 03
a	Lowercase “ante meridiem” (“before midday”) and “post meridiem” (“after midday”), used for times that are in 12 hour format.	am or pm
A	Uppercase “ante meridiem” (“before midday”) and “post meridiem” (“after midday”), used for times that are in 12 hour format.	AM or PM
g	12-hour format of an hour without leading zeros.	1 to 12
G	24-hour format of an hour without leading zeros.	0 to 23
h	12-hour format of an hour with leading zeros.	01 to 12
H	24-hour format of an hour with leading zeros.	00 to 23
i	Minutes, with leading zeros.	00 to 59
s	Seconds, with leading zeros.	00 to 59
u	Decimal fraction of a second (minimum 1 digit, arbitrary number of digits allowed).	Examples: 001 (that is, 0.001s) 100 (that is, 0.100s) 999 (that is, 0.999s) 999876543210 (that is, 0.999876543210s)
O	Difference between the time being converted and Greenwich Mean Time (GMT) in hours and minutes, with no character between the hours and minutes.	Example: +1030 (time is 10 hours and 30 minutes ahead of GMT)
P	Difference between the time being converted and Greenwich Mean Time (GMT) in hours and minutes, with a character between the hours and minutes (colon in the example).	Example: -08:00 (time is 8 hours and 0 minutes behind GMT)
T	Time zone abbreviation of the machine running the code.	Examples: EST, MDT, PDT
Z	Time zone offset in seconds (negative if west of UTC, positive if east)	-43200 to 50400
c	ISO 8601 date format. For details, see the Ext JS API web site at http://dev.sencha.com/deploy/dev/docs/?classes=Date .	Examples: 1991 1992-10 1993-09-20 1994-08-19T16:20+01:00 1995-07-18T17:21:28-02:00 1996-06-17T18:22:29.98765+03:00 1997-05-16T19:23:30,12345-0400 1998-04-15T20:24:31.2468Z 1999-03-14T20:24:32Z 2000-02-13T21:25:33 2001-01-12 22:26:34
U	Seconds since the Unix Epoch (January 1 1970 00:00:00 GMT)	Examples: 1193432466 -2138434463

Format Specifier	Description	Values / Examples
M\$	Microsoft AJAX serialized dates. Note: This format specifier should only be used in code. It should not be used in localization properties.	Examples: \\/Date(1238606590509)\\/ (UTC milliseconds since epoch) \\/Date(1238606590509+0800)\\/

10 Glossary of terms and acronyms

This section contains a glossary of terms, abbreviations, and acronyms relating to the localization of Caplin Trader applications.

Term	Definition
Base locale	<p>The primary locale for a particular language. This identifies the main set of localization properties files for a locale. For example, the primary locale for English is <code>en_US</code> (American English), which identifies the localization properties files called <i>en.properties</i>.</p> <p>Also see variant locale.</p>
Base localization properties file	<p>A localization properties file that contains the localization properties for a base locale.</p>
Blade	<p>A business component that provides domain specific functionality in a Caplin Trader application. Each Caplin Trader blade implements a small, well-defined set of closely related functions.</p>
Byte-Order Mark	<p>The Byte-Order Mark (BOM) is a Unicode character used to signal the byte order of a text file or stream. Byte order has no meaning in UTF-8, so a Byte-Order Mark only serves to identify a file or text stream as UTF-8 encoded, or to identify that it was converted from another format that has a Byte-Order Mark.</p> <p>In Caplin Trader, localization properties files, though encoded in UTF-8, must not contain Byte-Order Marks.</p> <p>[Definition adapted from Wikipedia contributors, "Byte order mark," <i>Wikipedia, The Free Encyclopedia</i>, http://en.wikipedia.org/wiki/Byte_order_mark (accessed April 6, 2011)]</p>
Caplin Trader	<p>A web application framework for constructing browser-based financial trading applications.</p>
Caplin Trader application	<p>A browser-based client application that has been built using Caplin Trader.</p>
CSV	<p><u>C</u>omma <u>S</u>eparated <u>V</u>alues.</p> <p>A file format in which tabular data is stored in plain text form, with lines of the file representing rows of a table, and commas separating the individual items in each "row". Most spreadsheet applications are able to read CSV files and display the data in tabular form.</p>
Decimal radix character	<p>The character that is used in a decimal number to mark the separation between the integer part of the number (to the left) and the fractional part (to the right).</p> <p>In the USA (locale <code>en_US</code>) and UK (locale <code>EN_GB</code>) this is a dot, as in 1234.56789</p>
Display component	<p>A GUI component of Caplin Trader that can be rendered in a page on the screen.</p> <p>The term also refers to the JavaScript code that generates the component and handles its user interaction. Caplin Trader has a number of pre-defined, customizable display components, such as Grids, Trade Tiles, and the Blotter.</p>
i18n	<p>Abbreviation for internationalization ("i<18chars>n").</p>

Term	Definition
Internationalization	An alternative term for the process of adding localization support to a software application.
L10n	Abbreviation for localization ("L<10chars>n").
Locale	The aggregate of a software user's language, country, and any special variant preferences that the user wants to see in their user interface (such as particular date formats and number formats). Such an aggregate is uniquely identified by a locale identifier .
Locale identifier	<p>The unique identification of a locale.</p> <p>In Caplin Trader, locale identifiers are of the form: <language-code>_<country> For example: en_US (American English) or en_GB (British English).</p>
Key	In this document, this term is short for localization key .
Localization	<p>The process of implementing a particular locale within a software application. This typically involves:</p> <ul style="list-style-type: none">◆ Translating text that appears on screen, and in reports and logs, into the language of the locale.◆ Defining the formats of dates, numbers, currencies, and so on, according to the requirements of the locale.
Localization key	<p>The string of characters that identifies a particular localization property. For example: blade.fxtile.buy</p> <p>A localization key can belong to a localization namespace.</p> <p>For more information, see the document Caplin Trader: Localization Overview And Concepts.</p>
Localization namespace	<p>A namespace convention and standard used for localization keys that partitions the keys according the particular areas of a Caplin Trader application they apply to.</p> <p>For more information, see the document Caplin Trader: Localization Overview And Concepts.</p>
Localization property	<p>A localization key and the value of that key in a given locale. A localization property has the general form: <keyname>=<value> For example: blade.fxtile.buy=Buy</p> <p>For more information, see the document Caplin Trader: Localization Overview And Concepts.</p>
Localization properties file	<p>A file containing a set of localization properties for a given locale. Also called (when in context) a "properties file".</p> <p>Localization properties files are UTF-8 encoded, with no Byte-Order Mark.</p>
Localization support	The functions and features within a software application that allow it to (easily) support multiple locales . The software can then be localized to actually implement a given locale or locales (see localization).

Term	Definition
Localization token	<p>A marker embedded in the Caplin Trader software that indicates that an item, such as text, a date, or a number, must be localized. The marker takes the form <code>@{localization-key}</code>.</p> <p>When the software is converted to a given locale (see localization), the token is replaced with the corresponding text, date format, number format, and so on, for that locale.</p> <p>Each localization token identifies a localization key.</p> <p>Also called (when in context) a “token”.</p> <p>For more information, see the document Caplin Trader: Localization Overview And Concepts.</p>
Properties file	<p>In this document, this term is short for localization properties file.</p>
Token	<p>In this document, this term is short for localization token.</p>
Terminology management software	<p>Terminology management software allows a translator to automatically search a terminology database for terms appearing in a document, either by automatically displaying terms in the translation memory software interface, or through the use of hot keys to view the entry in the terminology database.</p> <p>[Wikipedia contributors, "Computer-assisted translation," <i>Wikipedia, The Free Encyclopedia</i>, http://en.wikipedia.org/wiki/Computer-assisted_translation (accessed April 6, 2011).]</p>
Translation memory	<p>A translation memory, or TM, is a database that stores "segments", which can be sentences or sentence-like units (headings, titles, or elements in a list), that have been previously translated. A translation-memory system stores the words, phrases and paragraphs that have already been translated, to aid human translators.</p> <p>[Wikipedia contributors, "Translation memory," <i>Wikipedia, The Free Encyclopedia</i>, http://en.wikipedia.org/wiki/Translation_memory (accessed April 6, 2011).]</p>
Unicode	<p>A computing industry standard for the consistent encoding, representation, and handling of text expressed in most of the world's writing systems.</p> <p>[Wikipedia contributors, "Unicode," <i>Wikipedia, The Free Encyclopedia</i>, http://en.wikipedia.org/wiki/Unicode (accessed April 6, 2011).]</p> <p>The Unicode standard is maintained by the Unicode Consortium (see http://unicode.org).</p>
UTF-8	<p>A way of encoding Unicode using multi-byte characters. The localization properties files used in Caplin Trader applications must be UTF-8 encoded.</p>
Variant locale	<p>A locale that is a variant of the base locale for a particular language. For example, a variant locale for English is <code>en_GB</code> (British English), which identifies the variant localization properties files called <code>en_GB.properties</code>. (the association between the locale and the properties files is defined by configuration).</p>
Variant localization properties file	<p>A localization properties file that contains the localization properties for a variant locale.</p>

Contact Us

Caplin Systems Ltd
Cutlers Court
115 Houndsditch
London EC3A 7BR
Telephone: +44 20 7826 9600
www.caplin.com

The information contained in this publication is subject to UK, US and international copyright laws and treaties and all rights are reserved. No part of this publication may be reproduced or transmitted in any form or by any means without the written authorization of an Officer of Caplin Systems Limited.

Various Caplin technologies described in this document are the subject of patent applications. All trademarks, company names, logos and service marks/names ("Marks") displayed in this publication are the property of Caplin or other third parties and may be registered trademarks. You are not permitted to use any Mark without the prior written consent of Caplin or the owner of that Mark.

This publication is provided "as is" without warranty of any kind, either express or implied, including, but not limited to, warranties of merchantability, fitness for a particular purpose, or non-infringement.

This publication could include technical inaccuracies or typographical errors and is subject to change without notice. Changes are periodically added to the information herein; these changes will be incorporated in new editions of this publication.

Caplin Systems Limited may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

This publication may contain links to third-party web sites; Caplin Systems Limited is not responsible for the content of such sites.