L-Soft Sweden AB

Translating the LISTSERV Maestro User Interface

LISTSERV® Maestro, Version 1.2

Last Updated: March 18, 2004 11:10 AM

Information in this document is subject to change without notice. Companies, names and data used in examples herein are fictitious unless otherwise noted. L-Soft International, Inc. does not endorse or approve the use of any of the product names or trademarks appearing in this document.

Manuals for LISTSERV are available in ASCII-text format from LISTSERV and in PDF format from **ftp.lsoft.com**. They are also available on the World Wide Web at the following URL:

URL: http://www.lsoft.com/manuals/index.html

L-Soft invites comment on its manuals. Please feel free to send your comments by e-mail to: MANUALS@LSOFT.COM

Contents

Section 1

About This Document

Every effort has been made to ensure that this document is an accurate representation of the functionality of LISTSERV Maestro. As with every software application, development continues after the documentation has gone to press, so small inconsistencies may occur. We would appreciate any feedback on this manual. Send comments by e-mail to: MANUALS@LSOFT.COM

The following documentation conventions have been used in this manual:

- Quotations from the screen will appear in italics enclosed within quotation marks.
- Clickable buttons will appear in bold.
- Clickable links will appear in bold.
- Directory names, commands, and examples of editing program files will appear in Courier New font.
- Emphasized words or phrases will be underlined.
- Some screen captures have been cropped for emphasis or descriptive purposes.

 This symbol denotes an important note or warning.

This symbol denotes optional advice to save time.

.

Section1 Introduction

This document provides an overview of how to add your own translated versions of the text displayed in the LISTSERV Maestro user interface and online help to the LISTSERV Maestro program. L-Soft international does not provide the actual translations of the English language text contained in LISTSERV Maestro. The translation kit does provide the means and instructions on how to add your own translation.

The tasks for making and installing a successful translation fall into two basic categories, administering the translation files within the application, and performing the actual translating of the interface files. The same person may be responsible for both tasks, or the tasks may be shared among individuals. To administer the translation files it is necessary to have administrative access to LISTSERV Maestro as well as system administrator privileges for the server(s) running LISTSERV Maestro. Administering the translation files involves downloading the translation kit, maintaining the file structure integrity, and installing the translated files.

Performing the actual translation of the text displayed in the interface from English to another language involves strictly following the rules and formatting outlined in this document. Translating the files requires the use of a ZIP tool and a text editor.

Section 2 Requirements

To be able to translate the LISTSERV Maestro user interface into another language, you must have the following:

- An installation of LISTSERV Maestro.
- A translation kit for the target language that matches the LISTSERV Maestro version that you have installed.
- A ZIP utility like WinZip® or PKZIP®.
- A text editor such as Microsoft® Notepad or a word processing program that is able to save files in plain text format. The editor/word-processing program must also be able to save the file using the character encoding (also called character set or charset) that matches the target language. More information on character sets is available in the LISSTERV Maestro User's Manual Appendix B
- The ability to edit HTML code, understand the meaning of the various HTML tags, and how to escape reserved HTML characters.

Section 3 Translation Steps

The translation process is comprised of the following steps:

1. Unzipping the files from the translation kit, keeping the file and folder structure intact.
2. Editing and translating the text files into the target language.
3. Zipping the edited files back into a ZIP archive, preserving the original file names and folder structure.
4. Uploading or copying the ZIP archive to the "`lib`" folder(s) in the LISTSERV Maestro installation.
5. Restarting LISTSERV Maestro on all servers where there is a new ZIP file.

3.1 Translation Kit

The translation kit for LISTSERV Maestro comes in the form of a ZIP file. This ZIP file contains a number of text files. Each target language comes in its own ZIP file, and there is a matching ZIP file for each LISTSERV Maestro version. For example, the file "`Maestro-1.2-TransKit-de.zip`" is the German language (="de") translation kit for LISTSERV Maestro 1.2.

Caution: Do not use a language kit that is not meant for your target language or for your LISTSERV Maestro version. Damage to the installation and loss of data could result.

3.1.1 Translation Kit Folder Structure

The ZIP file contains a number of files that are organized in a subfolder structure. An example of this structure for LISTSERV Maestro 1.2 and the German language kit appears below:

```
com\lsoft\hub\adminui\res\de\*.*
com\lsoft\hub\adminui\res\de\userPassword\*.*

com\lsoft\hub\hubplugin\it004\res\de\*.*

com\lsoft\lui\hubplugin\it004\res\de\*.*
com\lsoft\lui\hubplugin\it004\res\de\include\*.*

com\lsoft\lui\res\de\*.*
com\lsoft\lui\res\de\content\*.*
com\lsoft\lui\res\de\handledJobs\*.*
com\lsoft\lui\res\de\newJobs\*.*
com\lsoft\lui\res\de\recipients\*.*
com\lsoft\lui\res\de\reports\*.*
com\lsoft\lui\res\de\senderProfiles\*.*
com\lsoft\lui\res\de\testSend\*.*
com\lsoft\lui\res\de\tracking\*.*
com\lsoft\lui\res\de\unhandledJobs\*.*
com\lsoft\lui\res\de\userSettings\*.*

com\lsoft\lui\trk\hubplugin\it010\res\de\*.*
```

The folder structure displayed here is only a sample. The structure for versions other than 1.2 may differ. In addition, the sample shown here is for the target language German, represented by the two-letter ISO-language-code "`de`". This code is reflected in the name of the "`de`" subfolders that appear in the folder structure. For other languages, these folders have the respective code for that language, for example "`fr`" for French and "`sv`" for Swedish.

The folder structure is divided into five main groups. The most important group is the one containing the resources for the Maestro User Interface:

`com\lsoft\lui\res`

The other four groups contain the resources for the Administration Hub. The resources are separated into component unspecific resources and resources specific to one of the three components, the Administration Hub itself, the Maestro User Interface, and Maestro Tracker:

`com\lsoft\hub\adminui\`	component unspecific
`com\lsoft\hub\hubplugin\`	Administration Hub specific
`com\lsoft\lui\hubplugin\`	Maestro User Interface specific
`com\lsoft\trk\hubplugin\`	Maestro Tracker specific

If you only want to translate the Maestro User Interface (the part of the program that is seen by regular users) then you only need to translate the resources in the "`com\lsoft\lui\res`" hierarchy. It is not necessary to edit the files contained in the other four groups because they are related to the Administration Hub alone, and the Hub is only ever seen by administrators.

It is important that the folder structure present in the ZIP file is preserved during all steps of the translation process including unpacking the ZIP file for editing and later zipping the files into a new archive. Unpack the ZIP file so that the folder structure in it is recreated in your file system. Later, when zipping up the translated files into a new archive, make sure that the folder structure is again preserved. See the documentation of your ZIP utility for more information about how to unzip and zip files while preserving the folder structure.

3.1.2 Translation Kit Files

Inside of the folder structure of the translation kit the following files may be found, some of which may appear several times:

File Name	Type of File	Editing
`*.class`	System files	Do not change these files in any way.
`res_toc.properties`	System files	Do not change these files in any way.
`*.properties`	Resource files	These files need to be translated (except for "`res_toc.properties`", see above).
`encoding.dat`	Encoding files	Usually there is no need to change these files, except if you need to use a different charset then what the translation kit was prepared for (see below for details).

Caution: Do not delete any files and do not rename any files or folders! Doing so could result in a corrupted installation and loss of data.

The resource files (`*.properties`) contain the text strings that need to be translated. In a "fresh" (yet un-translated) language kit, these files still contain the default language, English.

3.2 Translating

To translate the resources files contained in a translation kit, first unzip all the files from the translation kit ZIP file, and save them to a suitable location in your file system, taking care that the folder structure is preserved.

Next, use a text editor or a word processing program that is capable of editing and saving plain text files, to edit and translate the resource files. L-Soft international is not providing translations of the original English language text that appears in the resource files. The files in the kit are for you to translate and save in the original file structure.

When using a word processing program, save the edited files in plain text format. Do not use the word processing program's native format (for example "`*.doc`" in Microsoft Word), to save the files. Also, be careful to save the files with their original filenames including the `*.properties` extension. The default extension for plain text files is usually `*.txt`, and your editor or word processing program may attempt to force this extension onto your files. Keep the original file names and extensions. See Section 4 for translation rules and guidelines.

When finished with your translation of the property files, zip them up into a new ZIP archive similar to the original translation kit ZIP file. The only difference will be that the ZIP archive now contains the translated versions of the files. Make sure to preserve the folder structure inside of the ZIP file when creating it. Make sure to also include all files in the ZIP file, even the files that you did not change during translation, such as the `*.class` and `res_toc.properties` files.

The new ZIP file can have any name, but we recommend that you choose a meaningful name that will show the version of LISTSERV Maestro that the file is meant for, and the language contained therein. Also, choose a name different from the original translation kit ZIP file's name, so that you will not confuse the already translated file with the original un-translated file. A name like "`Maestro-1.2-Lang-de.zip`" if your language is German ("`de`") and the kit was translated for version 1.2 is a good example.

3.2.1 Text File Charsets

When saving the translated `*.properties` text files, make sure that you save them with a character set (also called "charset" or sometimes "file encoding" or simply "encoding") that matches the language you are working with. All character sets that are supported by your Java version are allowed. Please see the following pages at Sun's Java Web site for a list of supported encoding schemes:

For Java 1.3: http://java.sun.com/j2se/1.3/docs/guide/intl/encoding.doc.html

For Java 1.4: http://java.sun.com/j2se/1.4/docs/guide/intl/encoding.doc.html

Note: If you choose to use an alias for an encoding scheme that is not listed on these pages (most notably, “ASCII” instead of “US-ASCII” or “ISO-8859-x” or “iso-8859-x” instead of “ISO8859_x”), it may not work in future versions of Java.

In general, all `*.properties` text files must use the same charset. You may not save one file using ASCII, another using ISO-8859-1 and a third using UTF-8 (or a similar mix) within the same translation kit. However, having different encoding schemes for the five main hierarchies is possible. For each hierarchy branch, the encoding used for the files therein is defined by a single entry in a text file called “`encoding.dat`” that can be found in the directories below:

Maestro User Interface resources hierarchy:
`com\lsoft\lui\res\<language>\encoding.dat`

Administration Hub, component unspecific resources hierarchy:
`com\lsoft\hub\adminui\res\<language>\encoding.dat`

Administration Hub, Hub specific resources hierarchy:
`com\lsoft\hub\hubplugin\it004\res\<language>\encoding.dat`

Administration Hub, Maestro User Interface specific resources hierarchy:
`com\lsoft\lui\hubplugin\it004\res\<language>\encoding.dat`

Administration Hub, Maestro Tracker specific resources hierarchy:

`com\lsoft\trk\hubplugin\it004\res\<language>\encoding.dat`

The folder names given above are for Maestro 1.2. For other versions, the folder names may vary.

Each “`encoding.dat`” file must be a standard ASCII text file with a single entry as follows:

`encoding=<ENCODING>`

Replace “`<ENCODING>`” with the name of the charset that was used to save all of the files in the hierarchy branch where the “`encoding.dat`” file is found. For example, for West European languages like French, German, Spanish, Swedish, and so on, you would probably use the most common “Latin 1” charset “ISO-8859-1”, so that the entry in the “`encoding.dat`” file would look like this:

`encoding=ISO-8859-1`

Each translation kit comes readily prepared with these “`encoding.dat`” files, specifying an encoding scheme that is most commonly used for the language in question. When starting a translation, look at one of the “`encoding.dat`” files to see which encoding is specified. In an original, unchanged translation kit, they will all specify the same encoding, so it does not matter which file you check. Use this same encoding when saving the translated `*.properties` files, or, if the encoding is not appropriate for your language, change the “`encoding.dat`” files. Remember to change all of them so that they specify the name of the encoding that you want to use. Find the correct name for the appropriate encoding by looking at the Java documentation page, referenced in the links above, and then use this encoding when storing the `*.properties` files.

3.3 Installing a Translated Language

To install a translated language into an existing LISTSERV Maestro installation, simply move or copy the ZIP file that you created during translation (containing the translated text files) to the “`lib`” folder inside of your LISTSERV Maestro installation folder. Do not overwrite any existing

file. Rename your ZIP file to a different, meaningful name, if there is any possibility of overwriting an existing file.

If you have only translated the Maestro User Interface part in the "`com\lsoft\lui\res`" hierarchy of the folder structure, then you only need to copy the ZIP file into the "`lib`" folder on the server where the Maestro User Interface is installed. If you have translated the full set of resource files, including the files for the Administration Hub, then you also need to copy the file into the "`lib`" folder on the server where the Administration Hub is installed. If all the LISTSERV Maestro components are on the same server, they share the same "`lib`" folder, so you would have to copy the ZIP file only once.

Restart LISTSERV Maestro on all servers where you put the new ZIP file.

3.4 Using a Translated Language

All languages that are installed are cumulative, meaning that they will all be available at the same time, in parallel. On any LISTSERV Maestro installation, the default language, English, will always be available. In addition, all other languages that were translated and installed as described in this document will be available. Individual users will only see the language that their browser has been set to display.

3.4.1 Setting Different Languages in a Web Browser

LISTSERV Maestro automatically determines the correct language for users by detecting settings in their Web browser. The Web browser sends the user's locale information to LISTSERV Maestro, and LISTSERV Maestro displays the user interface using the language matching this locale, or, if no matching language is found, using the default language, English. To choose a language in LISTSERV Maestro, set the Web browser to the desired language.

Setting the Language in Microsoft Internet Explorer

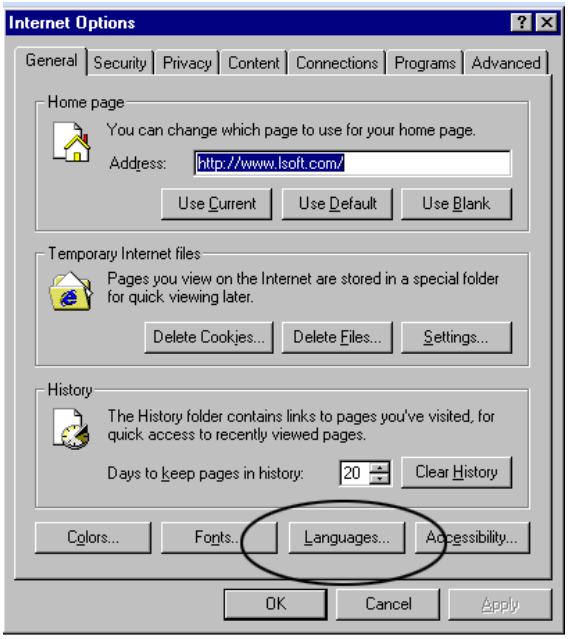
In Internet Explorer 5.5 and later:

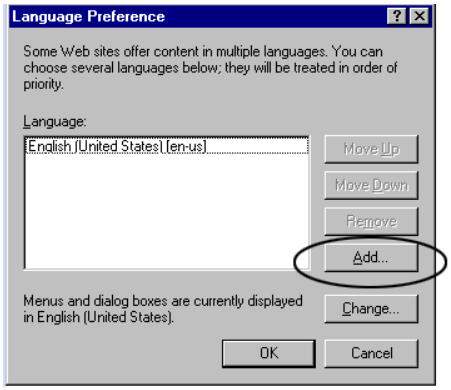
Select “Tools → Internet Options…”.

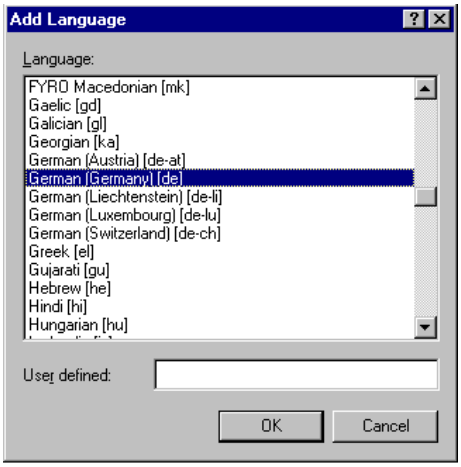
On the “General” tab, click the “Languages…” button.

Edit the list of language preferences by adding/removing languages and changing the order of the selected languages.

Setting Languages in Microsoft Internet Explorer







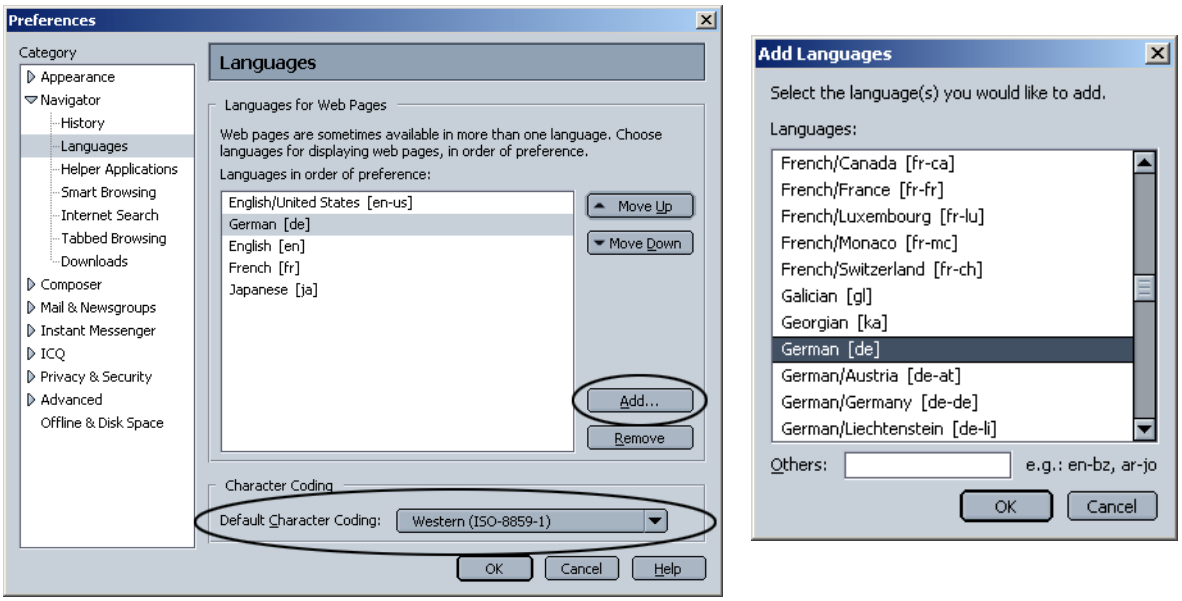
Setting the Language in Netscape

In Netscape 7.0 and later:

Select "Edit → Preferences" from the top menu bar.

From the Category list, select "Navigator → Languages". Click on the Add button, scroll through the list and select the desired language(s). You can also change the character encoding to match the language.

Setting Languages in Netscape Navigator



Section 4 Editing and Translating Files - General Concepts

The following sections describe some general concepts that need to be followed when editing and translating the text files. It is important to read all of them, and understand and employ the concepts and rules as explained. Incorrectly formatted or edited text files may, at best, cause the user interface to look strangely, and, at worst, cause LISTSERV Maestro to break down and not function correctly.

4.1 Key/Value Entries

The text files contain all language dependent text in the form of "key/value" pairs. Each key/value pair looks like the example below:

`<KEY>=<VALUE>`

"`<KEY>`" is replaced with the name of the key and "`<VALUE>`" with the language specific value associated with this key. For example:

`app.caption.cancel=Cancel`

This example defines a key called "`app.caption.cancel`", which has a value of "`Cancel`".

Whenever a language dependent text needs to appear in the user interface, the program simply contains the key for that text and a replacement directive. As a result, when LISTSERV Maestro

needs to display a page containing this text, it finds the directive: “Display the text associated with the key XYZ”. LISTSERV Maestro then looks up the key in the data from the `*.properties` text files and displays the value associated with this key.

Simply by changing the value of a key, you can change whatever LISTSERV Maestro displays. LISTSERV Maestro can then display any desired text. It is important, however, that a thorough and thoughtful translation, wisely changing the key values, will preserve the meaning and functionality of the program. In order to change a key correctly, the following two rules apply:

1. Do not change the name of a key. Whatever is on the “left side” of the equals sign must be left unchanged.
2. Do not remove any keys from any of the `*.properties` files.

Any whitespace right after the equals sign is ignored. For example, if you include spaces like this: “`key=  value`”, then the actual value will be “`value`” and not “ `value`”, as you might expect. If you actually need to include spaces right after the equals sign, before the first non-space character of the value, but do not want these spaces to be treated as ignored whitespace, you must type these spaces in the form of “escaped spaces” (see section 4.3 Special Characters for more information).

4.1.1 Continuation Lines

In general, a key/value pair occupies only a single line, meaning that a value ends with the line break that delimits the line (excluding the line break itself from the value). Since there is no limit to how long a line may be, this does not impose any restrictions of how long a value may be.

Important Note: Do not confuse a “real” line break (inserted when you press the “enter” key in the editor) with a “visual” line break that appears, when your editor “wraps” lines that are too long to fit into the editor window. To help prevent confusing “real” line breaks with those that are only added by the editor to wrap a line that is too long, switch off word-wrap in your editor when editing the `*.properties` text files.

With word wrapping turned off, the lines will often be very long and you might have to do a lot of scrolling when editing these long lines. For that reason, LISTSERV Maestro uses the feature of “continuation lines”. A continuation line is a line with text that continues the “value” defined in the previous line. A continuation line is marked by including a backslash “`\`” as the very last character of the preceding line (the line that is being continued by the following line).

An example of using continuation lines appears below:

```
key=This is a value that \
goes over several lines, \
ending here.
```

The example above defines a single key/value pair, where the key is named “`key`” and the value is: “`This is a value that goes over several lines, ending here.`” The backslash characters at the very end of the first two lines signal that the value is continued in the next line. In the third line, there is no such “`\`”, since this line is the last line of the value, and it is not continued in the next line.

There must not be any spaces, tabs, or other whitespace (except for the line break) after the "\", otherwise the backslash would lose its meaning as a signal that the following line is a continuation of the current line. The backslash character itself, and the line break that follows it appear solely in the `*.properties` text file. They are not part of the actual value, but are only inserted into the text file to make the text file more readable.

To enhance readability further, all whitespace (including spaces and tabs) at the beginning of any continuation line is ignored. The whitespace will not become part of the value (just like whitespace right after the equals sign). Applying this rule, the following example defines exactly the same key/value pair as the previous example:

```
key=   This is a value that \
    goes over several lines, \
    ending here.
```

Since leading whitespace is ignored in continuation lines, you must include any whitespace that you actually want to have, at the end of the previous line. In the examples above, this is visible with the spaces after "`that`" and "`lines,`". If these spaces were not included, like the next example:

```
key=This is a value that\
    goes over several lines,\
    ending here.
```

The resulting value of "`key`", would be "`This is a value thatgoes over several lines,ending here.`". There would not be a space between "`that`" and "`goes`" or "`lines,`" and "`ending`", making the sentence appear incorrect. Take care to include all whitespace at the end of the previous line, just before the "\". Optionally, you could type any spaces at the beginning of a continuation line that you do not want to be ignored in form of "escaped spaces" (see section 4.3 Special Characters for more information).

Continuation lines, and the feature that leading whitespace in continuation lines is ignored and does not become part of the value, are frequently employed in the English language text in the `*.properties` files found in the original, un-translated translation kits, to make the text files more readable. When translating keys with continuation lines, take care to translate all lines that belong to a key (those that are continuations of the previous lines) and to use backslash "\" characters at the end of your lines to signal that the line's value continues in the next line.

4.2 Whitespace

"Whitespace" characters are characters that do not have a printed representation, but that appear as "space" between words or lines. Most commonly used are normal space characters, tab characters, and line breaks. (In Windows, line breaks usually consist of two characters per line break: one "carriage return" character followed by one "line feed" character. Carriage return and line feed are both considered "whitespace" characters.) The concept of "whitespace" characters is frequently used in this document to describe these types of characters.

4.3 Special Characters

Some characters are “special” characters, meaning that they cannot be easily typed or that they have a special meaning in the program. These characters are:

Space

Tab

Line feed

Carriage return

Backslash

If inserted into the text file, these characters have special meanings. Therefore, if you need to use them without their special meanings, you have to “escape” them by preceding them with a backslash “`\`”.

The following sections describe how to “escape” each of these characters in more detail, but here is an example of the concept:

```
key=\ \ \ \ \tFirst Line\r\nSecond Line\\with backslash\r\n
```

This example defines a key with the name “`key`” and a value that:

Starts with four non-whitespace spaces

Is followed by a non-whitespace tab

Is followed with the text “`First Line`” that ends with a Windows-style carriage-return/line-feed line break

Is followed by a second line with the text “`Second Line\with backslash`” that contains a backslash between the words “`Line`” and “`with`”

And ends with a Windows-style line break

4.3.1 Space

Normally, when you press the “space” key, a space character will be inserted into the text, moving the cursor (and all subsequent text) one position to the right. Such a “normal” space is considered whitespace and may therefore be ignored in some cases where whitespace is ignored (at the beginning of a value or of a continuation line, for example). If you want to include a space-character that is not interpreted as whitespace in the `*.properties` text file, then you would have to “escape” it with a backslash:

“`\ `” (Quotation marks added for readability)

4.3.2 Tab

Normally, when you press the “tab” key, a tab character will be inserted into the text, moving the cursor (and all subsequent text) to the next tab stop position. Such a “normal” tab is considered whitespace and is therefore ignored in cases where whitespace is ignored. If you want to include a tab-character which is not interpreted as a tab-feed to the next tab stop in the `*.properties` text file, but that instead appears as a tab-feed on the actual page where the value is used, then you would type the following escaped character (which actually consists of two characters) instead:

```
\t
```

4.3.3 Carriage and Return Line Feed

Normally (on Windows), when you press the “enter” key, a carriage return character followed by a line feed character is inserted into the text, moving the cursor (and all subsequent text) to the start of the next line (on Unix, Linux, or Macintosh, only a line feed character is inserted). Such a “normal” line break is considered whitespace. It marks the end of the current value, or, if there is a “`\`” as the last character of the line that is ended by the line break, it is ignored and the next line is interpreted as a continuation line. If you want to include a line feed or carriage return character that is not interpreted as a line break in the `*.properties` text file, but that appears as a carriage return / line feed on the actual page where the value is used, then you would type one of (or both) of the following escaped characters (which actually consist of two characters each) instead:

For carriage return: `\r`

For line feed: `\n`

For line break: `\r\n`

4.3.4 Backslash

As you have seen in the previous sections, the backslash “`\`” is frequently used to “escape” special characters, as well as to mark continuation lines. Therefore, if you actually want to include a normal backslash in the text, then you have to escape the backslash with another backslash and type:

```
\\
```

4.3.5 Special Unicode Characters

If you need to include a character that you can either not type on your keyboard, or for which there is no representation in the charset that you have chosen to store your `*.properties` files with, then you can still include this character, if you know its international Unicode value, expressed in hexadecimal form. Simply type:

```
\uXXXX
```

Replace “`XXXX`” with the four-digit hexadecimal value of the desired character. The entire 6-characters sequence you type will appear in the user interface as a single character, corresponding to the Unicode value. For example, if you type “`\u03B1`” a Greek alpha “α” will be displayed. Alternatively, if you type “`\u304A`” a Chinese Hiragana character will be displayed (if the browser where the page is viewed has support for Chinese characters installed). For information about the Unicode values of characters, please see http://www.unicode.org.

4.4 Comment Lines

All lines in one of the `*.properties` files that contain only whitespace, or that have a "`!`" or "`#`" as the first non-whitespace character are considered to be comment lines and are ignored. You will frequently find empty lines and even a few comment lines in the `*.properties` files in the translation kits. You can leave them as they are, remove them, or change them, and add your own empty lines or comment lines. Be aware that you cannot put comment lines in the middle of a continuation (see Section 4.1.1 for information on continuation lines).

4.5 HTML Code

LISTSERV Maestro uses a Web based user interface consisting of HTML pages that are displayed in a Web browser. The largest part of the language dependent text is displayed as part of HTML pages, and therefore must follow HTML coding conventions.

4.5.1 HTML Escape Sequences

In HTML, some characters have special meanings. For example the "<" and ">" characters are used to enclosed special HTML tags used to format the HTML page. Such an HTML tag may look like this:

```
<body>
```

The appearance of a word enclosed in "<" and ">" signals to the browser that this word is not supposed to appear as a visible word, but that instead is a special HTML tag that conveys some special meaning to the browser, for example how to render and format the text. As a consequence, if you want a "<" or ">" character to appear on the page, then you need to "escape" this character in form of an HTML escape sequence. Another typical character that needs to be escaped in the form of a HTML escape sequence is the ampersand character "`&`". Below are a few of the most common character escape sequences:

Character	HTML Escape Sequence
<	`<`
>	`>`
&	`&`
"	`&quot;`
A non-breaking, non-whitespace space	` `

HTML escape sequences always start with an (un-escaped) ampersand and end with a semicolon. Please see the many related resources in the Web, like this one: http://www.w3.org/TR/html401/sgml/entities.html that describes how to escape HTML.

Important note: Do not confuse HTML escape sequences with the escaping of special characters in the `*.properties` text files as described in earlier sections. These are two different types of escapes, but they both happen to apply to the same text - you must employ both of them where applicable. When LISTSERV Maestro interprets these different levels of escapes, “normal” “escaped characters” are resolved first, then HTML escape sequences are resolved.

For example: If you wanted the text “`<\>`” to appear on a page, you would have to write the key/value pair like this:

```
key=&lt;\\&gt;
```

LISTSERV Maestro would first resolve the escaped character “`\\`” to a single backslash. The result would be “`<\>`”. Next, the HTML escape sequences “`<`” and “`>`” would be replaced, and the resulting text of “`<\>`” would be displayed on the screen.

4.5.2 HTML Whitespace and Line breaks

Whitespace in HTML is interpreted specially. Any sequences of whitespace characters (space, tab, line breaks, and so on) are rendered as a single space by the Web browser. For example:

```
key=word 1  \t\t\r\n\t\t\t  \r\nword 2
```

This defines a value that:

- Starts with “`word 1`”
- Is followed by two spaces
- Then is followed by two tabs
- Is followed by a Windows-style line break
- The is followed by three more tabs
- Is followed by another two spaces
- The is followed by another line break
- And finally ends with “`word 2`”.

All the characters between “`word 1`” and “`word 2`” - all the spaces, tabs and line breaks - are considered whitespace. This means that they will all be collated into a single space when rendered as part of a HTML page and will display on the screen as: “`word 1 word 2`”. There will not be several spaces but only one. There will not be any visible tab or line breaks whatsoever.

To include more than one space, you need to employ “special” spaces that are written in the form of their HTML escape sequence. Widely used is “` `” for a non-breaking space. The `key=word 1   word 2` defines a value “`word 1   word 2`” where the three spaces between the two words will actually appear on the rendered page (and, since they are non-breaking spaces, the browser will also not introduce a line break at any of these spaces).

Line breaks are usually introduced by the browser that renders the page whenever a line break is required. That is whenever the line length is not enough to show the rest of the text, the text will be wrapped to the next line. The browser will use whitespace between words as the places where a possible line break may be inserted for this wrapping.

Line breaks that are visible on the rendered page cannot be inserted by using the "`\r`" or "`\n`" characters in a value. Instead, you need to use a special HTML tag that has the effect of introducing a visible line break:

```
<br>
```

For example:

The following key/value pair

```
key=line 1<br>line 2
```

Will actually appear as two lines "`line 1`" and "`line 2`" on the rendered page, although for better readability in the `*.properties` file, you might want to write this key/value pair with a line break and a continuation line even in the `*.properties` file, so that it becomes more apparent that there are two lines in this value:

```
key=line 1<br>\
    line
```

4.5.3 HTML Formatting in Text Resources

The largest part of all text resources in the `*.properties` files are pure text strings, without any formatting directives (in form of HTML tags) in them. All formatting is done on the pages themselves, into which the texts are inserted. However, some text resources do contain actual HTML tags that do some sort of formatting that cannot easily be imposed from the outside.

For example, if you have a single word in the middle of a sentence that you want to emphasize by rendering it in a bold font, you have to include the corresponding bold-tag right into the text of the sentence itself, bracketing the word you want to emphasize.

If you find an HTML tag (recognizable as being enclosed in "<" and ">") in a text, then you must see to it that the same tag, with the same semantics, appears even in your translated version of the text. Most commonly used are span-tags "`<span>`". A span-tag actually consists of an opening part "`<span>`" and a closing part "`</span>`". The two parts bracket one or several words in the text that special formatting is applied to, for example an emphasis. The type of formatting is then included in the opening span-part, in form of a class attribute: "`<span class="emphasis">`".

For example:

```
key=Do you <span class="emphasis">really</span> want to delete this?
```

This key defines a text with a question to the user, where the word "really" is to be emphasized, and therefore it is bracketed with the opening and closing parts of the span-tag, using the "emphasis" class. Translate this text to a corresponding meaning in your target language, and employ the opening and closing parts of the span-tag (with the "emphasis" class) to the word that comes closest to the same meaning as the word "really" in the sentence above.

Sample for Swedish:

```
key=Skall detta <span class="emphasis">verkligen</span> blir raderat?
```

Sample for German:

```
key=Wollen Sie dies <span class="emphasis">wirklich</span> löschen?
```

The span-tag is frequently used throughout the text resource of LISTSERV Maestro, so be careful to transfer them with a similar meaning to your translation, using the same “class” attribute. Other tags that are sometimes used are:

`<ul> ... </ul>`

`<li>`

`<p> ... </p>`

(This list may not be complete.)

If you find any HTML tags, be careful to use the same tags, with the same meaning, in your translated version.

4.5.4 Non-HTML Exceptions

Even though the bulk of all texts are used inside of HTML pages, there are some exceptions. Inside of these exceptions, the rules of HTML escape sequences, HTML whitespace and line breaks, and HTML tags, do not apply and you must be careful not to use a HTML escape sequence in one of these texts. If you do so, and, for example, write “`&amp;`” instead of “`&`”, then the user would actually see the text “&” instead of the desired “&”.

These exceptions are:

- All texts appearing in alert or confirmation dialog boxes that pop up from the browser (they are not displayed on the page but in a Windows popup window).

  You can recognize these texts by looking at the names of the keys. All texts that appear in alert or confirmation dialog boxes have a key name of:

  `PART1.alert.PART2` or

  `PART1.query.PART2`

  Where “`PART1`” before the “`.`” and “`PART2`” after the “`.`” may be any text (but not empty). For any key that has a name with “`.alert.`” or “`.query.`” in the middle, the special HTML escaping and whitespace rules do not apply.

- All texts that are written to downloaded files, for example downloaded text files containing report results. There are only a few `*.properties` files that contain resource texts that are used for downloaded text files. In these files, all such key/value pairs (or blocks of pairs) are marked with #-comments as texts “for download” or “for downloaded files”. If you see this comment, do not apply the special HTML rules on the texts marked by the comment.

  For the 1.2 version, such texts are found in:

  `com\lsoft\lui\res\de\reports\reportDetails.properties`

  `com\lsoft\lui\res\de\reports\reportDistribution.properties`

  `com\lsoft\lui\res\de\reports\reportRaw.properties`

com\lsoft\lui\res\de\reports\reportSum.properties

("de" would be replaced with the language code of your target language).

4.6 Word Replacement

In some resource texts, LISTSERV Maestro employs word replacement to ease translation. Consider the following example:

"500 duplicates have been removed from the job".

Obviously, this text contains a value that may vary at run time. Therefore, you cannot simply have a key/value pair with this text in the *.properties files, or otherwise you would have to have an infinite number of texts, one for each number of duplicates that has been removed.

The immediate solution is to simply have the text "duplicates have been removed from the job" as the value, and prepend the number of deleted duplicates at runtime to this text. This solution means that it would be hard coded, and that the number of deleted duplicates appears first in the sentence, followed by some text. What if a language's syntax did not allow such a sentence? For example, the syntax of that language might only allow a sentence like "From the job, 500 duplicates have been removed", where the variable part appears right in the middle of the sentence. To avoid this "hard coded" solution, LISTSERV Maestro uses word replacement instead.

4.6.1 Replacement Placeholders

In the text resources, wherever there is a variable, a replacement placeholder is used. At runtime, the placeholder is replaced with the variable value, forming the desired finished sentence. Placeholders are numbered entities, starting with "0". In the text, they appear in the form of their number, enclosed in curly brackets, like this: "{0}", "{1}", "{2}", and so on. If there is only a single placeholder in the text, only "{0}" will appear, if there are two, both "{0}" and "{1}" will appear. There will never be a placeholder "{1}" without a placeholder "{0}". There will never be a "{2}" placeholder without its companions "{1}" and "{0}", and so on.

From the previous example sentence, the following key/value pair would be written like this:

```
key={0} duplicates have been removed from the job
```

When at runtime, the "{0}" is replaced by the variable number "500", and the correct English sentence "500 duplicates have been removed from the job" will be formed. If another language requires the variable to appear in another position, simply move the placeholder forming a key/value pair like this for example:

```
Key=From the job, {0} duplicates have been removed
```

Here too, after replacement of "{0}" by the variable value "500" (or whatever value), the correct sentence would be formed.

Whenever you find one of these numbered placeholders, recognizable by their curly brackets in the text string, you must make sure to:

- Include exactly the same number of placeholders in your translated version, with exactly the same number-indexes (always starting with "{0}").
- Determine the meaning of each placeholder (what will it be replaced with), so that you can use the same placeholder with the same meaning in your translated version.

- Use the correct order of placeholders, which may vary from the order in the original text (see next section).

4.6.2 Placeholder Order

Replacement placeholders are always a number starting with "`{0}`". However, this does not mean that the first placeholder in a text (when read from left to right) has to be numbered "`{0}`" and that the following variable has to be "`{1}`". In most cases, this may be so, but it is not a general rule that needs to be followed. The two examples below are both valid expressions:

```
This {1} contains {0} two placeholders
```

```
This {0} contains {1} two placehodlers
```

More important than the actual numbers (0, 1, 2), is that the semantically correct placeholder is used at each position. Consider the following sentence: "250 recipients have opened mail XYZ". This sentence contains two variable values, the number of recipients that have opened the mail ("250"), and the ID of the mail ("XYZ"). Another language's syntax may require this sentence to be written as "The mail XYZ was opened by 250 recipients". Since we are aware of this, we write the original key/value pair with placeholders:

```
key={0} recipients have opened mail {1}
```

This means that "`{0}`" will be replaced with the number of recipients and "`{1}`" will be replaced with the ID of the mail. Therefore, in the "other language" version, we would have to make sure to use the correct placeholders in the right positions:

```
key=The mail {1} was opened by {0} recipients
```

This effectively reverses the order of the two placeholders (when read from left to right), resulting in the desired sentence. If the order of the placeholders was not reversed, and the text "`The mail {0} was openend by {1} recipients`" was used, a nonsense sentence like "The mail 250 was opened by XYZ recipients" would result.

Whenever there are more than one replacement placeholder in the same key/value pair, make extra sure you understand the meaning of each placeholder (what it will be replaced with), so that you can then use each placeholder in the correct position in your translated text.

4.6.3 Special Characters

Similar to the special characters in the text and in HTML that need to be escaped to actually use on the screen, the curly brackets used in the replacement placeholders are special characters that need to be escaped if they are to actually be displayed on the screen. To escape the curly brackets, place them between single quotation marks like this:

Type `'{'` to get `{`

Type `'}'` to get `}`

Now the single quotation marks have a special meaning too - to quote curly brackets.

Type `''` (two single quotation marks) to get `'` (an individual single quotation mark).

Single quotation marks that are not escaped by "doubling" them will simply not appear when displayed in the user interface.

This also applies to the genitive apostrophe. The value

```
key=Change the job's name
```

Appears as “Change the jobs name” in the user interface, since the single quotation mark used as the apostrophe was not doubled. To get the desired text of “Change the job’s name” you would have to write:

```
key=Change the job''s name
```

4.7 Escape Character Summary

With LISTSERV Maestro’s `*.properties` text resource files, there are three different levels of escapement, which are resolved in the following order:

Special “escaped characters” (escaped with backslash) are resolved

Text replacements are performed and quoted curly brackets and single quotes are un-quoted

HTML escape sequences are turned back into their actual character representation

For example:

```
key=The job''s size {0} is &lt; the '{'threshold'}' \\ margin\
    that is allowed\r\nPlease resubmit.
```

First, the “escaped characters” are resolved. In this example, there are an escaped backslash, a line continuation and a Windows-style line break. After the first step, we get:

The job''s size {0} is < the '{'threshold'}' \ margin that is allowed

Please resubmit.

Next, text replacements are performed and single-quotes are un-quoted. We find a doubled single quote and two quoted curly brackets (one opening, one closing), and one replacement placeholder. Assuming the placeholder’s value is 100, we would get:

The job’s size 100 is < the {threshold} \ margin that is allowed

Please resubmit.

Finally, HTML escape sequences are resolved, so the final text appears as:

The job’s size 100 is < the {threshold} \ margin that is allowed

Please resubmit.

4.8 Date and Time Formats

Date formats, the order in which day, month, and year appear when printed as a date, differ from country to country. While Americans usually write “`month/day/year`”, the Swedes write “`year-month-day`”, and Germans use “`day.month.year`”. LISTSERV Maestro will allow you to match the date format to the target language you are translating, but there are some special date format keys that you need to take into special consideration:

4.8.1 Output Format for Date

Whenever LISTSERV Maestro outputs dates, for example the send-date of a job, you have two attributes of the output format that you can influence:

- The name of the month:

In the file "com\lsoft\lui\res\de\date.properties", edit the twelve keys called "date.month.1" to "date.month.12". Assign values to these twelve keys that match the name of the months in your target language (starting with January), preferably in an abbreviated textual form. If there is no good abbreviated textual form in your target language, and the full textual form appears as too long, you may use numerical values 1 to 12 instead.

- Print month or day first:

  In the file "com\lsoft\lui\res\de\date.properties", edit the key called "date.printMonthFirst". Set the value to "true" to get an output date format of the type "Month, Day, Year" (for example "American" - "Aug. 12, 2003") or to "false" to get a format of the type "Day, Month, Year" (for example "European Style", - "12 Aug. 2003").

4.8.2 Output Format for Time

Whenever LISTSERV Maestro outputs times, for example the send-time of a job, you have one attribute of the output format that you can influence:

Use AM/PM format or not:

In the file "com\lsoft\lui\res\de\date.properties", edit the key called "date.useAM/PM". Set the value to "true" if you want to use a 12 hour format with AM/PM or to "false" if you want to use a 24 hour format (military time).

4.8.3 Report Download Format for Date and Time

Whenever LISTSERV Maestro prepares report results for download, in form of a ZIP file, it puts a "readme.txt" text file into that ZIP file with some information about the downloaded report. This information also contains some date and time information, for example about when the report was executed. You can modify the appearance of this date/time format by editing the following four keys (one for each report type):

- Details Report:

  In the file "com\lsoft\lui\res\de\reports\reportDetails.properties", edit the key "reports/reportDetails.msg.downloadDateFormat".

- Distribution Report:

  In the file "com\lsoft\lui\res\de\reports\reportDistribution.properties", edit the key "reports/reportDistribution.msg.downloadDateFormat".

- Raw Events Report:

  In the file "com\lsoft\lui\res\de\reports\reportRaw.properties", edit the key "reports/reportRaw.msg.downloadDateFormat".

- Sum Report:

In the file "com\lsoft\lui\res\de\reports\reportSum.properties", edit the key "reports/reportSum.msg.downloadDateFormat".

Any of these four "reports/XXX.msg.downloadDateFormat" keys must contain a pattern that will be translated into an actual date and time string at runtime. In this pattern, the following pattern fields must appear, but you may move them around, change their order, and insert other fill characters to create a format that suits you. Note that uppercase and lowercase are important, and that all pattern fields will be replaced with numerical counterparts (no textual names of months and so on).

- MM: Two uppercase M-characters: Will be replaced with the month of the year 0-12.
- dd: Two lowercase d-characters: Will be replaced with the day of the month 01-31.
- yyyy: Four lowercase y-characters: Will be replaced with the 4-digit value of the year.
- HH: Two uppercase H-characters: Will be replaced with the hour of the day 00-23.
- mm: Two lowercase m-characters: Will be replaced with the minute of the hour 00-59.

Examples: 28th of August 2002, 5 minutes after 3, p.m., will be formatted as follows:

MM/dd/yyyy HH:mm	08/28/2002 15:05
yyyy-MM-dd HH:mm	2002-28-08 15:05
HH:mm [dd.MM.yyyy]	15:05 [28.08.2002]

4.8.4 Input Format for Date and Time

Whenever the user needs to input a date and time value for LISTSERV® Maestro to process, the user must follow a certain pattern (that is also displayed to the user). You can influence this pattern:

- Date pattern:
  In the file "`com\lsoft\lui\res\de\date.properties`", edit the key called "`date.dateFormat`". Set the value to a pattern string that contains exactly the three pattern fields "`MM`", "`dd`" and "`yyyy`" (using the exact case displayed). See above for details of these three fields and examples. You can order them in any way you like and include any fill characters you like, but you must be aware that the user must enter them in exactly the same order with the same fill characters, so that LISTSERV Maestro can understand them. Choose an order and fill characters that are most common in the locale matching your target language.

- Time pattern:
  In the file "`com\lsoft\lui\res\de\date.properties`", edit the key called "`date.timeFormat`". Set the value to a pattern string that contains exactly the two pattern fields "`HH`" and "`mm`" (using the exact case displayed). See above for details of these two fields and examples. Again, you can order them in any way you like and include any fill characters you like, but keep in mind that the user must input them in the same way.

Section 5 Translating the Database Plugin Resources

LISTSERV Maestro uses a plugin mechanism to be able to connect to as many different kinds of databases as possible, both to store the internal data and to retrieve recipient data. To be able to connect to a database, the system needs to query a number of values from the user, such as the name of the server where the database is running, the user name, and the password to access the database.

These values are usually different from database to database, as well as from plugin to plugin. Therefore, it is not possible to provide standardized text resources (for example to label the various input fields required to input the parameters), but instead, each plugin must provide its own resources for display. Naturally, if you translate the Maestro User Interface (and possibly the Administration Hub user interface), you will also want to translate these texts, otherwise the users will unexpectedly see English language (=default language) text mixed with your target language.

The plugins are independent from the rest of the Maestro User Interface, so they need to be translated independently. The mechanism to translate the plugin resources is quite similar to that of translating the main user interface (you have to translate "values" from key/value pairs, which are stored in `*.properties` text files), however, there is a slightly difference:

For the main user interface, you have one translation kit provided for each target language. For the database plugins, there is only a single translation kit, which can be adapted for any target language, and this kit contains resources for all available plugins. The database plugin translation kit is provided in the form of a ZIP file like "`Maestro-1.2-DatabaseTransKit.zip`". Similar to the other translation kit files, this ZIP file contains an internal folder structure that must be preserved both during unzipping (to prepare for translation) and during zipping (after translation). It also contains one `*.properties` text file for each

available plugin. Most of them are located in their own subfolders inside of the ZIP file, but some files also may share a subfolder.

At the current time, this ZIP file has the following contents:

`com\lsoft\lui\db\ibm\DB2V72DriverResources_xx.properties`

`com\lsoft\lui\db\mysql\MySQLDriverResources_xx.properties`

`com\lsoft\lui\db\oracle\Oracle8iThinDriverResources_xx.properties`

`com\lsoft\lui\db\sqlserver\MSSQLDriverResources_xx.properties`

`com\lsoft\lui\db\sqlserver\SPRINTADriverResources_xx.properties`

In the future, additional files may be added as additional plugins become available.

Each of these properties files stands for one of the available plugins (recognizable by their names and the subfolders they are in). However, the files are still language-independent - they not prepared for any special language yet.

To translate a plugin resource file, you first have to rename it so that it matches your target language. Rename it so that what is currently the two-letter placeholder "`xx`", matches the two-letter ISO 639 language code for your target language. For example "`de`" for "German", "`fr`" for "French", "`es`" for "Spanish" and "`sv`" for Swedish. Language codes must be lower-case. A list of language codes is available at: http://userpage.chemie.fu-berlin.de/diverse/doc/ISO_639.html.

For example, a French translation would rename the files as follows:

`com\lsoft\lui\db\ibm\DB2V72DriverResources_`**`fr`**`.properties`

`com\lsoft\lui\db\mysql\MySQLDriverResources_`**`fr`**`.properties`

`com\lsoft\lui\db\oracle\Oracle8iThinDriverResources_fr.properties`

`com\lsoft\lui\db\sqlserver\MSSQLDriverResources_`**`fr`**`.properties`

`com\lsoft\lui\db\sqlserver\SPRINTADriverResources_`**`fr`**`.properties`

(Bold emphasis added to make the example clearer.)

After you have renamed the files, simply edit them just like the other `*.properties` text files from the main interface resources; the same rules and concepts apply. Each `*.properties` file must be saved in text format, using the Latin 1 (ISO-8859-1) charset (this is the default, see below about how to use a different charset). After you have translated the files, zip them up again into a new ZIP file, with a meaningful name (taking care to preserve the same folder structure inside of the ZIP file).

Do not rename a file or folder in any way except for the "`xx`" part.

Here are a few other "short cuts" you can use:

If you only need a certain database plugin:

If you only want to use a certain database plugin that matches your database, but have no interest in the other plugins, then you only need to translate the file or files for the plugin(s). In that case, simply delete all the other files (and, if after deletion a folder is empty, delete that folder too). Your zipped-up archive only needs to contain those files that you have actually translated, all others should be left out.

If you want to translate to more than one target language:

If you have more than one target language to translate to, simply make as many copies of the files you want to translate as you have target languages. Rename each one of them so that there is one for each target language. For example, if you only want to use the DB2 and Oracle plugins, and want to translate them to German, French and Swedish, you would start with the following files:

`com\lsoft\lui\db\ibm\DB2V72DriverResources_de.properties`

`com\lsoft\lui\db\ibm\DB2V72DriverResources_fr.properties`

`com\lsoft\lui\db\ibm\DB2V72DriverResources_sv.properties`

`com\lsoft\lui\db\oracle\Oracle8iThinDriverResources_de.properties`

`com\lsoft\lui\db\oracle\Oracle8iThinDriverResources_fr.properties`

`com\lsoft\lui\db\oracle\Oracle8iThinDriverResources_sv.properties`

The three IBM files are copies of each other (only with different names) as are the three Oracle files. Then you translate each of these six files independently and zip them up (leaving out the files from the plugins you do not want to use, as explained above).

If you need to use a text-file encoding other than Latin 1 (ISO-8859-1):

Each `*.properties` text files is, by default, assumed to be encoded with the Latin 1 (ISO-8859-1) charset. If your target language requires a different charset (for example ISO-8859-7 for Greek), take the following steps:

1. Create a second text file (that must be stored either using US-ASCII or ISO-8859-1) with the name "`BASENAME_encoding_xx.dat`",

   A. Replace "`BASENAME`" with the same plugin name the original `*.properties` file uses (for example "`DB2V72DriverPlugin`" for the IBM plugin).

   B. Replace "`xx`" with the matching two-letter, lower-case language code of your target language.

2. Save this file in the same subfolder as the original `*.properties` text file. In that "`BASENAME_encoding_xx.dat`" file there must be a single key/value pair with a key named "`encoding`" and as the value the name of the charset encoding that was used to store the text file with the translated texts (see section 3.2.1 for details).

If you want to translate the IBM and Oracle plugins, for example, but this time for Russian, using the ISO-8859-5 charset for Cyrillic, then you would copy, rename and create files so that you have the following set of files to start with:

`com\lsoft\lui\db\ibm\DB2V72DriverResources_ru.properties`

`com\lsoft\lui\db\ibm\DB2V72DriverResources_encoding_ru.dat`

`com\lsoft\lui\db\oracle\Oracle8iThinDriverResources_ru.properties`

`com\lsoft\lui\db\oracle\Oracle8iThinDriverResources_encoding_ru.dat`

The two "`*_encoding_ru.dat`" files would have a single entry like "`encoding=ISO8859-5`" each, and you would edit and translate the two "`*.properties`" files to your Cyrillic/Russian texts and store them using the ISO-8859-5 charset.

This page intentionally left blank