Loway
research

QueueMetrics
call center monitor

Translating QueueMetrics into a new language

Translator's manual

AUTORE:	LOWAY RESEARCH
VERSIONE:	1.3
DATA:	NOV 11, 2006
STATO:	

Loway Research di Lorenzo Emilitri
Via Fermi 5 – 21100 Varese – Tel 0332 320550 – Fax 0332 328609 – p.i. 02888500127

Index

Document contents 3
Revision history 3
Translating QueueMetrics 3
The locale 3
Setting the correct editor encoding 4
Translating the language file 4
Translating Decoding Tables 5
Testing your work 6
Licence 7
Appendix I: recoding ISO-8859_1 data 7

Document contents

This document explains how to translate QueueMetrics into a different language or for a foreign country. After version 1.3, QueueMetrics offers full internationalization primitives, so translating QueueMetrics may take less than a couple of hours.

Revision history

- Sep 18, 2006: first draft
- Sep 22, 2006: added language name
- Sep 28, 2006: added docs on QueueMetrics 1.3 alpha
- Nov 11, 2006: QM now uses UTF-8 as the basic encoding.

Translating QueueMetrics

To translate QueueMetrics, you need to translate two different kinds of documents:

- The language file
- All relevant Decoding Tables

The language file is where most of the strings found in QueueMetrics can be found. It is named something like queuemetrics_en_US.properties and can be found under QueueMetrics/WEB-INF/LIntl. Each language has a separate language file.

The Decoding Tables are used to store multiple values and expanded strings, and will be found under QueueMetrics/WEB-INF/LDec. Each language has a separate folder containing its set of Decoding Tables.

No tools are needed to translate QueueMetrics but a modern text editor and a little patience.

The locale

QueueMetrics translations work by applying a “locale” to QueueMetrics, i.e. a set of conventions known for a certain language as spoken in a certain country. A locale must be used that is compatible with the Java language; if in doubt, ask Loway Research first.

Valid examples of locales are:

- en_US: English as spoken in the USA
- it_IT: Italian for Italy
- es_ES: Spanish for Spain
- fr_FR: French for France

- de_DE: German for Germany
- pt_PT: Portuguese for Portugal and Brazil

A complete list of possible locales is available at:
http://java.sun.com/j2se/1.4.2/docs/guide/intl/locale.doc.html

Setting the correct editor encoding

QueueMetrics uses the encoding UTF-8 for its pages, so you should make sure that the text editor you use will save non ASCII characters as UTF-8. Failure to do so might end up making your non-ASCII characters unreadable on QueueMetrics pages, even if they look perfectly on the language files.

Most text editors today support saving text files with different encodings.

Please note that the language files require an additional step after being edited in UTF-8: they must then be converted into Java native encoded files before being used by QueueMetrics[1].

Translating the language file

The language file is a standard Java property file. For each line a placeholder is defined and then the actual text that appears for that placeholder. E.g.

```
hdr_print=Print
hdr_logoff=Log off
hdr_home=Home
hdr_answered=Answered
hdr_unanswered=Unans.
```

Empty lines are allowed in the file and lines starting with the # key are considered comments.

To translate a language file, start by making a copy of the queuemetrics_en_US.properties file to your own locale and then change the values after the "=". DO NOT modify the keys, as they are used by QueueMetrics to find the correct string.

It is possible that some values are HTML or contain HTML entities; just change the English text to your own language.

[1] Feel free to thank the original Java library developers for this additional step. Processing non-ASCII text streams in Java is for the brave and patient only!

Keys are usually grouped by QueueMetrics page; it is a good idea to translate them by looking at the page on which they will appear. Do not turn a very short sentence into a much longer one, as this may break the formatting of some table.

A few keys – placed at the top of the file - have special meaning:

- 00-LanguagePack: do not touch
- 00-Author: place your name here
- 00-Licence: place the licence type here (we suggest to type "Public domain")
- 00-LanguageName: the name of the local language, i.e. the one that appears in the drop-down combo box on the auth page.
- Dateformat_*: these values are used to format date and time values. The syntax is explained in http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html . If in doubt, leave them as they are or ask Loway Research. A wrong formatting code may break QueueMetrics.

If you use non ASCII characters in your language files (as will be the case for most non-English languages), you need a further step to encode your UTF-8 characters into a format that the Java subsystem will understand. You can accomplish this by typing the following commands on a system with a Java SDK installed:

```
mv queuemetrics_xx_XX.properties qm_xx_XX.orig
native2ascii -encoding UTF8 qm_xx_XX.orig
queuemetrics_xx_XX.properties
```

This will copy your UTF-8 file to .orig and then recreate a native properties file with the same name.

Translating Decoding Tables

To translate decoding tables, make a copy of the directory en_US and translate it to your locale. Each decoding table is made up of a key and a set of values, separated by the pipe "|" symbol.

An example is given below:

```
L|Agent is currently logged on|
X|Agent is currently logged off|
P|Agent is currently paused|
?|Agent status cannot be determined|
```

You NEVER have to change the first value of a word,, even if it contains an English word. You must change all other occurrences. Do not change the case of the keys and do not remove trailing pipes.

Translating all Decoding Tables is quite fast and will not take much. A few decoding tables are fully numeric, so no translation is needed.

Make sure that the text in your decoding tables is all encoded using UTF-8, as this is the charset used for reading. Wrongly-encoded characters will end up showing as question marks in QueueMetrics.

Testing your work

To test your work:

- Install a version of QueueMetrics whose version number is greater to or equal to 1.3.0
- Copy your language file to queuemetrics/WEB-INF/LIntl with the name queuemetrics-*xx_YY*.properties
- Copy your translated Decoding Tables to the folder queuemetrics/WEB-INF/LDec/*xx_YY*/
- Log off QueueMetrics
- Stop Tomcat
- Delete the contents of the folder /tomcat/work/Catalina (this is where user sessions are stored when Tomcat is shut down). It is VERY important that you follow this step, otherwise the Decoding Tables may not be reloaded when Tomcat starts.
- Restart Tomcat
- Go to QueueMetrics home page and select your new language from the selector combo.

If you forgot to translate a label into your new language, you will see QueueMetrics display the string **#labelname#** instead, where *labelname* is the name of the missing label.

You should check the following points for accuracy:

- Dates and decimal numbers have the correct representation for your language
- All fields appear to be filled, i.e there re no blank labels (check the “Inbound” / “outbound” setting of the “Answered calls, by direction” box)
- The length of each field is consistent with its graphic space, i.e. tables are not deformed by their headers

If you encounter any problem in the translation process, feel free to contact Loway Research for help.

Licence

We suggest that you release your translations of QueueMetrics into the Public Domain or the GNU LGPL licence, so that we can redistribute them with QueueMetrics itself. If you do so, we are going to add your name and/or your firm's on the Contributors page in QueueMetrics itself, and we believe this may be a powerful way to make yourself known to QueueMetrics users worldwide.

Appendix I: recoding ISO-8859_1 data

If your text editor only supports ISO-8859_1 (latin) charset, you can still use it to traslate QueueMetrics. This will require an additional step in order to translate the Decoder files to simple UTF-8 and the language file to native-encoded UTF-8.

The following Perl script, to be executed in Linux on a machine with an installed Java SDK, will help you in recoding ISO-8859 files to UTF8.

The following example will search for it_IT files in a directory called *src* and will copy them to a directory called *dst*. Make sure that *dst/it_IT* (or whatever your locale is) exists before running it.

```
# recode I18N files

@files = ();

push @files, mkfiles( "it_IT" );

foreach $file (sort @files) {
  print "$file....\n";
  if ( $file =~ /.properties$/ ) {
     `native2ascii -encoding ISO8859_1 src/$file dst/$file`;
  } else {
     `iconv --from-code=ISO_8859-1 --to-code=UTF-8 --output
dst/$file src/$file`;
  }
}

sub mkfiles() {
  my ($lang) = @_;
  return ( "queuemetrics_$lang.properties",
     "$lang/agentLevel.txt",
     "$lang/analisi_prefisso.txt",
     "$lang/analisi_wizard.txt",
     "$lang/combo_analisi_code.txt",
     "$lang/day_month.txt",
     "$lang/disconnect_ko.txt",
     "$lang/disconnect_ok.txt",
     "$lang/live_agent_status.txt",
```

```
      "$lang/masterkey_abilitato.txt",
      "$lang/offset_orario.txt",
      "$lang/pgag_callstatus.txt",
      "$lang/queueDirection.txt" );

}
```

If you are unsure about these recoding operations, just send us your files and tell us the codepage you used to compose them, and we will recode them for you as needed.

Please note that the name of the codepages used by *iconv* and *native2ascii* do not match, feel free to consult:

- Run *iconv –list* to see a list of supported encodings
- See http://java.sun.com/j2se/1.4.2/docs/guide/intl/encoding.doc.html for encodings supported by native2ascii