Openbravo Tutorial:

Translating Openbravo ERP

March 5th, 2007

Revision 1.4

openbravo™
opening ERP's future!

Visit us at www.openbravo.com

Table of Contents

I. Introduction

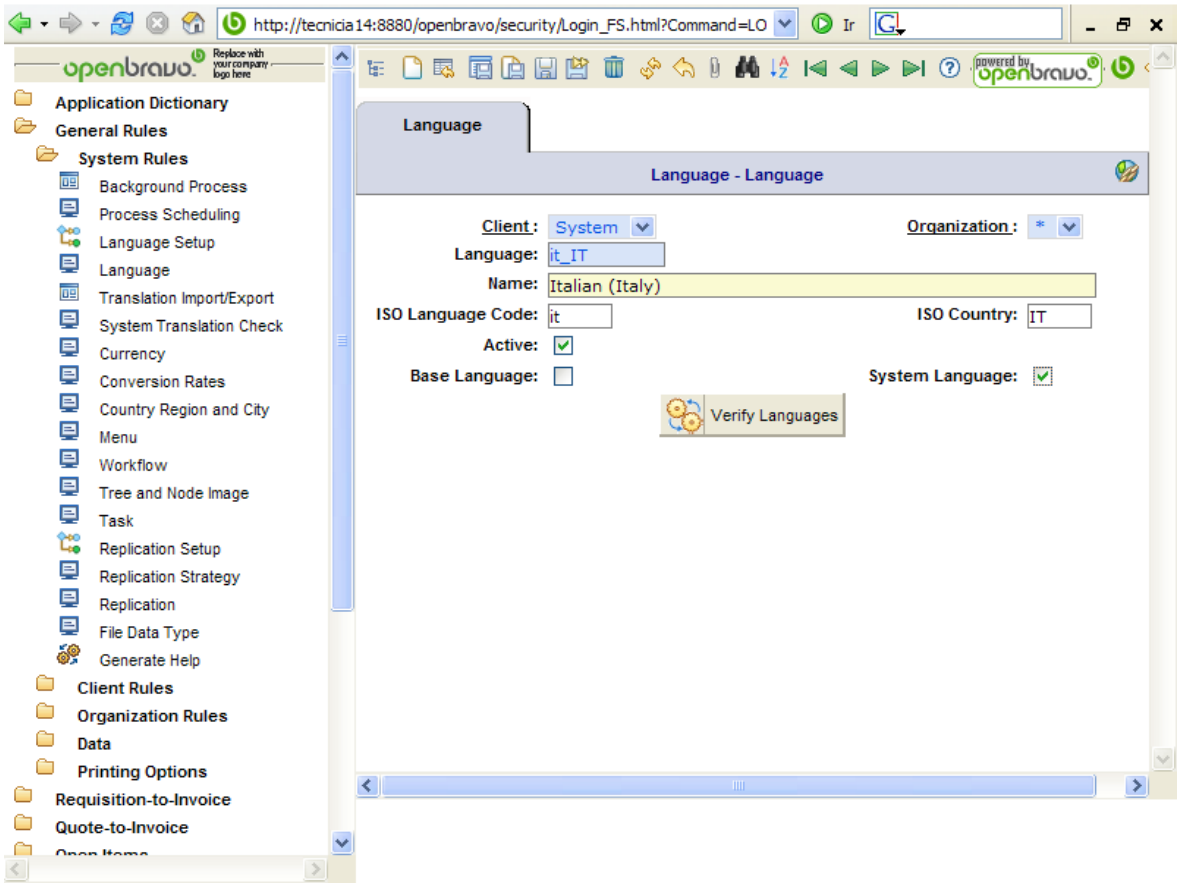
This document explains the process of translating and maintaining Openbravo in different languages.

In order accomplish this task, the user must first create and activate a language, obtaining an exact copy Openbravo ERP in a base language which can then be modified and translated to the language of choice. The user must then create a language pack for this new language, translate the language pack editing the xml files using the xml or text editor of choice, import the translated files again, and finally compile Openbravo applying all desired language modifications.

II. Creating a Language

In order to translate Openbravo, the first step is to create a language or to activate a previously defined language. Indeed, there are many languages listed so it is likely that the user will only need to activate one of them to translate.

The user must sign in to Openbravo using the *System Administrator* role located in the login area. Next, he or she can go to *General Rules* > *System Rules* > *Language*.



Here the language of choice may be selected from the list (if the user is not at relation view, he or she should click the *relation* button) or created ( ). If a new language is going to be created it should follow the Java convention of country and language: the language with 2 lower letters followed by an underscore (_) and the country as 2 capital letters (e.g. fr_CN would stand for Canadian French). By checking the *System Language* box, the user actives the language to translate and from this point, it will then appear in the login window (where the user signed in as a *System Administrator*).

The next step is to copy all text from the *Base Language* to the new

language. The base language is the language that is used as almost a starting point for translating the application, and it is English (en_US). By clicking on the *Verify Language* button. Openbravo will be confirmed as ready to be translated and for the time being, appear in the new language will exactly as it does in the base language. The translation process may now begin.

III. Language Pack

I.1 Structure

A language pack is a set of XML files containing the translations for a desired language. Each file contains the translation for a single table in the database and is given the same name as the table which contains its corresponding translation.

These language packs are stored in a server directory by language using the convention of language and country (e.g. *en_US, es_ES…*). They are inside a *lang* directory in the folder that is defined to contain all attachments. This is defined at *web.xml* file, is requested during the installation process, and by default is */AppsAttachment*. An example of a complete path for Spanish/Spain could be */AppsAttachment/lang/es_ES.*

The structure of the XML is as follows:

- A tag containing the table and the language names.
- For each row in the table there will be a *row* tag width attributes *id* for the row identification and *trl* which will be *Y* or *N* depending on whether or not it has been translated.
- Inside the *row* tag, there is a *value* tag for each column in the table. This tag includes the attributes *column* for the column name and *original* signifying the value for the column in the base language (English). The value for the tag will contain the translated text. **This is the text that you have to change if you want to make a translation.**

The following piece of code is an example for the file: */AppsAttachment/lang/es_ES/AD_TASK.xml:*

```
<?xml version="1.0" encoding="UTF-8"?>
<compiereTrl language="es_ES" table="AD_TASK">
  <row id="102" trl="Y">
    <value column="Name" original="Java Version">Versión de java</value>
    <value column="Description" original="Displays the version of the default
Java VM">Muestra la versión de la java VM por defecto</value>
    <value column="Help" original="The java version used by the application
might be different."/>
  </row>
  <row id="103" trl="N">
    <value column="Name" original="Database Export">Exportar BD</value>
    <value column="Description" original="Export (save) the database">Exportar
(guardar) la base de datos</value>
    <value column="Help" original="Run this command from the server">Run this
command from the server</value>
  </row>
</compiereTrl>
```

The complete list of xml files that compose a language pack called lg_CT, where lg stands for language and CT for country, is:

- **AD_DESKTOP_TRL_lg_CT.xml:** Translations for AD_Desktop. This table is not used at this moment.
- **AD_ELEMENT_TRL_lg_CT.xml:** Translations for the AD_Element table. This table contains all the elements of the application. They are used in order to have centrally maintained a description and help of the elements.
- **AD_FIELDGROUP_TRL_lg_CT.xml:** Translations for AD_FieldGroup. Field groups are used to group a fields in a window.
- **AD_FIELD_TRL_lg_CT.xml:** Translations for the AD_Field table. This table holds the information about the fields that are shown in each window of the application.
- **AD_FORM_TRL_lg_CT.xml:** Translations for the AD_Form table, where are defined the name, description and help for all the forms of the application. A form is a manually generated window.
- **AD_MENU_TRL_lg_CT.xml:** Translations for AD_Menu table. Here can be found the menu tree that appears on the left side of the application.
- **AD_MESSAGE_TRL_lg_CT.xml:** Translations for AD_Message table. This table defines all the messages that application displays.
- **AD_PRINTFORMATCTEM_TRL_lg_CT.xml:** Translations for AD_PrintFormatCTEM. This table is not used at this moment.
- **AD_PRINTLABELLINE_TRL_lg_CT.xml:** Translations for AD_PrintLabelLine. This table is not used at this moment.
- **AD_PROCESS_PARA_TRL_lg_CT.xml:** Translations for the AD_Process_Para table. AD_Process
- **AD_PROCESS_TRL_lg_CT.xml:** Translations for AD_Process table. Here appears the name, description and help for the processes invoked from the application.
- **AD_REFERENCE_TRL_lg_CT.xml:** Translations for AD_Reference table. In this table are all the references. References are used to define data types, lists of values or tables.
- **AD_REF_LIST_TRL_lg_CT.xml:** Translations for AD_Ref_List. In this table are found the values for the references of list type.
- **AD_TAB_TRL_lg_CT.xml:** Translations for AD_Tab table. This table contains the name, description and help for all the tabs in the application.
- **AD_TASK_TRL_lg_CT.xml:** Translations for AD_Task. Definition of

system tasks.

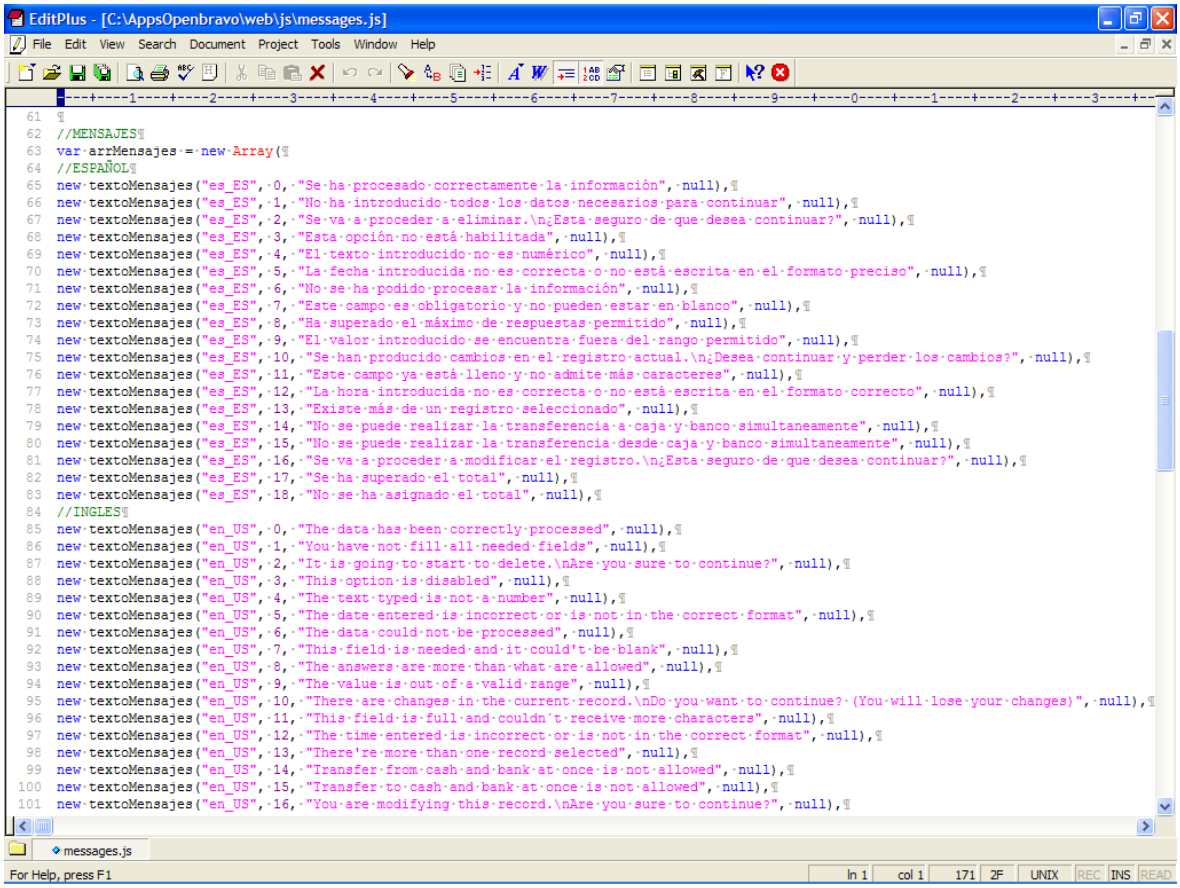
- **AD_TEXTINTERFACES_TRL_lg_CT.xml:** Translations for AD_TextInterfaces table. This table holds the texts that will be displayed in all the manually generated windows.
- **AD_WF_NODE_TRL_lg_CT.xml:** Translations for AD_WF_Node. This table contains the workflow nodes.
- **AD_WINDOW_TRL_lg_CT.xml:** Translations for AD_Window table. AD_Window has names, descriptions and helps of the WAD generated windows.
- **AD_WORKBENCH_TRL_lg_CT.xml:** Translations for AD_Workbench. This table is not used at this moment.
- **AD_WORKFLOW_TRL_lg_CT.xml:** Translations for AD_Workflow. Name, description and help for the defined workflows.
- **C_COUNTRY_TRL_lg_CT.xml:** Translations for AD_Country. This table holds a list of countries.
- **C_DOCTYPE_TRL_lg_CT.xml:** Translations for C_DocType. Description of the document types.
- **C_DUNNINGLEVEL_TRL_lg_CT.xml:** Translations for C_DunningLevel. This table is not used at this moment.
- **C_ELEMENTVALUE_TRL_lg_CT.xml:** Translations for C_PaymentValue table. This table contains the values for the elements used in accounting.
- **C_GREETING_TRL_lg_CT.xml:** Translations for C_Greeting.
- **C_PAYMENTTERM_TRL_lg_CT.xml:** Translations for C_PaymentTerm. Name and description for payment terms.
- **C_TAXCATEGORY_TRL_lg_CT.xml:** Translations for C_TaxCategory table. Name and description for the tax categories.
- **C_TAX_TRL_lg_CT.xml:** Translations for C_Tax table. Name and description for taxes.
- **C_UOM_TRL_lg_CT.xml:** Translations for C_OUM table. This table contains the Units of Measure.
- **M_PRODUCT_TRL_lg_CT.xml:** Translations for M_Product table. A table containing the different products.

I.2 Additional necessary changes

In order that the translation works properly it is necessary to apply additional changes in the files **messages.js** which can be found in:

- AppsOpenbravo/web/js/
- Tomcat/webapps/openbravo/web/js

The following script lines have to be copied and translated into a desired language:



- The above example shows the correct translation from Spanish to English. In any translation the same lines have to be added in a desired language. For example:

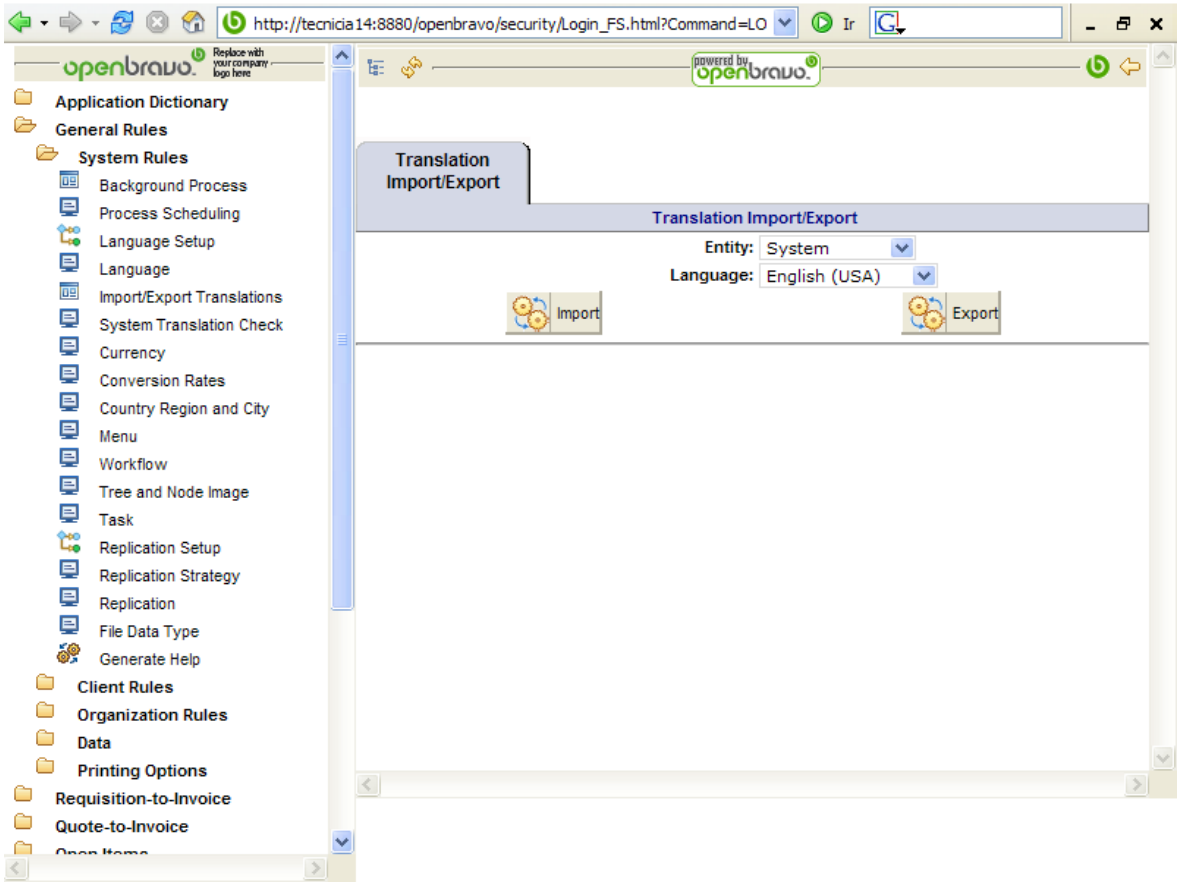
// GERMAN

...

etc.

I.3 Export/Import

To export or import a language the user must log into the application as *System Administrator* and go to *General Rules* > *System Rules* > *Import/Export Translations.*



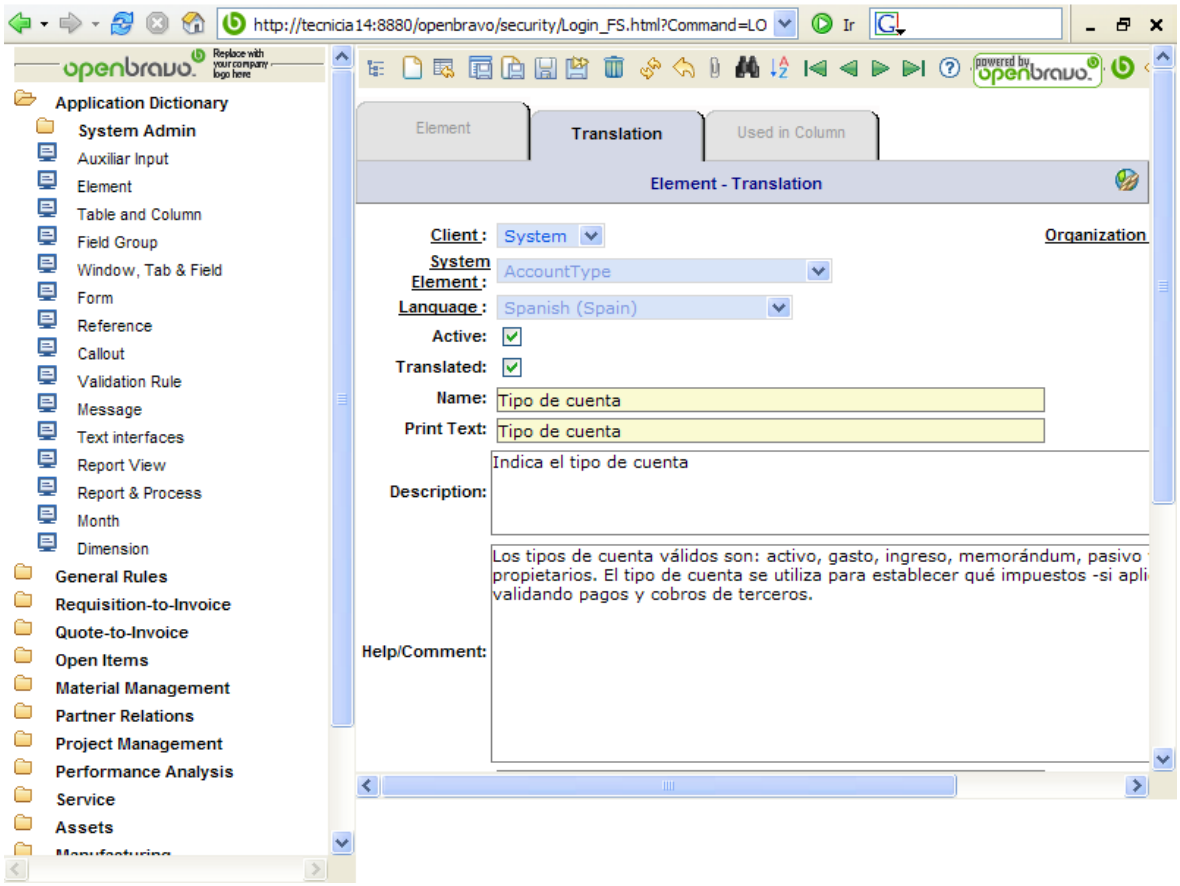
Here the user can choose one of the languages defined as *System Language*.

- ***Export***: This will create the language directory (see above) if it does not already exist and export files for the selected language to the directory.
- ***Import***: This will take the files from the selected language directory and import them into the application tables.

IV. Maintaining a Translation

If the user does not want to make a complete translation nor significant changes, he or she can modify a translation in a different way, without exporting/modifying/importing xml files. Every table that has a translation (the ones that are exported into xml files), also has in its window a tab called *Translation.* Here, translations can be edited for each row of the table.

For example, if the user simply wants to change the Spanish translation for the *AccountType* element, this change is made by going to the *Application Dictionary* > *Element* > *Translation* tab. Once there, changes can be made.



V. Compiling Openbravo

Even if the language pack has been imported or changes have been made, modifications will not take effect until Openbravo is re-compiled .

It is recommended web server be stopped before the compiling process begins. For example: if using *Tomcat* as web server in a *Linux*, shut it down by typing this in the console:

```
service tomcat stop
```

To compile this must be typed into the console from application root directory :

```
ant compile.complete
```

This process will compile and translate every window at the application.

Once everything is compiled, the user must create the war file, deploy the application and restart the web server. Following the previous example, the following would be typed in:

```
ant war
ant deploy
service tomcat start
```