Zaval JRC Editor
Version 2.0

User's Guide

Zaval Creative Engineering Group
http://www.zaval.org

Contents

Introduction to the Zaval JRC Editor

Since JDK 1.1.x. Sun introduced Internationalization concept for Java.

> *Internationalization* is the process of designing an application so that it can be adapted to various languages and regions without engineering changes.
>
> *Localization* is the process of adapting software for a specific region or language by adding locale-specific components and translating text.
>
> *The Java™ Tutorial*

The text used in programs is locale dependent, that's why we store all text strings in the separate files as a key-value pairs. When we need to localize a program for yet another language support we have to translate values in these files to the desired language. And that's all! These files are called *resources*. We provide GUI tool to manipulate these resources. The files being produced by this tool is fully compatible with ResourceBundle expectations.

So, the Zaval JRC Editor (also known as Zaval Java Resource Editor) solution is small and easy-to-use visual editor for resource files.

What Can You Do with the Zaval Java Resource Editor

Zaval JRC Editor is not a simply editing tool - it provides rich and flexible way to manage text-based resources. In fact, you are able to make any distributed translations as you want in several forms outside JRCE. But in all cases you will need to merge all changes after translation and do a clean up texts being translated; and JRC Editor provides all necessary features to accomplish these tasks.

The Zaval Java Resource Editor can be used for new and existing software localization, resources synchronization and any other resources manipulations.

It provides full support for any language specific resources (it depends on the fonts and font metrics settings of the host OS at your computer). The target of this tool is localization strings manipulation for all Java-based software that has appropriate support embedded.

When To Use the Zaval Java Resource Editor

The Zaval JRC Editor is best used for regular access to various resource files. You can add your own language support to the existing software if strings are not hard-coded to the software. One of the greatest things in the internationalization is that you don't need to make code changes.

Another great area in this tool usage is resource bundle synchronization. Our tool can handle this task easily – it compares the files set and highlights all differences. It allows separating development process and resource management process. That's why this tool can be used as part of your software pack to provide 3rd party localization.

Internationalization concept overview

Internationalization means that developers can customize their product for different languages and locales. In Java language terms this task handles by ResourceBundle

class (see [1,2,3,4]). All resource files' names have two parts: resource name and localization suffix. When you want your program to 'speak' French, for example, the only thing you need to do is adding corresponding properties file. This procedure does not require any interaction with the software creator.

Installing the Zaval Java Resource Editor

We did our best to make the installation procedure of the product easy, so just follow the instructions.

Requirements

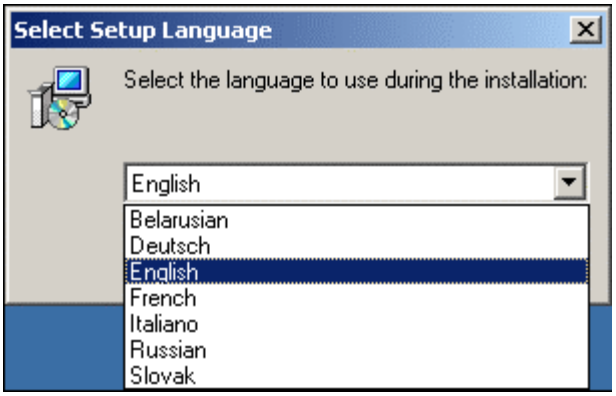
To use the Zaval JRC Editor you need a JDK/JRE installed on your computer. All Sun's JDK (versions since 1.1.7) can be used without any problems. We suggest you to use Sun's JDK 1.2.2 or later as most stable (see [5]). Alternatively you can use any other vendors' JDK (IBM's were ok for versions 1.2.x, 1.3.x and 1.4).

Installation procedure

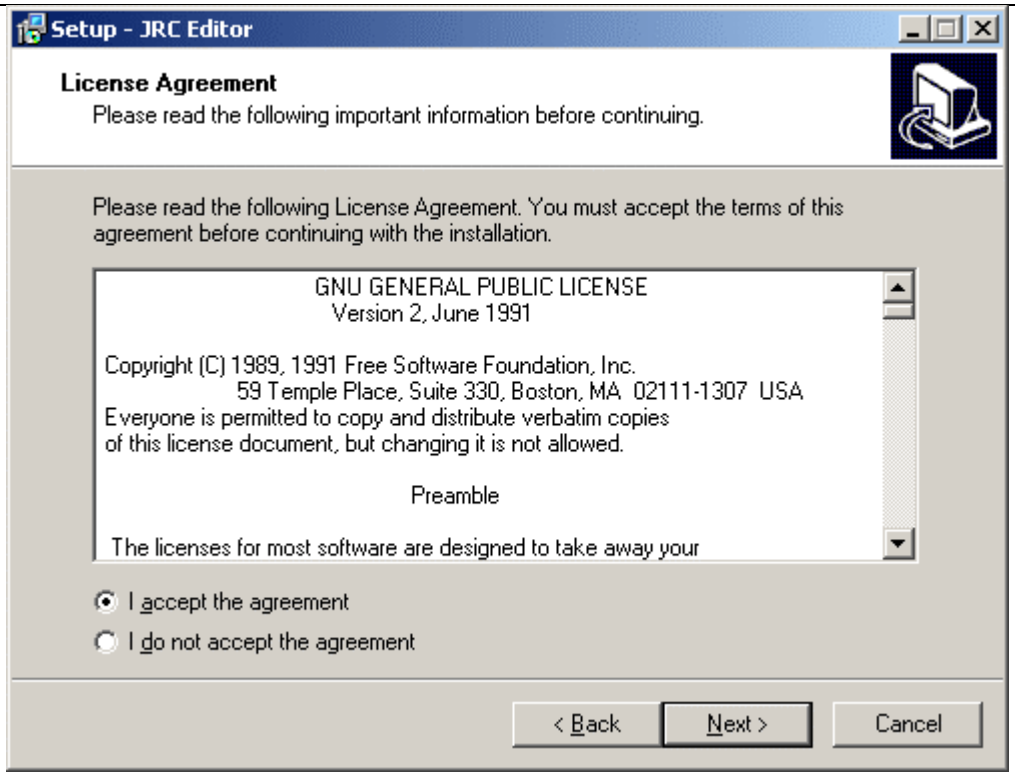
Please, download the package corresponding to your OS to make the installation process easier.

- **Installing on Microsoft Windows (jrc-editor-2.0.X-setup.exe)**

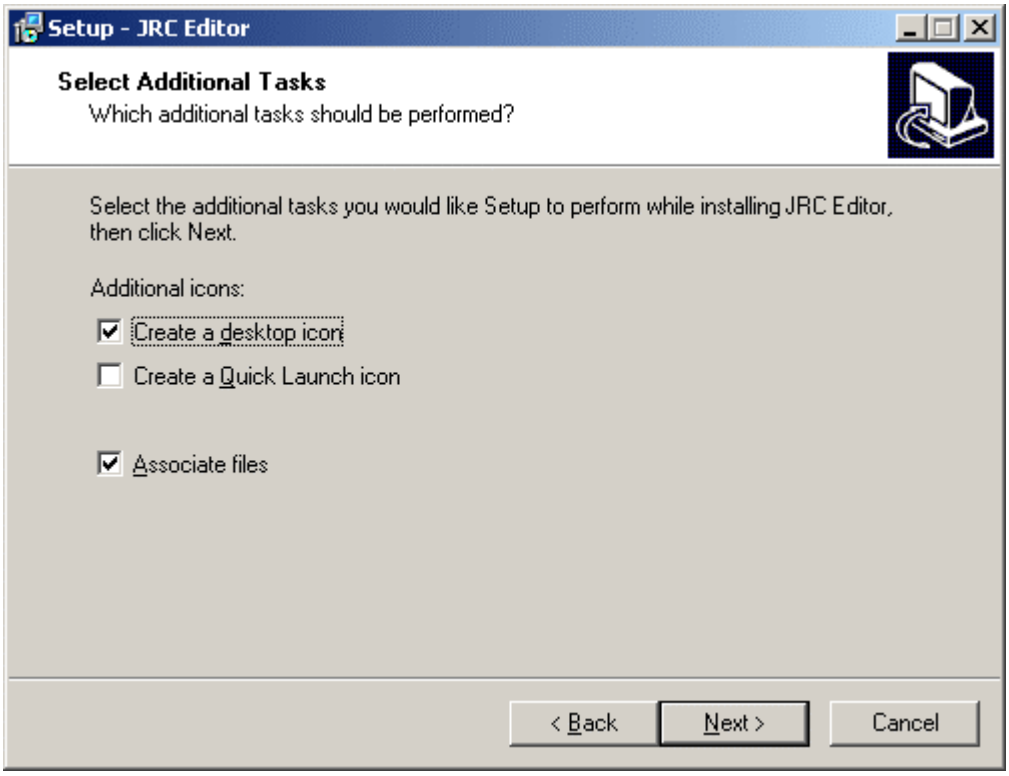
  Run the installation wizard and simply follow the instructions:



Screenshot 1. Choosing installation language



Screenshot 2. Accepting the GNU General Public License agreement



Screenshot 3. Additional options

- **Installing on Unix/Linux (jrc-editor-2.0.X-1.i586.rpm)**

Use rpm: **rpm -i jrc-editor-2.x.x-x.i386.rpm**

- **Installing on any OS (jrc-editor-2.0.X.tar.gz)**

  1. Unpack zip/tarball with Zaval JRC Editor to an empty directory.
  2. If you are using JDK 1.2 or above start the batch file (jrc-editor.bat) to run Zaval Java Resource Editor, otherwise go to step 3.
  3. For any other non-standard or old versions of JDK/JRE you need to make several changes to the batch file (or you need to setup environment variables):

```
set JAVA_HOME=<location of JDK>
set PATH=%JAVA_HOME\jre\bin;%PATH%
set CLASSPATH=.;%JAVA_HOME%\jre\lib\rt.jar;%CLASSPATH%
```

     The last line can have variation for old versions of JDK, such as 1.0.2 or 1.1.x. In this case you need specify the following line instead:

```
set CLASSPATH=.;%JAVA_HOME%\jlib\classes.zip;%CLASSPATH%
```

  When you complete the steps above the Zaval JRC Editor is ready to use.

Using Zaval JRC Editor

Application start

To start this application you need simply start corresponding batch file. When the application is started you are able to manipulate resource files.

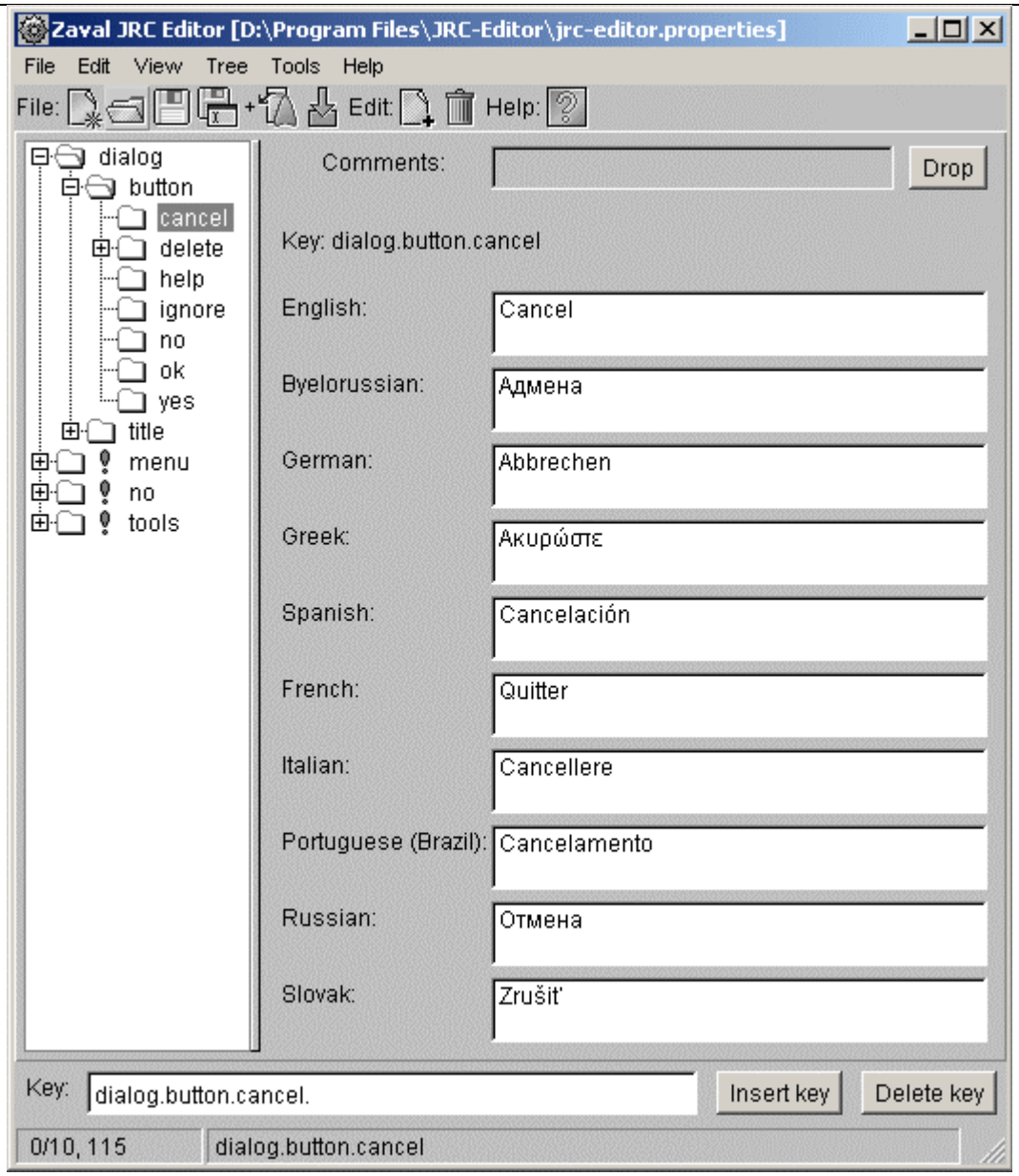
Editing existing resources

Zaval Java Resource Editor consists of two panels displayed on Screenshot 1. To start working with your resource set you should make several steps:

- From the **File** menu, choose **Open** or **New resource**.
- In the opened dialog select resource file that meets mentioned requirements. **Note:** the files previously opened will be closed automatically.

Left panel is a key hierarchy that was stripped out from the appropriate resource files set. The key hierarchy is created automatically using "." or "_" as a separator (see section **Preferences**). So, adding "." in key name automatically creates new hierarchy level.

Right panel displays a list of the languages defined by the resource files set and values assigned to them.



Screenshot 4. Working area of the Zaval Java Resource Editor

To hide a language from the right panel go through the following steps:

- From the **View** menu, select **Show resource**.
- In the **Show resource** list, clear a checkbox next to the language.

To edit key property for any language available do the following:

- In the left panel select item you want to edit - it will appear in the **Key** field at the bottom of the panel (in our example **dialog.button.cancel**).
- In the right panel make necessary changes to the corresponding field (see Screenshot 4).

Note: starting from version 2.0 you can use Shift and Ctrl key for fast navigation inside text.

To apply changes use **File -> Save** from the top menu or press on the  (**Save**) icon. Resource file's name consists of base name (e.g. **'editor'**), language suffix (e.g. **'de'**), and **.properties** extension, for example, **editor_de.properties** (German), **editor_fr.properties** (French). In our example **jrc-editor.properties** file contains default English messages and is displayed in the title of the Zaval Java Resource Editor's panel.
To save all files with a different base name and/or write them to a different directory, use **File -> Save as** or press  (**Save As...**) icon instead.

Creating new resources

There are several ways to create new resources: either enter them by hands, or importing them from *.java files, *.jar file, Unicode file or XML file. For more info see **Importing resources from external sources** section.

Inserting new keys

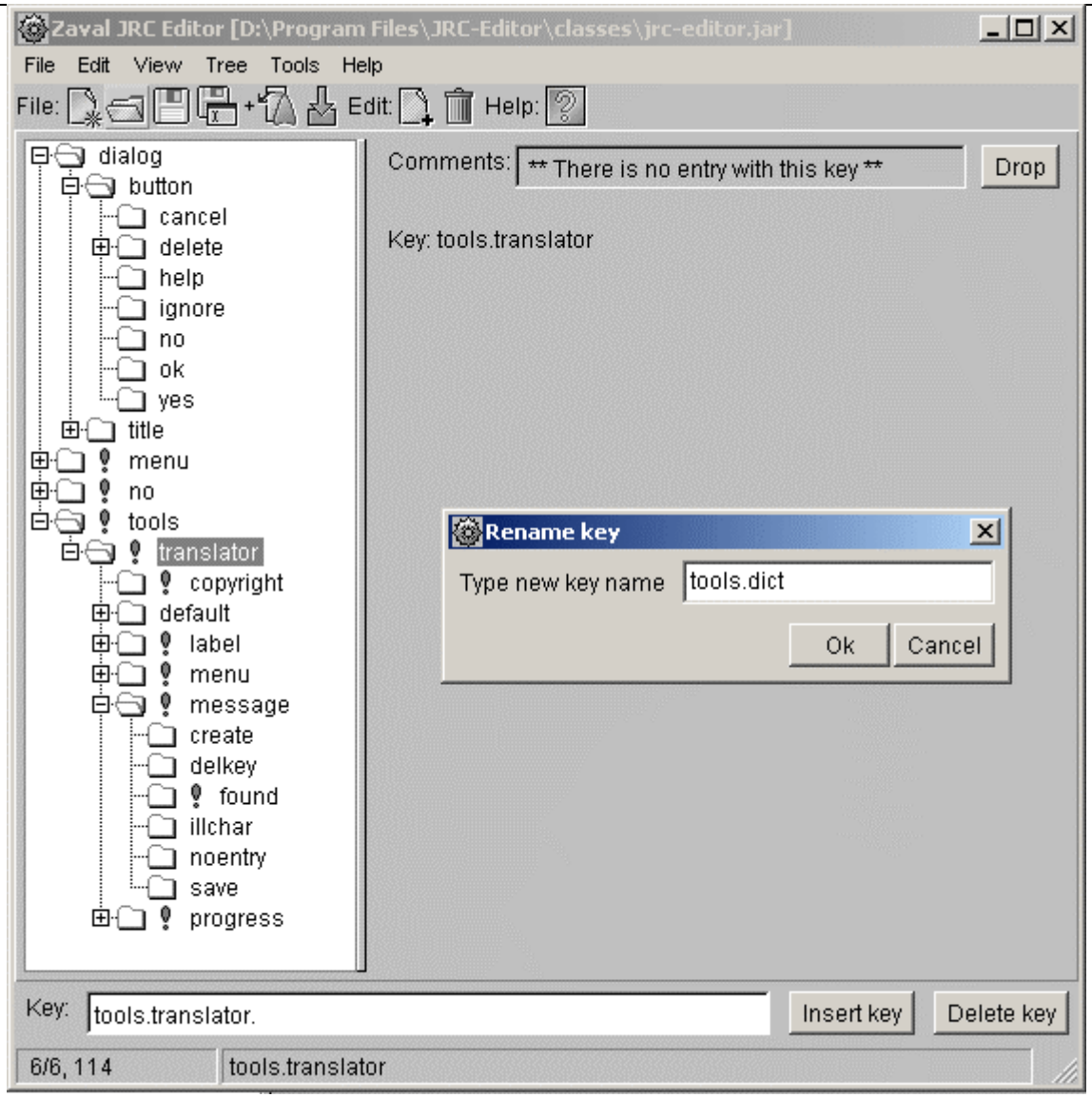
To add your own keys to the tree proceed through the following steps:

- In the **Key** field, type tree key, for example **dialog.button.new**.
- Press **Insert** button.
- Fill in all Key's values.
- Repeat these steps for all items you want to add.

Renaming keys

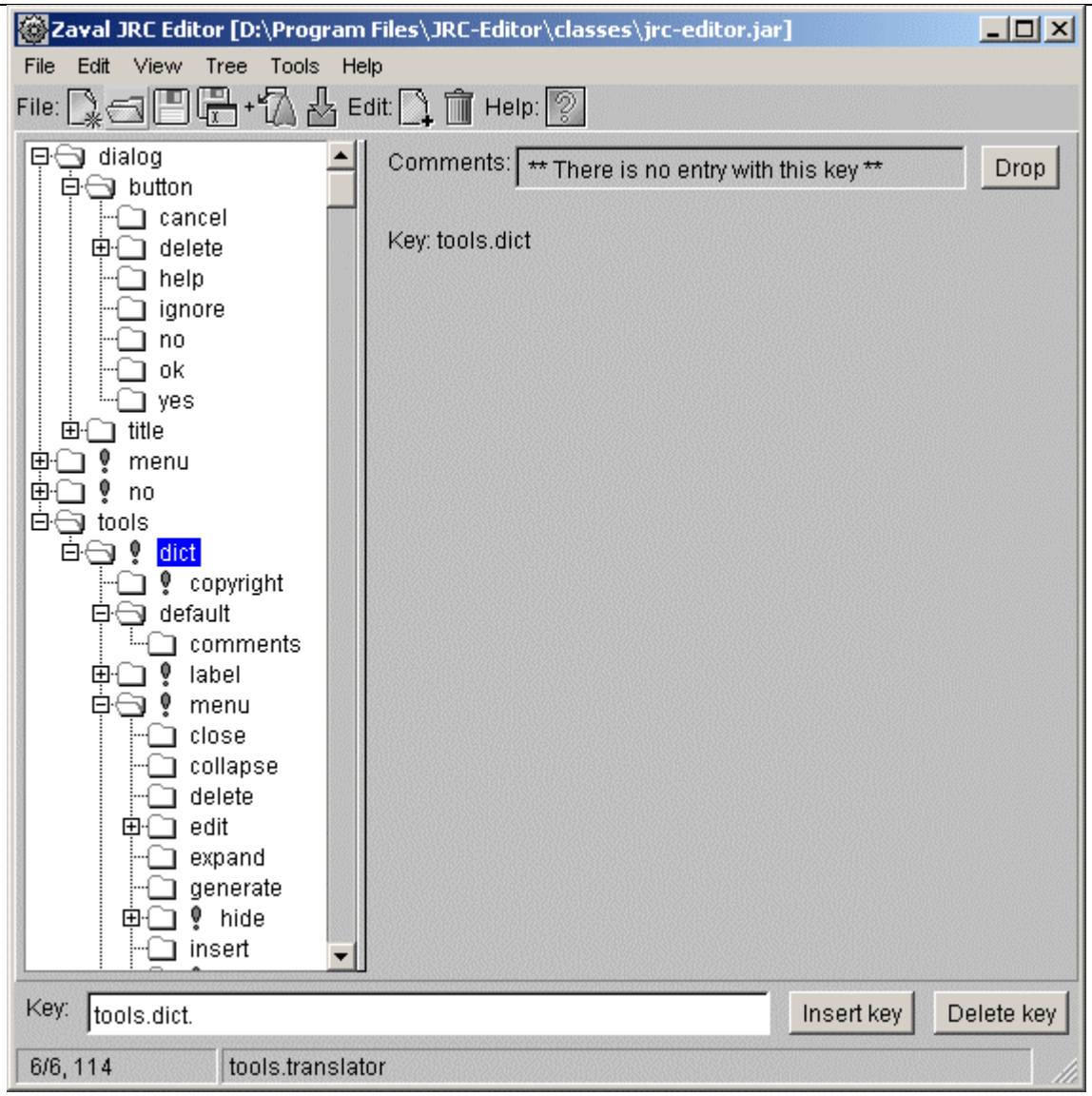
To rename a key or key hierarchy you should do the following:

- Choose a key to rename.
- Choose **Rename key** from the right mouse context menu or use **Edit -> Rename key** (see Screenshot 5).



Screenshot 5. Renaming key – entering new name

- Enter new key name.



Screenshot 6. The whole sub-tree was renamed

The whole underlying key hierarchy was renamed.

Deleting keys

The remove a key-value pair do the following:

- Select the appropriate key at the left panel.
- Press **Delete** button.
- You will be asked either you want to remove this key with all sub-keys or this key only.

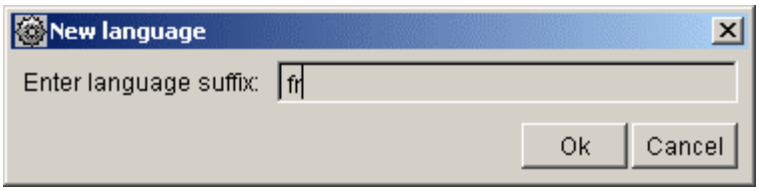
Note: removing keys from the .properties file by hand may cause your application to fail. Think over before you proceed. Unused keys will do no harm to the application stability.

Adding new language

Adding new language is simple:

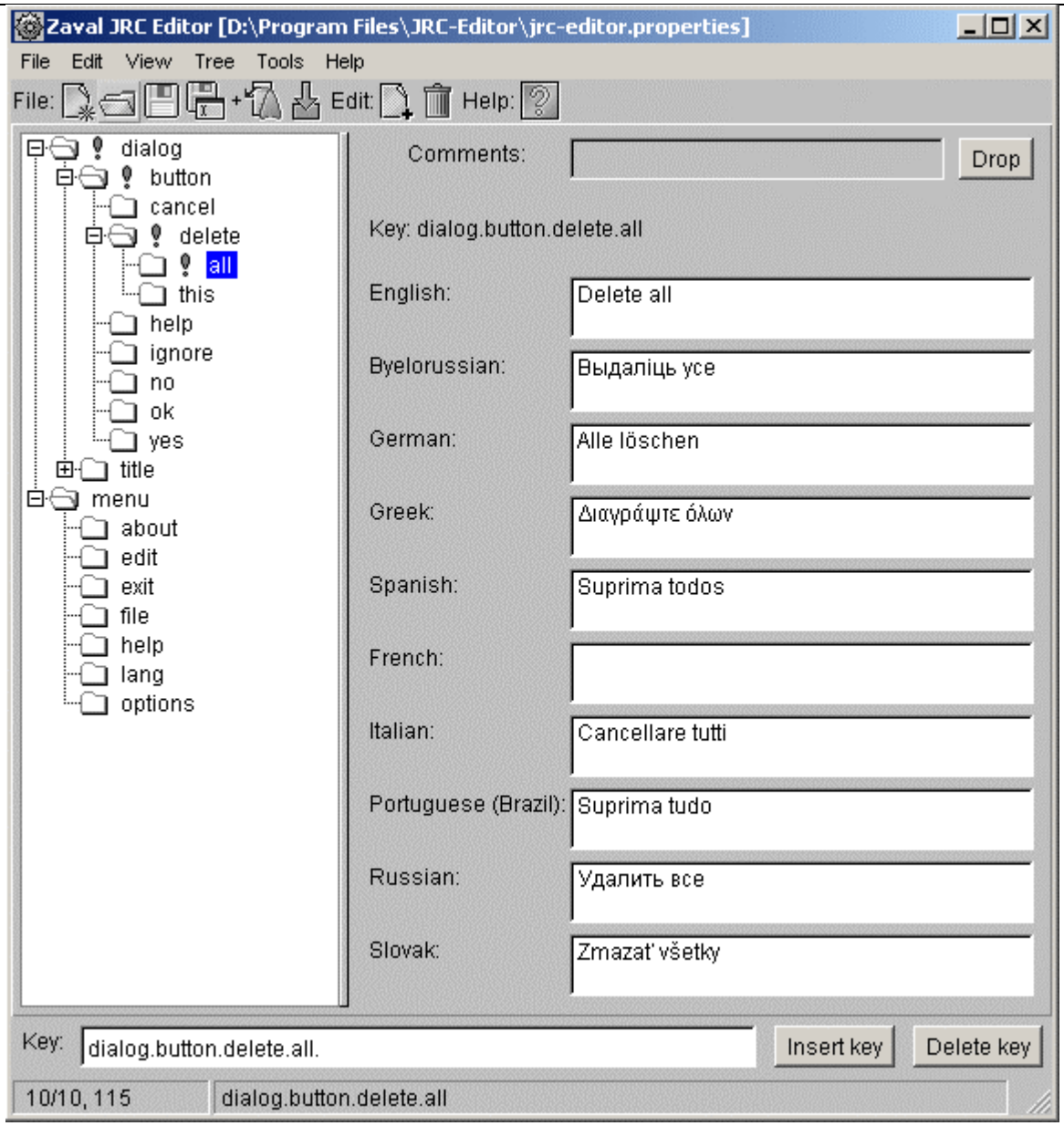
- Choose **Edit -> New language**.
- In the **New resource** dialog (see Screenshot 7), enter resource file suffix, for example, fr. When you click OK corresponding .properties file will be created (in our example **editor_fr.properties**) and edit fields for French language will appear in the right panel.

Note: for information on language codes (suffixes you need) see [6,7].



Screenshot 7. Adding new language

An exclamation mark next to tree node notifies you that there is at list one item in this node that is not translated into all languages displayed in the right panel (see Screenshot 8).

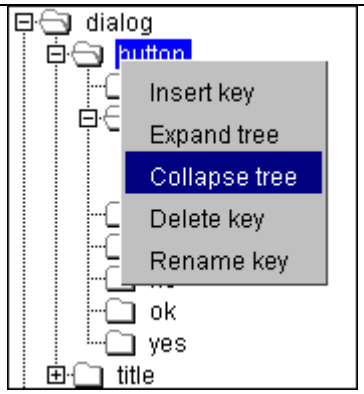


Screenshot 8. Resources synchronization

In our example missing translation is French. After filling in the filed near French exclamation mark will disappear.

Tree manipulations

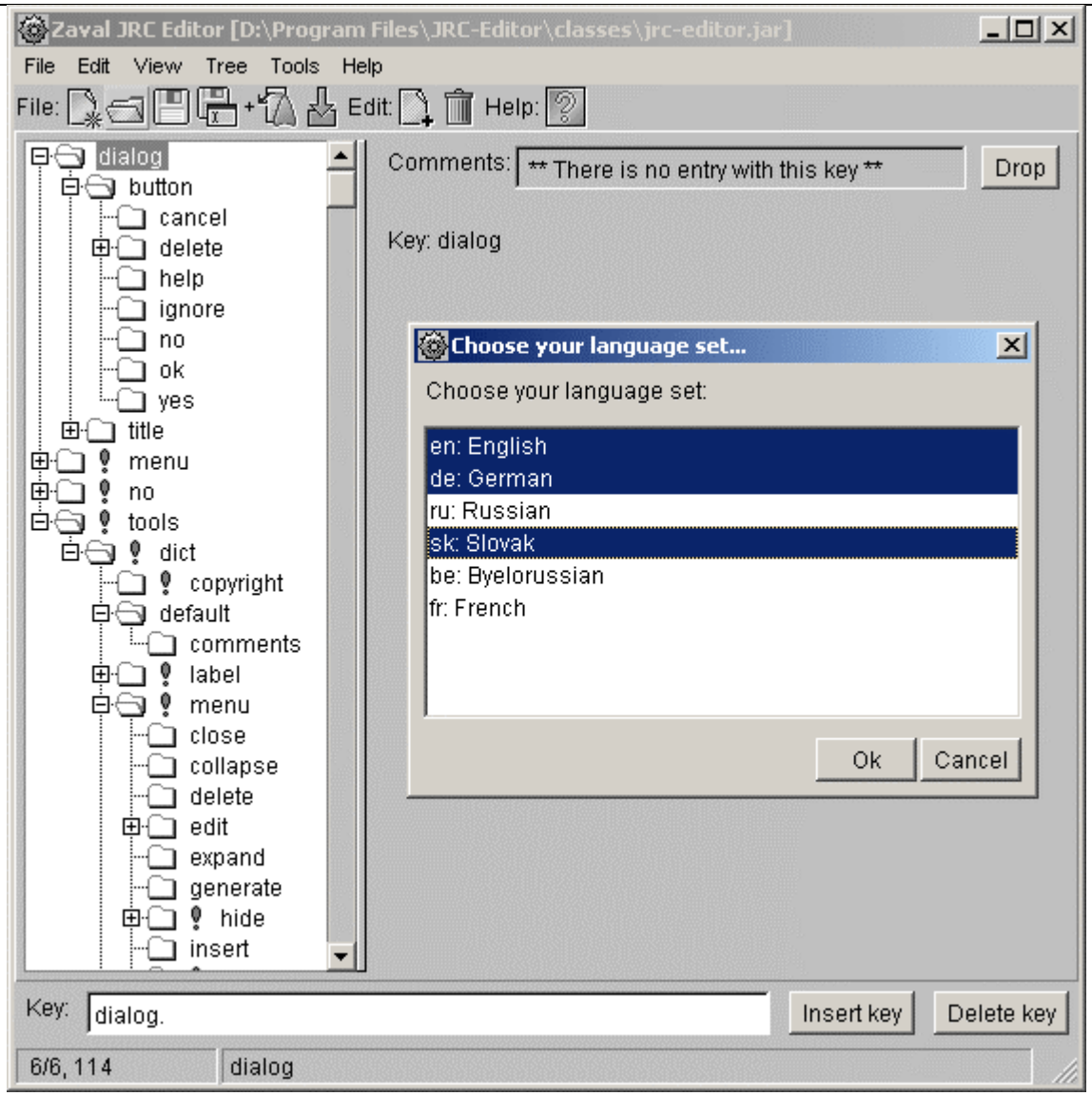
To make your life easier when working with really big files we added expand/collapse functionality to whole tree and any of its nodes. This feature is available under **Tree** menu and starting from version 2.0 from context menu (available with mouse right-button click).



Screenshot 9. Context menu

Import/Export

Starting from version 1.2 JRC Editor supports data import/export from/to single XML file (UTF-8 encoding) or single plaintext file (Unicode). Formats are pretty simple, so there shouldn't be any problems.
While exporting to Unicode you may choose to either export all resources or just choose only those you need.



Screenshot 10. Export to Unicode (Split into Unicode file)

Unicode file format:

KEY="key.name"
"language1"="value1"
"language2"="value2"
"language3"="value3"

Example:

KEY="dialog.button.cancel"
"en"="Cancel"
"sk"="Zrušiť"
"de"="Abbrechen"

XML file format:

```
<xml>
  <key name="key.name">
    <value lang="language1">value1</value>
    <value lang="language2">value2</value>
  </key>
</xml>
```

Example:

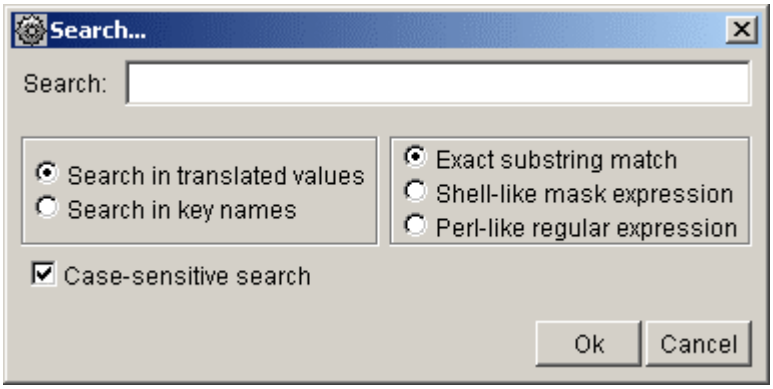
```
<xml>
  <key name="dialog.button.cancel">
    <value lang="en">Cancel</value>
    <value lang="sk">Zrušiť</value>
    <value lang="de">Abbrechen</value>
  </key>
</xml>
```

Fast navigation

To make your life easier we have added shortcuts support for all important actions. For example: Save – Ctrl+S, Open – Ctrl+O, etc.
Starting from version 1.2 you can access all JRC Editor functions without a mouse support – you can navigate through all JRC Editor controls with Tab key, access upper menu functions using shortcuts and more.

Search / Search and Replace

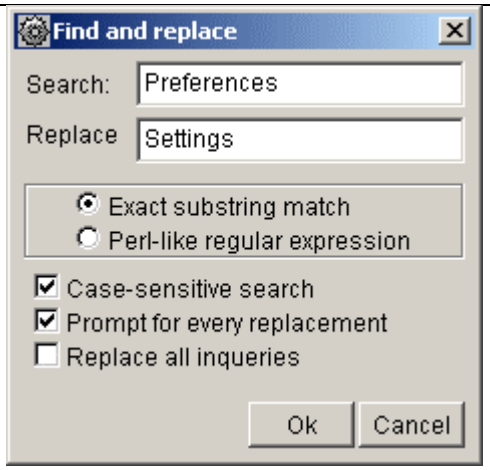
Starting from version 2.0 JRC Editor supports **Search** and **Search and replace** functions.



Screenshot 11. Search dialog

Search dialog supports almost all options possible:

- **Exact substring match** – only exact substring will be matched.
- **Shell-like mask expression** – it is possible to use simple masks like you used in MS-DOS for searching files. I.e. symbol '*' is treated as any character sequence, '?' as any single symbol, and '[]' as one character of the list specified in square braces.
- **Perl-like regular expression** – see 'man perlop' and 'man perlre' for details.



Screenshot 12. Search and Replace dialog

Importing resources from external sources

With JRC Editor you can easily import resources from either *.java file, *.jar archive (in release 2.0 all *.properties files found in this archive will be merged in one file), XML file or from Unicode file.

Importing resources from *.java file

To simplify existing applications transfer to the resources usage you can use **Parse source...** option. It goes through java source, extracts text strings and builds tree-like structure based on class name/package name. In version 2.0 you can import resources only from one file at a time and can't do iterations.
Importing process doesn't get all strings but only those that look right: for example, it wouldn't import any single-character strings.

Importing resources from *.jar package

Starting from version 2.0 we have added **Import resource from JAR...** option. How it works: it goes through archive (as far as you should know jar is a zip archive), finds all files with extension *.properties, merges all of them in single resource bundle regarding to the language extensions. In version 2.0 you don't have possibility to specify which resource file to import from *.jar and which – not to import.

Resource stencil source code generation

Stencil is a java source that contains all resource strings with their initialization and corresponding getters/setters. Initialization requires appropriate resource bundle.
To generate use **File -> Generate source code...**.

Understanding Join/Merge features

You can use JRC Editor with the following way:

1. Define new language inside JRC Editor;
2. Export only two languages: English and languages you've selected;
3. Send content being exported into unicode text file to any translators;
4. People who translates text works separately without intersections;
5. All content being translated able to merged back into common pool.

As you see, initial and final phases can be managed efficiently with JRC Editor, but real translation process can be done either with JRC Editor or without it – you can choose the most appropriate method you need.

Null keys

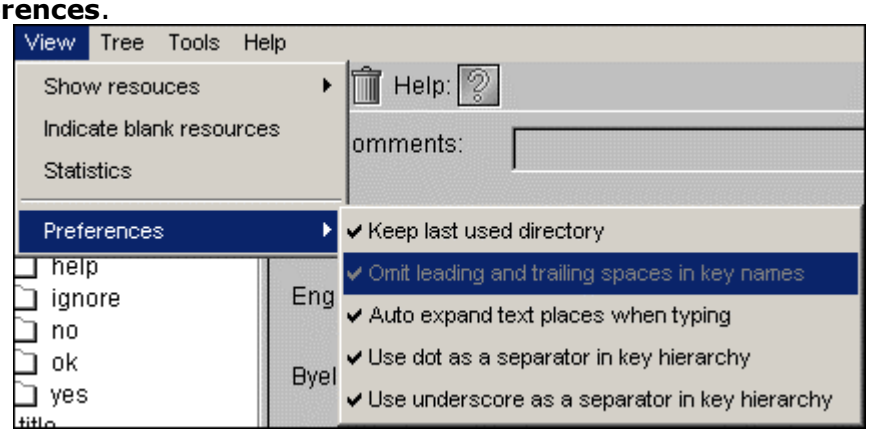
Null keys or blank keys are keys that have no value for all languages. Often it can be useful to see all of them (see **View -> Indicate blank resources**).

Statistics

Sometimes it is useful to see statistics over files you are working with, so we added this functionality (see **View -> Statistics**).

Preferences

There are several options that can be modified in run-time. They are available under **View -> Preferences**.



Screenshot 13. Preferences

Namely:

- **Keep last used directory** – controls what directory you will see while doing multiple **Open**, **Save as...** and other file operations.
- **Omit leading and trailing spaces in key names** – controls whether leading and trailing spaces are kept or removed from key names (in version 2.0 this option can not be changed).
- **Auto expand text places when typing** – in the right panel when you entering new value for any key text area will be automatically enlarged when needed.
- **Use dot as a separator in key hierarchy** – to build the tree in the left panel we need to know the separator in key names, this option just let JRC Editor use dot as a separator.
- **Use underscore as a separator in key hierarchy** – to build the tree in the left panel we need to know the separator in key names, this option just let JRC Editor use underscore as a separator.

Configuring Zaval JRC Editor

Some JRC Editor's functions can be configured via command-line options. These options are global and cannot be changed in run-time. All options can be set via

standard 'set' command, or can be attached to command line with '-D' parameter before main class name (the main class name is 'org.zaval.tools.i18n.translator.Main').

Options:

- 'inputControls' provides support of input controls ('native' or 'ownerdraw'(default)).

  In most cases 'native' is proper choice (for fully-unicode environment, such as WinNT/2K/XP, and Unix'es with *.UTF-8 locale settings). But if you are using old distributions (XFree 3.x for example, or Win95/98/Me, or you are using JDK 1.1.x on any platform) you may cause problems with typing unicode characters. This can be solved by using 'ownerdraw' setting.

  Note: in some cases 'ownerdraw' parameter can cause incorrect typing of East Asian national key strokes, such as China, Japanese anod other (our controls has no support of composite key symbols); so we recommend to use 'native' text fields in this case.

- 'user.language' and 'user.region'

  This option can be used to override your system locale. For example, if locale in your system is set to English (US) and you want to start JRC Editor with Italian interface you can do this by specifying user.language=it and user.region=IT

Examples

To set inputControls to 'native' modify starting script as showed below:

java -DinputControls=native -jar classes/jrc-editor.jar %1 %2 %3 %4 %5 %6 %7 %8 %9

To start JRC Editor with Italian GUI:

java -Duser.language=it -Duser.region=IT -jar classes/jrc-editor.jar %1 %2 %3 %4 %5 %6 %7 %8 %9

Command-line tool

Starting from v1.3.1 JRC Editor goes with command-line tool that allows you doing all resource bundles manipulations automatically, including split and merge features for every language you've set. Command line tool allows doing the same things as JRC Editor tool provides interactively, including source files parsing and code generation.

Operating with command-line version of the JRC Editor

JRC Editor command-line tool has the following syntax:

join srcFile ... addFile
or
split srcFile dstFile [lang ...]

where:

srcFile - a root file of properties bundle set. If this file does not exist it will be created automatically if needed; otherwise all data will be joined with content being added previously; the same keys will be replaced from addFile files if specified.

dstFile - XML or UCS16 text file. This file will be created from scratch (if file with the same name already exists it will be replaced). File type will be determined by extension of file specified (.txt, .java, or .xml can be used here).
You able to use this command to generate Java code stub files, but in this case all language parameters will be omitted. To use this feature you need to specify '.java' file extension here.

addFile - XML, Java, other bundle set or UCS-16 file. Content of these files will be joined with srcFile content, and srcFile will be replaced with new one. There is a common situation when the same keys exist in several files at once - join operation goes sequentially, so in this case last file will have higher priority.

lang - locale abbreviation (suffix of slave properties files). You can specify any language bi-literal codes here, such as en, ru, de, or any other values for east asian languages. All languages being specified will be added to appropriate target file. If no languages specified file will contain only key names without any translations.

Product limitations

There are several product limitations that probably will be fixed in the future versions:

- You can't operate with multiple resource sets – only one resource set available at a time.
- You cannot analyze a lot of Java sources at a time.
- You cannot split all lines being generated to multiple targets (Java files).

Further product plans

Current tool implementation follows the minimalist computing concept. In near future the following features will be added:

- Ability to merge different resource sets;
- Ability to split one existing resource set to multiple ones;
- Code generation and analysis improvements and multiple operations support;
- Automated machine translation and spell-checking support.

Support available

All support for software installation and problems should be sent directly to support@zaval.org with "Re: Zaval JRC Editor Support" in subject line and plain text in the message body, describing your request and/or your problem. Since this software is distributed under the General Public License and is maintained by its authors on non-

commercial basis, your request will be answered as soon as possible, but no later than 5 business days.

The Zaval Creative Engineering Group carries out its software customization/new software development on the regular basis. For more info contact us at info@zaval.org.

Stay informed!

Now you can receive information on latest products' updates and hotfixes via email. This is a low-traffic list (1-2 messages per month). To subscribe, send blank mail to news-subscribe@zaval.org.

References

1. http://java.sun.com/docs/books/tutorial/i18n/
2. http://java.sun.com/j2se/1.4.2/docs/guide/intl/
3. http://java.sun.com/developer/technicalArticles/Intl/
4. http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/ see ***Chapter 10: J2EE Internationalization and Localization***
5. http://java.sun.com/j2se/
6. http://www.loc.gov/standards/iso639-2/englangn.html
7. http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt