Script Translation

About Multilingual Scripts

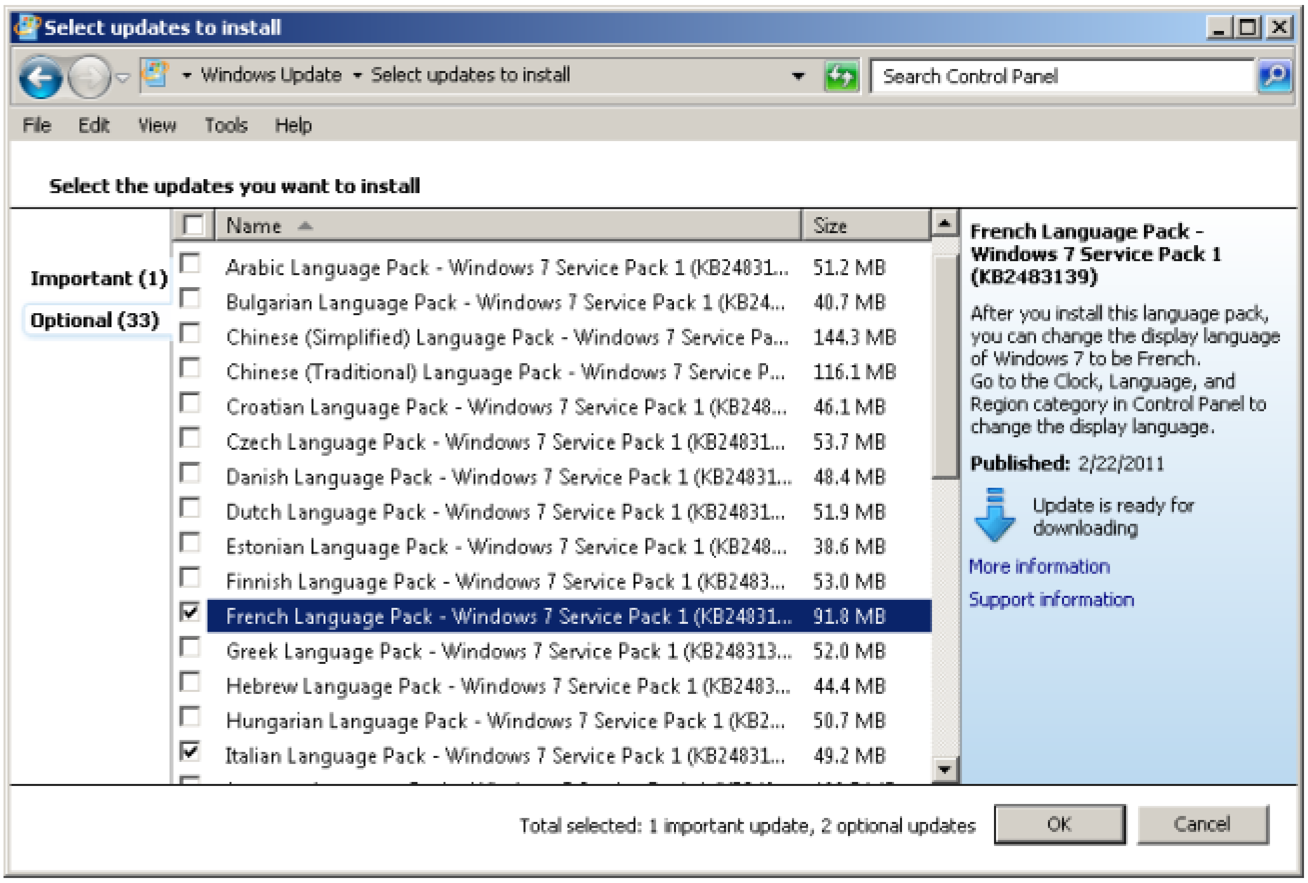
BPS itself has all inline texts in English as basic language, and then translation files are added for German, French etc. This means also that when running on a desktop in a language where there exists yet no BPS translation, all texts will be in the base language English.

Technically there is nothing stopping you from using any other language as base, but then you would get a mix of your base language and English whenever the application is running on a desktop language not yet translated. Also take into account that most every translator is able to understand English, while the choise may be limited for other language combinations.

Prerequisits

Windows Display Language

To test translations you must have the respective languages installed in Windows. On *Windows 7 Ultimate* and *Enterprise* you can conveniently install additional languages with *Windows Update*:


Select updates to install
Windows Update ▸ Select updates to install
Search Control Panel
File Edit View Tools Help
OK
Cancel

Select the updates you want to install

Important (1)
Optional (33)

	Name	Size
☐	Arabic Language Pack - Windows 7 Service Pack 1 (KB24831...	51.2 MB
☐	Bulgarian Language Pack - Windows 7 Service Pack 1 (KB24...	40.7 MB
☐	Chinese (Simplified) Language Pack - Windows 7 Service Pa...	144.3 MB
☐	Chinese (Traditional) Language Pack - Windows 7 Service P...	116.1 MB
☐	Croatian Language Pack - Windows 7 Service Pack 1 (KB248...	46.1 MB
☐	Czech Language Pack - Windows 7 Service Pack 1 (KB24831...	53.7 MB
☐	Danish Language Pack - Windows 7 Service Pack 1 (KB24831...	48.4 MB
☐	Dutch Language Pack - Windows 7 Service Pack 1 (KB24831...	51.9 MB
☐	Estonian Language Pack - Windows 7 Service Pack 1 (KB248...	38.6 MB
☐	Finnish Language Pack - Windows 7 Service Pack 1 (KB2483...	53.0 MB
✓	French Language Pack - Windows 7 Service Pack 1 (KB24831...	91.8 MB
☐	Greek Language Pack - Windows 7 Service Pack 1 (KB248313...	52.0 MB
☐	Hebrew Language Pack - Windows 7 Service Pack 1 (KB2483...	44.4 MB
☐	Hungarian Language Pack - Windows 7 Service Pack 1 (KB2...	50.7 MB
✓	Italian Language Pack - Windows 7 Service Pack 1 (KB24831...	49.2 MB

French Language Pack - Windows 7 Service Pack 1 (KB2483139)

After you install this language pack, you can change the display language of Windows 7 to be French.
Go to the Clock, Language, and Region category in Control Panel to change the display language.

Published: 2/22/2011

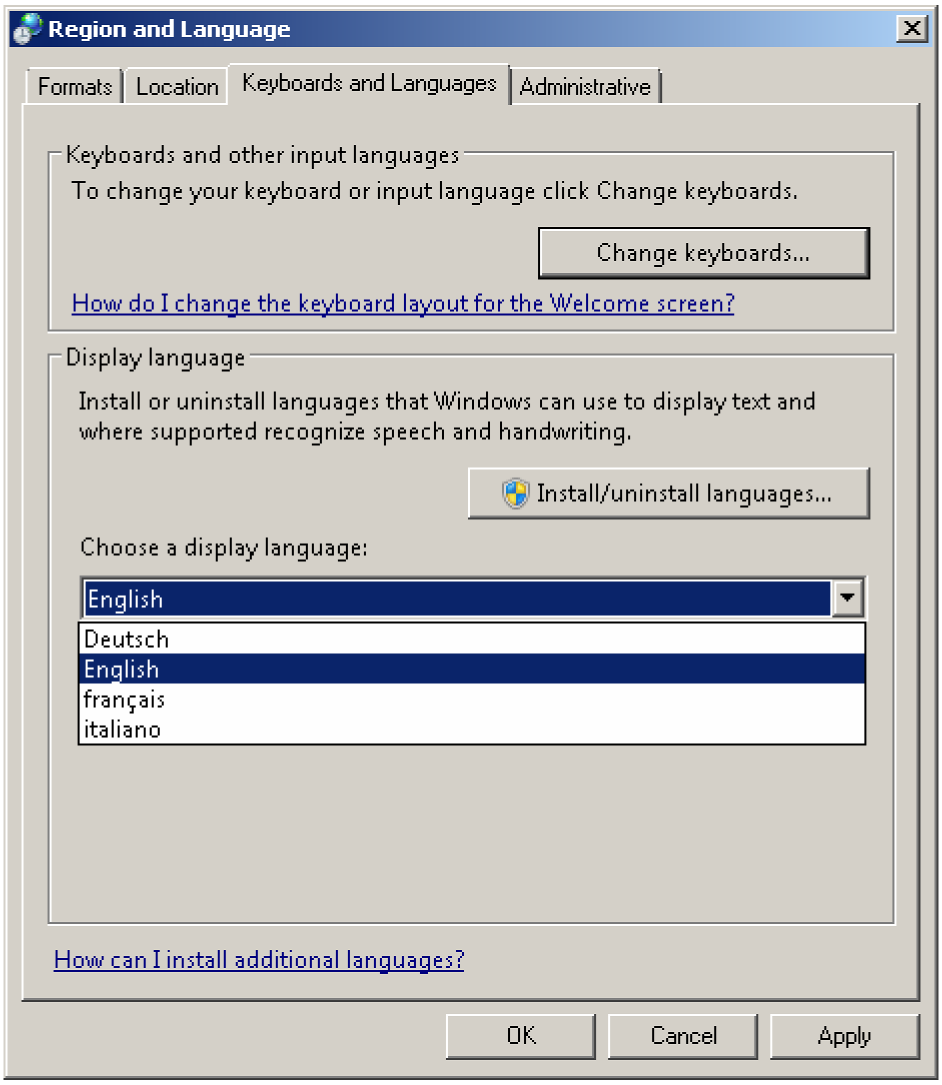
Update is ready for downloading

More information

Support information

Total selected: 1 important update, 2 optional updates

After installing additional languages you can switch the display language of the current user in the *Region and Language* settings:


Region and Language

Keyboards and other input languages

To change your keyboard or input language click Change keyboards.

How do I change the keyboard layout for the Welcome screen?

Display language

Install or uninstall languages that Windows can use to display text and where supported recognize speech and handwriting.

Choose a display language: English

Deutsch
English
français
italiano

How can I install additional languages?

The display language will be recognized by BPS applications, and if appropriate translations files exist they will get used.

Lupdate And Linguist

The translation process is supported by the *lupdate* utitity which extracts the translatable texts from a script, and the *Qt Linguist* which allows to translate the texts in the GUI application and then to generate a translation file that can be loaded and applied by BPS at script execution.

While it is up to the developer to extract the translatable file with lupdate into TS files, the actual translation with *Qt Linguist* may be delegated to a interpreter firm in the source and target languages. The interpreter returns the processed TS files to the developer who creates the QM translation files for distribution with the scripts.

In case you do not have these tools installed yet, please read in chapter Tools of the previous article *Basic Sctipts* how to get and install them.

Making Scripts Translatable

Making a script translatable is a process of identifying all fixed texts in your script that need to be translatable and embedd them in the special `qsTr()` or `qsTranslate()` functions.

Example:

```
print(qsTr("Hello world"));
```

The `qsTr()` function uses the basename of the script's filename as the translation context. If the filename is *not unique* in your project, you should use the `qsTranslate()` function and pass a suitable context as the first argument. You must especially use this also to translate extensions, because every extension script has the same name, `__init__.js`.

Example:

```
ext.text = qsTranslate("AcmeBaseExtension", "Hello world!");
```

The `String.prototype.arg()` function offers a simple means for substituting arguments. This way the translation texts stay as whole instead of being broken into several parts of the sentence which may be difficult to translate:

```
function showProgress(aDone, aTotal, aCurrentFileName)
{
    print(
        qsTr("%1 of %2 files copied.\nCopying: %3")
            .arg(aDone)
            .arg(aTotal)
            .arg(aCurrentFileName)
    );
}
```

Produce Translations

Translation of Qt Script scripts is a three-step process:

1. Run `lupdate` to extract translatable text from the script source code, resulting in a message file for translators (a TS file). The utility recognizes qsTr() and qsTranslate() functions described above and produces TS files (usually one per language).
2. Provide translations for the source texts in the TS file, using *Qt Linguist*.
3. Use the `Release` function of *Qt Linguist* to create a light-weight message file (a QM file) from the TS file, suitable only for end use. Think of the TS files as „source files“, and QM files as „object files“. The translator edits the TS files, but the users of your application only need the QM files. Both kinds of files are platform and locale independent.

Typically, you will repeat these steps for every release of your application. The `lupdate` utility does its best to reuse the translations from previous releases.

When running `lupdate`, you must specify the location of the script, and the name of the TS file to produce.

Example:

```
lupdate myscript.js -ts myscript_de.ts
```

Extracts translatable text from `myscript.js` and creates the translation file `myscript_de.qs`. Use this for simple standalone scripts.

```
lupdate mydialog.js mydialog.ui -ts mydialog_de.ts
```

Extracts translatable text from both, the script file `mydialog.js` and the user interface file `mydialog.ui`, and file create the translation file `mydialog_de.qs`.

Use this when your script loads one or more `UI` files (just list all files in case there are multiple). The `UI` files are automatically translatable, `lupdate` extracts the string elements of all properties so you need not to do any special preparation for it.

Sample Translation

Now let's make our custom dialog created in the Script Basics article translatable. We start by using `qsTr()` and the `String.prototype.arg()` function in the script file, as explained before:

```
if (!isGui) throw Error(qsTr("Please use gui.exe to run this script."));

importExtension('bps.gui');

// get widgets
var dialog = bps.gui.loadUiFile(0, bps.gui.uiFileName);
var lnedit = dialog.findChild('mLineEdit');
var cmbbox = dialog.findChild('mComboBox');
var chkbox = dialog.findChild('mCheckBox');
var buttons = dialog.findChild('mButtonBox');

// slot functions
function okPressed()
{
    bps.gui.MessageBox.information(
        dialog, qsTr('OK'),
        qsTr(
            '<h1>You pressed OK!</h1>'+
            '<p>Line Edit Text: %1</p>' +
            '<p>Combo Box Selection: %2</p>' +
            '<p>Check Box Status: %3</p>'
        )
        .arg(lnedit.text)
        .arg(cmbbox.currentText)
        .arg(chkbox.checked ? qsTr('checked') : qsTr('unchecked'))
    );
    dialog.accept();
} // okPressed

// make connections
```

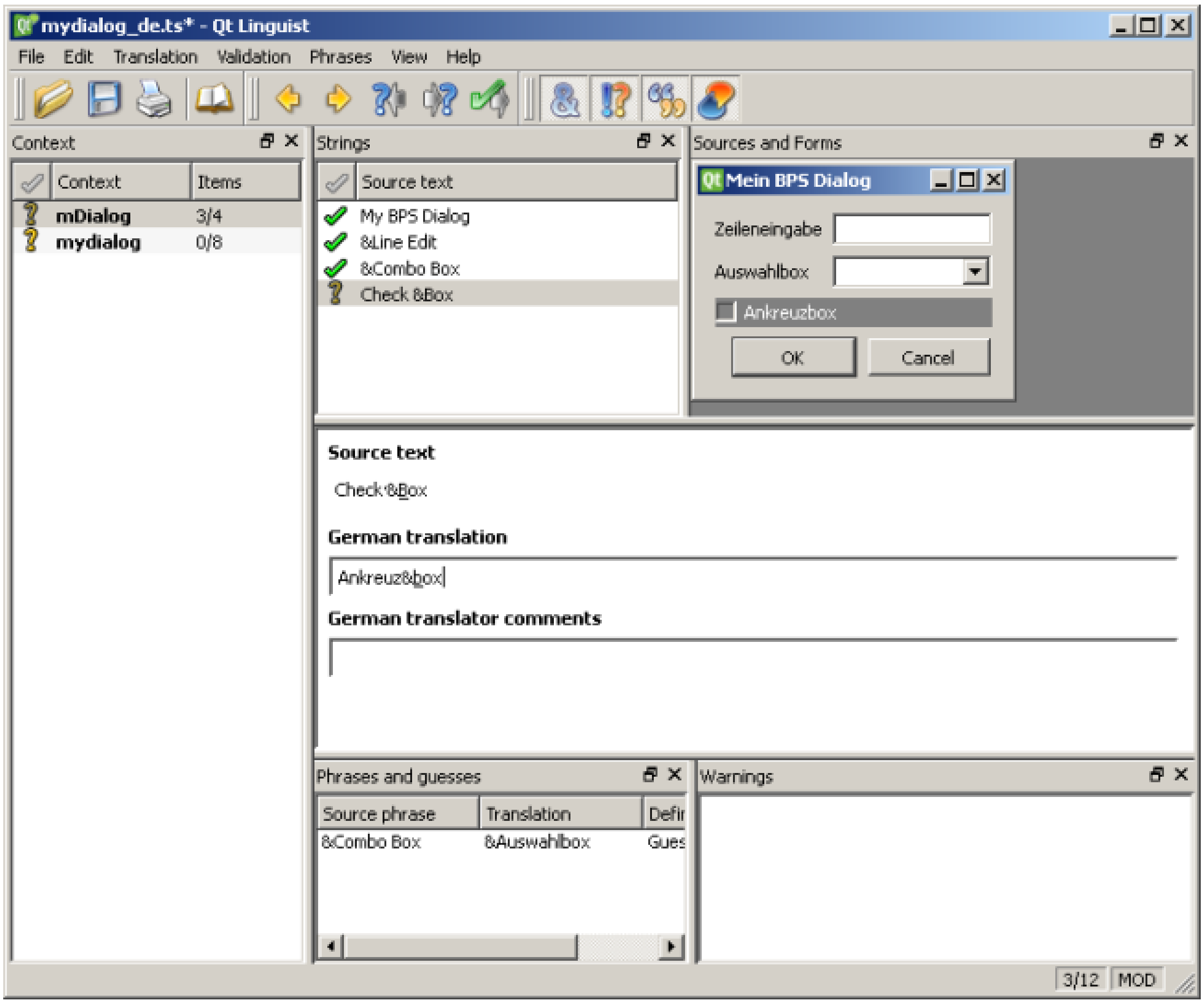
```
buttons.accepted.connect(okPressed);

// initialize inputs
lnedit.text = qsTr('sample text');
cmbbox.addItems([qsTr('red'), qsTr('green'), qsTr('blue')]);

// execute
dialog.show();
bps.gui.exec();
```

Now create the TS file:

```
lupdate mydialog.js mydialog.ui -ts mydialog_de.ts
```

And start the *Qt Linguist* on the TS file:

```
linguist mydialog_de.ts
```



	Context	Items
?	mDialog	3/4
?	mydialog	0/8

	Source text
✓	My BPS Dialog
✓	&Line Edit
✓	&Combo Box
?	Check &Box

Source text

Check &Box

German translation

Ankreuz&box

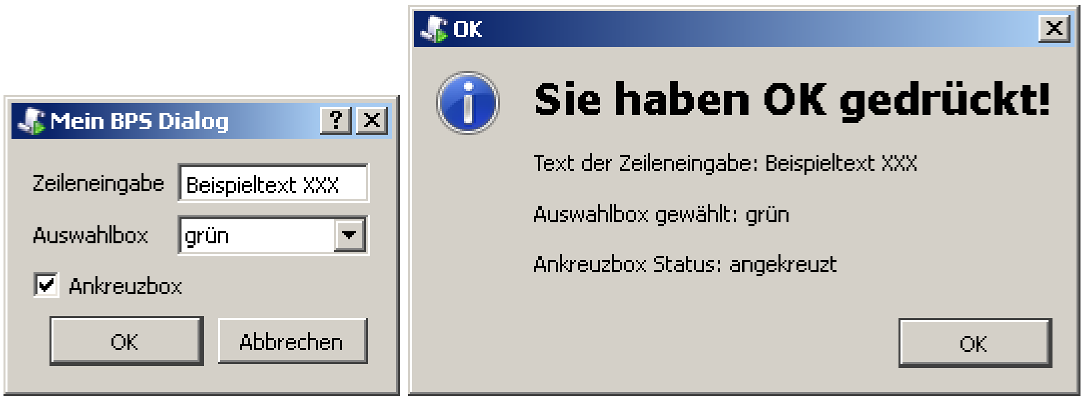
German translator comments

Source phrase	Translation	Defi
&Combo Box	&Auswahlbox	Gues

Basically all you need to do is enter the translation for every text and confirm it with the green checkmark. Do it for both contexts, `mDialog` and `mydialog`. There are additional helpfull features in *Qt Linguist* such as *Phrase Books*, validation functions and comments for translators; you might want to use them when you have to handle many translations.

When all is translated, save by clicking the blue diskette symbol and generate the `QM` file with *File - Release*.

Finally, switch the windows display language to the target language as explained before. You need to log off and on again to complete the switch. Then test your application:



Zeileneingabe: Beispieltext XXX
Auswahlbox: grün
☑ Ankreuzbox

Sie haben OK gedrückt!

Text der Zeileneingabe: Beispieltext XXX

Auswahlbox gewählt: grün

Ankreuzbox Status: angekreuzt

In case the application texts change or the translation needs corrections, just repeat the process. When the TS file exists already `lupdate` will do the best to reuse the existing translations.

Extension Translation

Use `qsTranslate()` with a context name unique to the plugin, instead of `qsTr()`. Otherwise there will be conflicts when using multiple extensions, because the script name of all extensions is `__init__.js`.

Usually BPS tries to load a translation file in the same directory and where the name is the base script file name followed by an underscore and the language code, for example `myscript.js` -> `myscript_de.qm`. However the underscore is omitted for plugin script files to reduce the number of consecutive underscores, so BPS expects it to be `__init__.js` -> `__init__de.qm`.

Report Translation

To translate reports do the following:

- All fixed texts in labels must now get set by the report script.
- Move the report script to an extension. Basically in the report itself just something like this is left:

  ```
  loadExtension('acme.reports.invoice');
  acme.reports.invoice.runReport(report);
  ```

- Finally translate the extension.

A modified approach is to only move the texts to variables in the extension, but keep the rest of the script in the report itself.