PREPARING WEB SITES *FOR STREAMLINED* LOCALIZATION

Lernout & Hauspie™

Lernout & Hauspie ™

Preparing Web Sites for Streamlined Localization

This L&H White Paper is designed for any company, organization, or institution that anticipates having its web site translated by Lernout & Hauspie. Its intended audience includes technology officers, project managers, site designers, content providers, and others who are involved in creating and maintaining web sites.

Converting a web site into another language is not simply a matter of exchanging one language for another. In addition to ensuring that translations match local customs and practices, L&H takes steps to see that words in each translated language fit cleanly into a site's design and match its graphics. By avoiding from the outset conditions that cause time-intensive translations, web site designers, content providers, and programmers can minimize the expenditure involved in localizing web sites.

Quick Start: Four Easy Steps

Before providing detailed information about preparing your web site for L&H localization, we offer four quick steps that will facilitate multilingual translation of your site.

1. Write clear, concise, and unambiguous sentences.
2. To the greatest extent possible, separate text from graphics.
3. Provide original graphics in all places where graphics contain text.
4. Separate strings from code.

Three Problem Areas in Localizing Web Sites

Three areas account for the majority of problems encountered when localizing web sites, and the remainder of this L&H White Paper addresses each area separately. These areas include:

- **Graphics -** problems caused by unorganized graphics files, flat GIF files, and other graphic elements.
- **Code Structure -** errors and delays caused by the need to search for strings buried within code.
- **Tools -** challenges caused by Microsoft® FrontPage and Web browsers.

GRAPHICS

If the following steps are taken with graphics, L&H's ability to localize your site with speed, accuracy, and cost-efficiency will greatly increase.

- **Indicate Which Files with Graphics Contain Text**

  When submitting a site for localization, either indicate by file name or provide a list of those graphics that have text in need of translation. Without such a list or indicator, time is needlessly spent opening and closing graphics files to check for text.

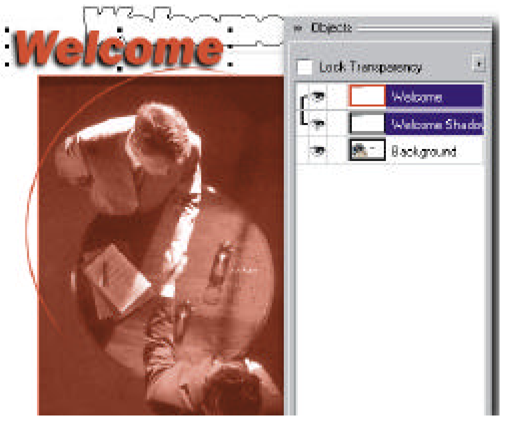
- **Leave Text Outside Graphics**

  To the greatest extent possible, when designing your site, simply leave text outside of graphics so that images do not have to be reprocessed.

- **Provide Source Graphics with Layers and Fonts**

  Flat GIF files contain one essentially inseparable image, even if that image contains art, text, shadow, borders, and so forth. Original picture files, with layers for each component, enable translators to remove text, translate, resize shadows or borders, and reinsert the translated text into the layered file. If special effects like shadow or stretching are applied to the text after the layers have been merged, this should be documented. One good idea is to create one additional layer, which contains only the instruction text.

Figure 1:

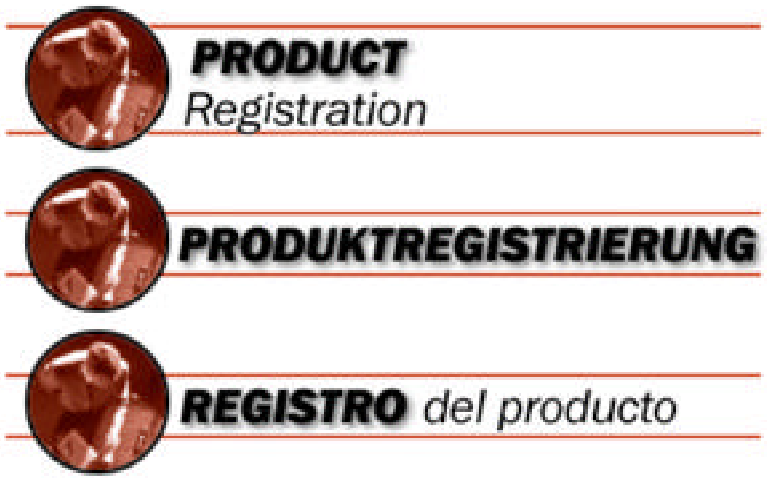


Provide text in separate layers to make it easier for the localizer to translate, move, and resize text easily, without having to guess the font that has been used.

- **Leave Sufficient Space for Translations**

  Site designers often fit frames tightly around text, but often more space will be needed when words are translated into another language. For instance, the word "Corporation" is much shorter than its German equivalent, "Aktiengesellschaft." The best solution to this problem is to provide extra white space so that L&H localizers do not have to redraw frames.

Figure 2:



- **Avoid Using Special Fonts**

  There are thousands of fonts available, which are not always easy to identify or match, especially in flat GIF files. Localization is faster and easier when standard, recognizable fonts are used. When special fonts are necessary, localization is easier if font lists, or better still, the fonts themselves are provided.

CODE STRUCTURE

The following steps will help you prepare code so that your site can be translated in a highly cost- and time-efficient manner.

- **Separate Strings from Code**

  You can eliminate the vast majority of localization problems that arise from code structure simply by separating strings from code. Separating strings from code prevents localizers from mistakenly translating code, variable names, and so forth. The easiest way to separate strings is by placing them in:

  1. One place within a file
  2. A separate include file
  3. A database

  If strings are separated within a file, a comment line can help your localizer complete the translation more quickly; e.g., “REM Localizable text ends here.”

  These principles also apply to source files written in any programming language. Translatable strings should be assigned variables in code, with the strings themselves residing in external resource files.

Problem:

Figure 3:

```
If Request("aspState") = "checkerror" Then
      REM===================================
      REM page is calling itself to check errors
      errorcount = 0
      errorstring = "”

      REM===========================================
      REM required form inputs: first name, last name
      REM and email
      REM

      If Request("FirstName") = "" Then
            errorcount = errorcount + 1
            errorstring = "<strong>Your first
      name</strong><br>”
      End If
```

Solution:

Figure 4:

```
L_FirstName_Missing="Your first name”
.
.
REM ===== Localizable text ends here. Do not touch
the rest of the file! =====
.
.
If Request("aspState") = "checkerror" Then
      REM====================================
      REM page is calling itself to check errors

      errorcount = 0
      errorstring = "”

      REM============================================
      REM required form inputs: first name, last name
      REM and email
      REM

      If Request("FirstName") = "" Then
            errorcount = errorcount + 1
            errorstring = "<strong>" &
      L_FirstName_Missing & "</strong><br>”
      End If
```

- **Storing content in a database**

In many cases the Web site is designed to publish content of an existing database, or a dedicated database is created for the content to be published. This effectively separates the content from the design. As a result it is easier to translate the user interface.

Often it is not appropriate to give the translators direct access to the database. Instead the content will have to be exported into a suitable exchange format such as HTML, XML or Excel. This also makes it possible to use existing translation tools to translate the file.

Additionally, content management systems can be used to keep track of what files and what database content has been modified and has to be translated. Alternatively a file exchange format can be developed to check out files from the content management system, and check in them again.

Control the use of Variables

Unlike developers who read code according to variables, localizers simply translate sequentially. When a variable is separated by several lines of code or different variables are used for one string, translation inconsistencies can arise.

The best solution is to use only one variable for each string in need of translation. In the case of variable and code separations, comments to translators that indicate a first use will help maintain consistent translations.

Use Variable Names that are not Words

Variable names are often common words that risk translation. For instance, consider the line <select name="salutation">. Translators may not know if “salutation” is a name used internally by the script, a database query statement, or an element to be translated. If code and strings are not separated, again add explanatory comments in the source file or use variable names that are not words, such as <select name="salutationfld">. This solution works when developers assign variables. It does not work with for example SQL statements, which use plain English.

Avoid Breaking Sentences

Broken sentences are difficult to read and translate. Sometimes sentences are unnecessarily broken just to give code a clean wrap. If text must be broken, look for punctuation—periods, commas, colons, semicolons, etc.—as a good place to insert a break.

Figure 5:

```
errorstring = "We encountered a
problem attempting
to process your "
errorstring = errorstring &
"registration. Please
try again later."
CIASession("ErrorString") = errstring
```

The translator may easily omit the space at the end of the line.

- **Write Locale-Independent Code**

  Locale settings can affect localization, and no translation process exists to account for differences in server settings. For instance, differences in European and U.S. dating systems cannot be accommodated by translators. Instead, limit applications to one locale format or write separate processing functions independent of the translation process. This problem applies only to functions that are part of the user interface.

- **Character Sets**

  The developer has to decide if the pages are to be translated using Unicode or a DBCS code page (Double-Byte Character Set). If the translated page has to be displayed using a particular character set, it should be indicated in the HTML file. For example to select the most common Japanese DBCS code page (Shift-JIS), the following META tag can be included:

```
<meta http-equiv="content-type" content="text/html; charset=shift_jis">.
```

  Local developers might forget these settings, because they are the default ones in the local operating system and browser, but not necessary in the international ones.

  Although Japanese readers are used to reading computer text from left to right (instead of top to bottom), Arabic users read the screen from right to left. This should be considered when designing the screen layout.

- **Avoid Breaking HTML Sentences With CRs**

  Some translation tools used in the localization process break single HTML sentences with a CR into two segments. To eliminate resulting problems, simply avoid breaking HTML sentences with CRs.

Tools

When preparing web sites for localization, issues arising from the use of the following applications should be considered.

Microsoft® Frontpage

Indented code is easier to read and results in fewer localization errors due to accidentally deleted or overwritten lines. Though a good tool, FrontPage changes the indentation of code and should be avoided when revising code. In addition, place tags on separate lines, again to avoid overwriting or deletion errors.

Web Browsers

To make the most of specific browsers, site designers will sometimes write optimized pages. While standard code for any browser may not deliver the most vibrant pages, it makes the task of localization easier and quicker, which holds down costs.

How Lernout & Hauspie Can Help You

With worldwide translation and documentation services, L&H can help you do business in any market around the globe.
Our technology and top-line people enable you to communicate with employees and customers anywhere.

L&H offers the following services

- Software localization
- Website building and globalization
- Custom Translation
- Multilingual Technical Writing
- Multilingual Electronic Publishing in SGML/XML
- Terminology Management

with specialties in nearly every industry.

For More Information on L&H's Services

Visit our web site at:
http://www.lhsl.com/

E-mail us at:
sales-services@lhsl.com

Telephone us at:
781-203-5133