Web Site Localisation and Internationalisation: a Case Study

Noelia Corte Fernández

September 2000

Submitted in partial fulfilment of the requirements for the degree of Master's in Electronic Publishing
Abstract

The report is divided in two sections. The first part is an overview of the processes involved in the internationalisation and localisation of web sites into other languages. It deals with the technical, cultural, linguistic and design issues that should be considered when localising a site, and offers a list of recommendations to optimise the process. It also includes a review of some of the most popular localisation software packages on the market. Finally it suggests a set of guidelines to internationalise and localise an English web site in the most effective manner.

The second part is a case study on the localisation of an English site for the Spanish market. All the theoretical assumptions made in the first part are analysed and empirically tested on a prototype model. The main localisation challenges are underlined and an internationalisation and localisation strategy is suggested to streamline the process and help with future projects in other languages. This strategy was developed taking into account the particular features and technical requirements of the original site.
# TABLE OF CONTENTS

List of Figures........................................................................................................................................... 6  
Acknowledgements....................................................................................................................................... 7  

## PART ONE: GUIDELINES FOR THE LOCALISATION OF ENGLISH WEB SITES........ 8  
1. Introduction............................................................................................................................................... 9  
   1.1. Internationalisation......................................................................................................................... 9  
   1.2. Localisation....................................................................................................................................... 10  
   1.3. Globalisation.................................................................................................................................... 10  
2. Aims and Objectives................................................................................................................................... 10  
3. Technical Issues....................................................................................................................................... 10  
   3.1 Character Encoding......................................................................................................................... 10  
      3.1.1. ASCII, ANSI and ISO............................................................................................................... 10  
      3.1.2. DBCS......................................................................................................................................... 11  
      3.1.3. UNICODE............................................................................................................................... 11  
      3.1.4. UCS or Transformation Formats UTFs.................................................................................. 11  
      3.1.5. Charset Parameter, in http protocol..................................................................................... 12  
      3.1.6. The Lang attribute.................................................................................................................... 12  
      3.1.7. Special Characters.................................................................................................................... 12  
      3.1.8. URIs encoding.......................................................................................................................... 13  
   3.2. Directional Language Requirement............................................................................................... 13  
   3.3. Sorting Order.................................................................................................................................... 13  
   3.4. Localising HTML............................................................................................................................ 14  
   3.5. Interaction between character encoding and mark up languages.................................................. 14  
   3.6. Localising code............................................................................................................................... 15  
      3.6.1. Functionality vs. content.......................................................................................................... 15  
      3.6.2. Text expansion............................................................................................................................ 16  
      3.6.3. Composite strings and error messages..................................................................................... 16  
      3.6.4. Target Language specific content............................................................................................ 17  
      3.6.5. Applets and Active X components......................................................................................... 17  
      3.6.6. Forms........................................................................................................................................... 17  
      3.6.7. Hot keys...................................................................................................................................... 17  
   3.7. Multimedia....................................................................................................................................... 18  
   3.8. Search Engines............................................................................................................................... 18  
   3.9. Identifying international users.......................................................................................................... 18  
   3.10. Recommendations......................................................................................................................... 18  

4. Cultural and Linguistic Issues.................................................................................................................. 18
3. Methodology Applied

3.1. Assessing the market and analysing the English site

3.2. Identifying the different elements

3.2.1. User Interface

3.2.1.1. Graphics

3.2.1.2. Style Sheets and tables

3.2.1.3. Navigation

3.2.2. Code

3.2.2.1. Character Encoding

3.2.2.2. HTML

3.2.2.3. Code

3.2.2.4. Database Connectivity

3.2.2.5. Other technical issues

3.2.2.6. Hosting

3.2.3. Content

4. Integrating all components

5. Testing

6. Suggestions

6.1. Layout

6.2. Character encoding and scripts

6.3. Content

6.4. Updates and maintenance

7. Recommended Software

7.1. TRADOS

7.2. WebPlexer

8. Testing and Quality Assurance

9. Conclusions

References

Bibliography

Appendix A

Appendix B

Appendix C

Appendix D

Appendix E

Appendix F
LIST OF FIGURES

Figure 1.1.: Web use in the world from Designing Web Usability, Jacob Nielsen……………….9
Figure 3.1.: HTML Localisation from “Localising Web Sites and Content on the Internet”.
http://www.itp.ie, August 1999…………………………………………………………………………14
Figure 3.1.1. : Case Study’s site Staging Server (July 2000)…………………………………… 31
Figure 3.1.2.: UK site file structure……………………………………………………………………. 32
Figure 3.1.3.: Spanish site file structure………………………………………………………………. 33
Figure 3.2.2.4-1: Prototype database table……………………………………………………………. 36
Figure 3.2.2.4-2: ODBC Data Source Administrator……………………………………………… 37
Figure 6.1.: Suggested graphic storage……………………………………………………………………. 41
Figure 7.1.: WebPlexer System from http://www.lai.com………………………………………… 43
ACKNOWLEDGEMENTS

I am very grateful to my supervisor at University Susan Jones for checking all my drafts and helping me with all the technical problems I encountered during the site development. Also to Eloise Grey in the host organisation for allowing me to carry out this project and all her support and encouragement.

I would also like to thank Georgina Wood for reading all the drafts correcting my grammatical and spelling mistakes.

A special and big thank you to Tim Wood for all his support and help during the Masters course both at a technical and a personal level.

Finally I would like to dedicate this project to my parents Isidro and Maria Luisa for always supporting and helping me whatever I intend to do.
PART ONE: GUIDELINES FOR THE LOCALISATION OF ENGLISH WEB SITES
1. INTRODUCTION

Since its conception English has been the predominant language in the World Wide Web. This is because the Web initially was the exclusive realm of the U.S. and UK markets and it was not until the late nineties that the Internet started to expand in non-English speaking countries. Currently the number of users in those countries is steadily growing. Even when English is considered a lingua franca in the Internet world and many Internet users can at least understand it, it is obvious that a site in their native language will be more appealing and easy to use. Above all to get information and buy products.

According to the E-Marketer and Forrester Research Inc. (1), by 2002 more than half of the Internet users will have a non English native tongue and by 2003 most web content will be in a language other than English. There are several reliable studies in the market assessing the language issue on the Web. The Inktomi and the NEC Research Institute, 31st Jan 2000, (2) found that there are more than one billion unique pages on the Internet, 86% of which are in English. In contrast around 35% of Internet users do not speak English, and this figure is increasing. According to a study by GlobalReach in 1999, the number of non-English speaking users has grown from less than 10% to 50% over the past 4 years.

The largest markets for localisation are France, Germany and Japan. But most companies usually localise their products into the so called FIGS (French, Italian, German and Spanish).

Customers tend to spend more time and are more keen to buy products on sites that are in their native language. Moreover, customer care costs can be drastically reduced if the instructions are in the users native language. Companies want to make the most of the global market, however before they can start doing business in another country the language barriers must be overcome. Here is were the issues of internationalisation (i18n), globalisation and localisation (l10n) come into play.

1.1. Internationalisation

Internationalisation is the first step towards full localisation of a Web site. It consists of making the site locale independent in order to adapt any locale information such as dates, times, numbers, currencies, etc. to any language specific locale.

![Distribution of web users in 1999. Source: NUA Internet Surveys.](image)

Fig. 1.1. Extracted from Designing Web Usability by Jacob Nielsen (3)
The most important factor is to separate the translatable elements (what the user sees) from the source code (back-end functionality).

1.2. Localisation
According to the Microsoft Computer Dictionary, localisation is “the process of altering a program so that it is appropriate for the area in which it is used” (4). This has been a common activity in the software industry for years, but is relatively new to the Web. The same definition can be applied to web sites with a slight shift of priorities. While in software programs the emphasis is in functionality and language quality is secondary (due to the fact that updating a product is expensive and time-consuming and users are more concerned with its functionality) in web sites, however, the language quality is as fundamentally important as its functionality, since content constitutes the core of the final product with a surrounding functionality needed to access that content (5). This involves more than simply translating content, it also involves adaptation to the local culture and, in certain cases, site re-designing.

1.3. Globalisation
Globalisation covers both internationalisation and localisation. It is the process of designing a web site in such a way that it can be used in different countries with minimal changes. It is a marketing oriented concept, rather than a technical one.

2. AIMS AND OBJECTIVES
This report will be divided into two parts, the first part will be an overview of the problems and issues posed by the internationalisation and localisation of web sites. The aspects to study will be technical, cultural and linguistic, and design. In each case a series of recommendations to optimise the process will be offered. The second part will be a localisation case study. The theoretical assumptions made in the first part will be practically assessed and adapted to the particular characteristics of the studied site.

3. TECHNICAL ISSUES
Although software localisation is a well-established and mature activity in the IT industry, the issue of Web site localisation is quite recent. For this reason most localisation tools are still in the process of development. Current web sites are usually multi-tier structures with static and dynamic content supported by a back-end database and involving several scripting languages. These characteristics increase the complexity of the localisation.

The WWW3 Consortium has published two RFCs dealing with the localisation of HTML, RFC 1766 on the tags used for the identification of languages in 1995 and RFC2070 on the internationalisation of the Hypertext Markup Languages in 1997.

3.1. Character encoding
A computer uses codes in its memory to represent characters so that text can be stored and reproduced. These character encodings are used to represent the character’s location in a particular place in the text without indicating anything about its external characteristics such as colour or size.

3.1.1 ASCII, ANSI and ISO
Early computers structured their memory into units equivalent to one character, in different computers the units had different sizes. This caused compatibility problems since different computers stored different number of code points. Different characters were stored at the same code point making impossible any intercommunication between computers. That was the reason for the development of the American Standard Code for Information Interchange (ASCII), a 7-bit byte that covered 128 different characters. This was enough to cover the English language but clearly insufficient for other languages with diacritics or different alphabetical characters.

In order to overcome some of the confusion caused by the different character-set standards developed by programmers to get round the limitations imposed by the ASCII system, the American National Standards Institute (ANSI) defined a new character set that became widely
The ANSI standard consists of an 8-bit byte that made 256 code positions possible. This allowed the creation of alternative sets of code pages for different languages in addition to the cp 437 ("US English") (6). Since ANSI is a member of the International Standards Organization (ISO), these sets were also specified as international standards in the series ISO 8859. The ISO-8859-1 (Latin-1) standard covers most of the Western European languages plus Swahili and Hawaiian.

3.1.2. DBCS
While a unique character in English can be expressed by a single byte (a number from one to 256), there are so many unique Chinese or Japanese characters that the computer needs two bytes (one to 65,536) to find one. These characters are stored in several tables, the first byte tells the operating system which table to look in, while the second tells it the location of the specific character on the table. This constitutes the Double Byte Character Set (DBCS) used to represent Asian languages such as Japanese, Chinese and Korean.

One of the problems faced when using DBCS is that, although most servers now handle double-byte encoding, the interfaces to back-end systems and, sometimes, the back-end systems themselves are still often written using single-byte programming. This makes them return corrupted results when they encounter double-byte characters.

Chinese, Japanese, Korean and Thai (although the latter can be represented with a single byte encoding) do not have spaces between characters. This can cause a problem when using line wrapping and breaks will probably have to be manually inserted.

3.1.3. UNICODE
Although the 8-bit byte provided more code points, it was still not enough to cover all languages. 16-bit bytes were created but in four different codes that did not allow interoperability between computers.

Unicode offers a unique number for every character, it provides platform and language independent building blocks that can be used to create content and applications in a variety of languages – and even in multiple languages, surpassing the need for double byte enablers. It reduces dependency on OS for language and font support. For example, without Unicode, in order to view a page in Chinese, it would be necessary either to have Chinese OS, the fonts or some kind of special software.

In order to maintain compatibility with legacy character sets, many characters have been added, this has reduced Unicode’s available space. Therefore an extension mechanism was needed. The standard mechanism uses pairs of Unicode values called surrogates to address over 1,000,000 possible extra values (7). In addition to this, some systems could not easily handle 16-bit units in processing and required a form of Unicode that could be handled in 8-bit bytes. All these different requirements resulted in the creation of different forms of Unicode.

3.1.4. UCS or Transformation Formats UTFs
HTML uses a more complete character set called the Universal Character Set (UCS). This was defined in ISO10646 and is equivalent character-by-character to Unicode. Both of these standards are regularly updated with new characters.

UCS-2: uses a two-octet per character encoding
UCS-4: uses a four-octet per Character encoding
Both UCS-2 and UCS-4 address the first 64k characters of UCS. This covers the Basic Multilingual Plan (BMP), however they were not compatible with many applications and protocols. For this reason the UTF standard was developed.

UTF is a subset of Unicode. Every 16-bit Unicode character is encoded as a sequence of one, two or 8-bit bytes, depending on the character. It remains an ASCII-compatible encoding method, as long as no more than 127 characters are present. Therefore an HTML document encoded as UTF-8 can remain a normal single-byte ASCII/ISO-8859-1 document (8). If it contains Unicode characters above 255 they are referred to by ampersand entities such as &eacute; for the character “é”. There are three different versions, UTF-8, 16 and 32. The most widely used in the
Web is UTF-8, since this is a variable length encoding, that leads to different compression and expansion factors for different classes of characters. ASCII is represented as 1 byte, characters up to U+17FF (Arabic) are represented with two bytes and the rest of the BMP are represented as three bytes. Characters outside the BMP, but representable with UTF-16, need four bytes (9).

Programming with UTF-8 and UTF-16 has clear advantages over other mixed-width character encodings. Firstly, it prevents overlapping, as the end of one sequence can never be the same as the start of another sequence, a false match will never be returned on code points. This is one of the biggest problems with common multi-byte encodings like Shift-JIS.

UTF-2 and UTF-16 are more compact than UTF-32 in storage terms, but this is not a big problem since conversion between different UTFs is very fast and it does not require table lookups.

3.1.5. Charset parameter, in http protocol.
The “charset” parameter identifies a Character encoding, i.e., a method of converting a sequence of bytes into a sequence of characters. HTTP 1.1 establishes that the default for HTML is ISO-8859-1, but in practice unlabeled documents are in all kinds of encodings. For this reason browsers have adopted a practice of not switching to ISO-8859-1, but switching to the reader’s preferred encoding, on the assumption that most readers read documents in their own language. It is better to always label HTML documents explicitly (10), if the user has to manually adjust the browser’s encoding options to view the page correctly, some forms may not work properly.

It is necessary to know which Character encoding(s) the server will be able to handle upon submission. Such an indication is provided by the ACCEPT-CHARSET attribute of the INPUT and TEXTAREA elements on the HTTP Accept-Charset header. When the request is sent to the server a Content negotiation takes place. The browser sends an http_accept_language request and the server uses a type-map file to find the correct file.

3.1.6. The Lang Attribute
This attribute can be used to control the rendering of a marked up document in various ways:

1. Glyph disambiguation, when the Character encoding is not enough to resolve a particular glyph;
2. Quotation marks;
3. Hyphenation, ligatures and spacing;
4. Assisting search engines and speech synthesisers;
5. Assisting spell checkers and grammar checkers.

It basically renders content more meaningfully.

RFC1766 defines and explains the language codes that must be used in HTML documents. Language codes consist of a primary code and a possibly empty series of subcodes:

language-code = primary-code (“-” subcode) * (11)

For example:
*en* = English
*en-US*: The US version of English

Language mark-up is useful as content mark-up for purposes such as classification and searching. Almost all HTML elements admit the LANG attribute. The value of the LANG attribute overrides the value specified in the HTTP Content-Language header.

3.1.7. Special Characters
HTML standards allow for named and numbered entities (see appendix A for list of named characters in ISO-8859-1). The use of named entities is not recommended since they do not work consistently in all browsers, operating systems or fonts. They are only advised as shortcuts for people who do not have a keyboard capable of entering such characters.
Special attention must be paid to those characters required for markups, such as <, > or &. They must always be coded by reference when not used as a tag.

3.1.8. URIs encoding
The Universal Resource Identifier is a superset of the Universal Resource Locator (URL), created in anticipation of the different resource naming conventions that are currently being developed. For the time being, the URIs are equivalent to URLs.

The problem with URIs is that they need to be readable by both the machine and the user. Since the English language can be represented by ASCII characters, this does not cause any trouble, but sometimes the URIs contain information that is not in ASCII.

URIs contain all kind of information from all kinds of protocols and formats. The official URI syntax (RFC2396) only allows a subset of ASCII, about 60 characters. Other characters can be encoded using the % HH escaping (% followed by two hexadecimal digits). (12)

UTF-8 is the recommended encoding for URIs, but its use is still very irregular with non Latin characters.

3.2. Directional language requirement.
\[ \text{dir} = \text{LTR} \mid \text{RTL} \]

The Dir (Unicode Bidirectional Algorithm) tag indicates the direction in which the text is read. This is used to cope with languages such as Arabic or Hebrew, which run from right to left. Sometimes script running left-to-right is also needed on-screen with the right-to-left script and the software must be able to cope with this too. This is called Bi-Directional (Bi-Di) language requirement. The BIDI algorithm is only necessary when it is necessary to display both right-to-left characters. (13)

Unicode can also assign directionality to characters and can be used to define an algorithm to determine the proper directionality of text. It is not advisable to use the LANG attribute to control directionality.

There can be conflicts if the DIR attribute is used together with the Unicode formatting characters. The best method to deal with bi-directionality issues is the markup method as it ensures better structural integrity of the document and causes less problems when editing bi-directional HTML text.

The BDO element allows authors to turn off the bi-directional algorithm for selected fragments of text. For example:

```
<!ELEMENT BDO - - (\%inline;)* - - 118N BiDi over-ride -- >
<!ATTLIST BDO
  \%coreattrs; - - id, class, style, title - -
  \%LanguageCode: \#IMPLIED - - language code - -
  dir (ltr | rtl) \#REQUIRED -- directionality - -
>
(Example extracted from http://www.w3.org/TR/REC-html40/struct/dirlang.html, "Language information and text direction")
```

(See Section 5.6 for the effect of style sheets on bidirectionality.)

3.3. Sorting order
Different languages use different sequence algorithms for sorting information. Alphabetically ordered lists will have different orders depending on the language. In fact, many Asian cultures sort their characters by their brush stroke. The best solution is to get the localisation engineers to change the sort order as they localise the code.

At the back end, sort orders in SQL are determined during SQL server installation. Any changes will need a rebuild of the database.

A possible way to automate the sorting of data is to parse it through a control that uses the correct sorting mechanism for the language required. This will require language specification in the control. If this is done at the client end, it may not work since the user’s OS and browser may be different to what would be expected in a specific market.

The fastest way of sorting lists is by manual interaction using a word processor.

3.4. Localising HTML
There are three main types of site in the web at present, each of them will demand different localisation approaches:

1. Sites created, edited and saved as Raw HTML;
2. Sites created, edited and saved as HTML using an application;
3. HTML generated dynamically as the content is accessed. (14)

The first type is the simplest to localise, whilst the last has the highest level of complexity. Depending on the type of site the content will be localised at different stages as shown in Fig. 1

![HTML Localisation processes](image)

Figure 3.1. HTML Localisation processes (14)

The Content Capture Point is where it is most efficient to extract the Content from its proprietary form and to re-insert it after translation. It is important to keep the localisable material in a format other than code as translators are not usually computer programmers (see Section 3 for more information about translating in HTML).

Once the HTML pages have been translated and proof read, the links need to be adjusted. It is advisable to use absolute links since they avoid confusion during browsing. When the site has
been simultaneously localised in several languages, it is essential to carefully check all the hyperlinks since they will point to different resource graphics or files.

### 3.5 Interaction between character encoding and mark up languages

Before mark up is applied, the underlying plain text should be properly encoded and have its own meaning, otherwise incompatibility problems will arise.

There are some considerations to take into account when using Unicode with mark up languages. Unicode contains additional characters that assure compatibility with older Character encoding, and characters with control-like function included to provide unambiguous interpretation of plain text. Markup provides essentially similar features to those provided by format characters in Unicode (15):

1. **Linearity vs. hierarchy**
   Encoded text is just a sequence of characters, markup defines a structure. Operations that are easy to perform on hierarchies are often difficult to perform on linear sequences. An adequate separation of Character encoding and markup functionality will achieve a simple and more powerful architecture.

2. **Overlap of control code and markup semantics**
   Many of the Unicode control characters have direct equivalents in particular markup languages. If both are present in the same text, the duplicity has to be resolved when applying the markup.

3. **Mark up and styling**
   Markup deals with the logical structure of the text or data, styling is used to present the information in various ways. Some character codes encode a styled character, freezing the styling information and preventing any change.

4. **Local content**
   Where local or point like context is required, markup is not very efficient, in fact to add markup in the middle of words can make search or sort operations more difficult. For these cases it is advisable to express the information as character codes.

### 3.6 Localising code

Localising HTML is relatively easy, above all with the use of specialised software that allows the “locking” of the HTML tags and the use of translation memory software. The main problem is to access localisable content within scripts, i.e., JavaScript, Visual Basic and applet parameter tags, which is not easily parseable by mechanical means.

There are different issues to consider:

#### 3.6.1. Functionality vs. content

It is essential to separate UI, content and code elements. Within the code there is a further distinction, back-end functionality and the code that governs the UI. The back-end functionality of the site should be identical for all languages, the only items affected by localisation should be those visible to the user. The different components should be placed in resource files and the same resource identifiers should be used throughout the project to facilitate updates. The description of the UI and the code that implements it must be kept together and not be embedded in the executable code base. (16)

Establishing a code base as a common ground for the localisation process will improve efficiency and reduce time and costs. When a string is used in multiple multilingual contexts, copies should be made to avoid errors.

There are different methods to achieve this:

a) **Header files**
   They are used to define parts of a program that may be used in various places in the code and may be modified often. The translator will only need to modify the translatable strings in the
header file without dealing with the program’s code. The main problem with this approach is the lack of context, which can cause problems when translating.

b) Resource files
On these files all the localisable text is isolated and can be compiled into dynamic link libraries (DLL) simplifying the management of multiple language localisation projects. There is only one executable file that pulls localised resources from the DLL file. To change the language, it is only necessary to change the DLL file. (17)

Resource files can be implemented in three different ways:

i) Language dependent binary with built in resources.
Ex. App_Eng.exe
   App_Ger.exe
   App_Fre.exe

ii) One core binary with one international resource DLL
Ex. app.exe
    intl_res.dll

iii) One core binary with multiple international resource DLLs
Ex. app.exe
    Eng_res.dll
    Ger_res.dll
    Fre_res.dll   (18)

3.6.2. Text expansion
Since most languages are longer than English, usually by 30%, this can cause a problem with dialog, pop-up windows, menus and dropdown lists. The problem is even worse when it comes to localising string resources since these are limited to 255 bytes (for 16-bit operating systems). Although it is possible to create longer strings, this will require recoding the resources.

There are specialised software packages to deal with the resizing of menus, buttons, dialogs, etc. e.g. Resource Workshop for Windows or ResEdit for Macintosh. These operations must be checked at run time to make sure all the elements are correctly displayed.

3.6.3. Composite strings and error messages.
A composite string is text dynamically generated and presented to the user in sentence form. This method is used to avoid programming repetitive messages and to save disk space. However this method is to be avoided in sites that are going to be localised. Different languages have different word order and particular grammatical rules regarding gender, articles, plurals, etc. The ideal solution would be either to have multiple error messages translated separately or to find different ways of presenting the same information (5). For example:

The String 1 you have requested is not available on this String 2

String 1 = "page, resource"
String 2 = "server, drive, location"

This construction will not work in Spanish for example, "page" requires a feminine article while resource is masculine,
La página
El recurso
The same happens with “String 2”.

Word order can also be a problem in certain languages such as German that places the main verb at the end of the sentence. Ideogram languages such as Chinese or Japanese follow grammatical rules radically different to those of the English language. When Arabic or Hebrew languages are used it must be taken into account the complexities of right-to-left text direction.
A similar problem arises with plurals:

E.g.

```html
<%
  If %d >1, then
  d% program%s searched"
else
  "d% program searched"
%>
```

The output will be either

1 program searched
or
3 programs searched.

But plurals are not always built with an “s”, for example, in German the plural of “Program” is “Programme”.

3.6.4. Target Language specific content.
- Date format (long and short);
- Time format;
- Money and anything related to it: currency, taxes, prices, etc.;
- Number formats;
- Address formats;
- Font names, sizes and decoration;
- Paper sizes;
- Measuring units;
- Case differences;
- Character order (A-Z a-z does not cover all alphabet letters in certain languages);
- Character format (not all languages consider that a word is any succession of characters between two spaces).

Some of these items are usually hard coded making it very difficult for the localiser to translate them. When possible it is recommended that the system settings or an automated function are used to present the information to the user. (5) The software engine needs to be adapted to handle this information correctly according to the rules and standards of the target market. See appendix B for a list of the specific locale information for Spanish.

Hard-coded characters should be replaced with characters that can be easily redefined if there is a conflict. (17)

3.6.5. Applets and Active X components

Java Applets and Active X are usually treated as independent applications and therefore localised in the same manner as traditional software. Usually the amount of content accessed via embedded applet tags is quite small, with the majority of content being accessed and displayed through the applet itself.

3.6.6. Forms

Forms are the main method available to web sites to get information about the users. They play a key role in e-business therefore it is very important to assure that they work well and that the users see all the information in their native language.

If the data is going to be posted in a specific character format, the right encoding should be specified in the metatag of the page, e.g.

```html
<META HTTP-EQUIV=Content-Type content="text/html; charset=UTF-8">  
```

Otherwise the characters in the form elements will not get converted to UTF-8, they will remain in the standard encoding set in the browser (8).
Apart from localising forms at the browser end, the user will also need to know which **Character encoding** the server will accept. This is achieved by using the ACCEPT-CHARSET in the INPUT areas.

The key to localising forms and submission pages is in correctly adapting the form handlers to return appropriate content after the “submit” button has been pressed. It is impossible to give a general rule for localising form handlers, since they can be coded in any language from Perl to Java or shell scripts. Therefore their **localisation** has to be done on a case-by-case basis.

3.6.7. Hot keys

Hot keys are those keys used to activate a certain command by pressing it in combination with the Ctrl key. (19) During **localisation** these hot keys may no longer exist or may need to be changed to some key from the translated text. These changes must be consistent throughout the site and must also comply with the standards established by the OS and hardware of the target language. For example, in English the hot keys to open a file is Ctrl+O (**Open**), whereas in Spanish it is Ctrl+A (**Abrir**). The “delete” button is **borrar** in a Spanish keyboard and so on.

Whenever possible it is advisable not to use accented characters as hot keys since not all keyboards support those characters.

For Japanese, Korean and Chinese, the English hot keys are retained in parentheses after the translation. (17)

3.7. Multimedia

This will involve the hiring of voice over actors and recording facilities for the audio clips that may be inserted. Also the images in a video clip will need to be cross-culturally checked. Special media packages will need to be used to integrate the new audio files.

3.8. Search Engines

Once the site is ready it is important to publicise it through the main search engines in the target country. In order to achieve this, information about the site will have to be submitted to the appropriate search engines in all the languages targeted. The keywords that identify the document will need to be localised as part of the normal site.

An issue to keep in mind when preparing the Metatag for a document is not to use accented letters that have been coded with a reference since the search engine may not recognise the character represented. In general, it is not advisable to use accented characters. A search engine such as Yahoo does not recognise the accented characters in Spanish, for example.

3.9. Identifying international users

If the site is going to have users from different countries and aims to provide personalised service to individuals, it will be necessary to employ server-side scripting techniques to achieve it. ASP, JSP and XSP can provide this functionality. (20)

3.10. Recommendations

- Use UCSs or UTFs to prevent **Character encoding** problems. For systems that only offer 8-bit **strings** but are multi-byte enabled, **UTF-8** is the best choice. For systems with no storage restrictions, **UTF-32** is preferable. If the programming environment is not a problem, **UTF-16** is a good compromise between performance and storage. **Unicode** is not advisable when using an SQL server since there are currently problems in supporting it.

- When working in the Windows environment, use Win32 API NLS functions to handle **locale** sensitive data. One of the main **globalisation** errors is to use hard-coded Natural Language Support (NLS) data. This could be avoided by identifying NLS data in the design phase to make the pages locally independent. Win32 API NLS functions should be used with all programming languages that can access these functions (Visual C++, Visual Basic, and Perl, for example).

- When working with **double-byte** characters all functions used must be examined to see if they can handle them.
• Separate the UI elements and code. The best system consists of creating a resource DLL per target language with a common functionality. It allows user interface switching and gives complete control over the languages installed. It is also easy to update with new languages and there is no need to worry about user, system and thread locales. The only disadvantage is that all language satellite DLLs have to be maintained synchronised. (18)

• Do not use concatenated strings nor build sentences at run time.

• Programmers should always comment their code to help translators since they do not usually have programming skills. Ideally the text strings will be isolated in different files and then included in the pages as SSIs.

• Code must be flexible enough to handle different locales. Programmers should never make assumptions about how a language interprets information.

4. Cultural and Linguistic Issues
4.1. Translation
It is vital to have native speakers localising the site. They need to be linguists as well as having a deep knowledge of the target culture. If the site has highly specialised content, it is also important to get native specialists in that field as the terminology must be adequate for that particular area.

There are two types of content:

Static content: This content does not change regularly and is usually kept in the web site pages as HTML, ASP, etc. The localisation will be done with an HTML editor or translation tool.

Dynamic content: The best way to store this type of text is in a database to make the maintenance easier. The ideal process of localisation will be to identify updated content and to automatically route this through a pre-defined workflow (21). This is the approach adopted at Microsoft (see section 6.2).

Most languages are longer than English, usually expanding by 30%. This will have a huge repercussion on the design of the pages and involve laborious work to find different alternatives. A solution would be “truncation”, using shortened versions of words. For example, a button with the word “NEXT” will be “SIGUIENTE” in Spanish, which could be truncated as “SIG.” or an arrow (➔) could be used instead.

Short cut keys also pose a problem. Commands are different in different languages and it is important to be aware of those nuances. For example, (ctrl-b) for bold corresponds in German to (Strg-f)- Steuerungstaste-fett. Consistency within the site and with existing localised OS and software must be strictly observed.

4.2. Cultural aspects
If the site is going to be successful in the target market it has to be accepted by the users. Sometimes it is not enough to just use the target language; the content has to be appealing and “feel” local.

It is important to work closely with native speakers who know the culture well in order to foresee which items may be seen as offensive or rude. For example, in Muslim countries it is an offence to show a picture of a woman showing her arms or legs and in Greece a map showing Macedonia as an independent state may cause problems, etc.

Negative reactions may also come from the use of colour or icons. The metaphor of a person running to symbolise the running of a program does not work in all languages. An open palm to signal “Stop” may be very confusing or even offensive in certain cultures. Colours have even stronger cultural implications. The colour black, which is widely used in Western sites for its neutral, elegant and sober properties symbolises death in China. White is the mourning colour in China, whereas in the Western world it signifies purity. The examples are countless.
A final consideration is the differences in bandwidth between countries. Some countries have very low bandwidth, which seriously affects downloading times. Apart from trying to keep graphic elements to a minimum and making sure the file sizes are reduced, another solution is the one adopted by Cisco Systems (20). They let users choose servers from the nearest region to reduce downloading times. However, this approach is only feasible for big companies with international presence. It is also worth remembering that some countries have to pay for the time that they are connected to the Internet. The rates are sometimes quite high, which will affect the time they may spend online.

4.3. Legal requirements
Legal requirements in the target country have to be met. Privacy and copyright laws; domain names, for example, to obtain the .es domain in Spain you need to have an office registered in that country; Terms and Conditions; there can be special requirements, for example, in Spain and France there is a regulation that all advertising material or software products must be translated. It is not sufficient to market only an English version.

In Asia, central governments play a crucial role in controlling the economic development of the countries and they actively intervene in all areas, including technology and Internet. A thorough research and professional advice is crucial to avoid legal troubles in the target country.

4.4. Updates and live events
A further issue concerns the currency of the site. Updating is essential in order to keep users coming to the site. The International Dateline of the country where the site is being produced and that of the target market can affect how current the users feel the site is. For example, an update uploaded at 12:00 in England will appear as uploaded the previous day in Australia since they are 9 hours ahead. When some live event is going to be online, for example, a chat with someone important or a sports event, it is advisable to offer the time in different time zones, e.g., the match will start at 17:00 London time, 12:00 New York time, 02:00 Sydney time.

4.5. Recommendations
• avoid slang, colloquialisms, play on words or humour. All these are very difficult to get right in another language.
• group information that can not be localised, such as prices and local promotions. In this way it will be easier to locate it and eliminate it from the localised version.
• avoid bitmaps and icons that may be ethnocentric or offensive.
• avoid maps with controversial national boundaries (22)
• avoid graphics with multiple meanings, for example, using a table as an icon for a table of contents.
• avoid religious symbols such as stars, crosses, half moons, etc.
• do not use hand gestures or body parts.
• when the local time is essential, offer the times in some of the main cities in the different time zones.
• avoid using colours to represent meanings, for example, red to stop, green to grant permission, yellow for danger/precaution.
• get professional legal and cultural advice.

5. Design issues
5.1. Graphics and images
Every graphic on the site needs to be examined both in cultural and technical terms. The images must have the adequate meaning in the target culture, for example, in some Asian sites the icon representing Home is a pair of shoes, instead of a little house.
The size of the images is also important since some countries may have slower connection or older equipment. It is therefore advisable to reduce the image size as much as possible.

But the main problem when dealing with graphics is when these are combined with text. This is increasingly frequent in banners and navigation bars and on many occasions these images are bitmaps with text overlaid. If this is the case it will be necessary to manipulate the image in order to insert the localised text. This is relatively simple when the background is a plain colour, but if
the background contains structure this task could be impossible to achieve. There are several methods that can be used to prevent this situation. Firstly it is possible to use a background image and position the text on top of it using the `<SPAN>` tag and specifying its relative position. Secondly a well-documented, layered PhotoShop (or any other graphics package) source file can be provided. In any case the best approach is to avoid **bitmaps** with text.

The provision of a full spec of the **fonts** and colours used will ensure the localised site has the right appearance, which is essential in keeping a consistent brand image. (8)

Animated graphics will be treated in the same way as static images although the software used will be different. It will be necessary to analyse the different frames to check if there is text to localise and ensure the images are adequate for the target market. A more complex issue are generated graphics with JavaScript based overlays. These graphics use text based **strings** embedded in JavaScript to display over a background image. These need to be localised as part of the JavaScript identification and extraction. If the text is considerably longer it may be necessary to re-size the graphics.

5.2. Nested tables

Nested tables containing images will pose a further problem as text **strings** are usually longer and may stretch the background image. These structures can be very hard to debug. Some possible solutions will be:
- include HTML comments to specify when a new section starts;
- keep the layout as simple as possible. Try to avoid nesting tables on more than two layers;
- use DHTML technology that allows absolute and relative positioning;
- leave enough space for localised text and specify all dimensions to avoid the area to expand. (8)

In general it is better not to use too many nested tables since they complicate the **localisation** process.

5.3. Text length

As mentioned previously, most languages are longer than English, therefore it may be necessary to resize graphics to accommodate text. This may have an impact on the "look and feel" of the page.

5.4. Cascading Style Sheets

Style Sheets can be used to change the visual image of a page. One of the best features of CSS is the facility to change **fonts** in a Web page, by just changing the **font** tag. It is recommended to write a style guide for each market with non Latin alphabets (fonts to use, size, bold, italics, etc.) CSS2 improves the typographic control and allows the dynamic download of fonts (See section 5.6. Fonts).

Sometimes the whole site will have to be re-designed to cope with multi-lingual navigation and adapt the site structure to the new environment. This can be achieved more easily when using CSS as only one file will have to be changed.

Style sheets can also be used to provide text bidirectionality as they control the visual image of the text. The BiDi algorithm works on the inline/block level distinction. If an inline element does not have a **DIR** attribute and is transformed to a block level element by a style sheet, it will inherit the **DIR** from its closest parent block element to define the base direction of the block. This can cause several problems when mixing r-l-l and l-r languages in the same page. It is preferable to use the inline tags to mark bidirectionality.

5.5. Fonts

A common mistake on the Web is to use the `<FONT FACE>` tag to achieve a particular alphabet. Characters and **glyphs** are different concepts and should be kept separately. A **glyph** is a particular image that represent a character or part of a character (7). If a character is coded as a font mapping, the rendering of the **glyphs** depends on the particular code of the font chosen. This causes incompatibility as the page can only be seen if the user has that particular font. Additionally, it causes problems with search engines and the algorithms used to sort lists of
words alphabetically may stop working. A real **Character encoding** such as **Unicode** or **UTF-8** should be used in order to carry the words in plain text, whatever the medium.

Recent advances in font-embedding with version 4.0x browsers allows the dynamical embedding of fonts in documents. For example, for Netscape 4.0 and above, an author can create a font definition file which can be included into a style sheet by using:

```html
<STYLE TYPE="text/css">

<!--@fontdef url (http://www.htmlib.com/fonts/htmlib.pfr);
</STYLE>
```

This will force the browser to download fonts to display downloaded pages.

Font names have to be consistent with the names given in the target country.

The most adequate way of dealing with fonts is by using Cascading Style Sheets (CSS). They allow the fonts to be changed for all pages in one single place and they reduce the number of tags in a document, thus making the localisers job easier.

5.6. Recommendations
- Try to avoid images with overlaid text. If it is not possible to avoid them, offer well documented layered files with details of fonts used, colour specifications, etc.
- Create generic icons and images that do not need to be localised.
- Make sure text can always be wrapped, above all after check boxes and radio buttons.
- Always use CSS to define fonts and format.
- Keep the design simple with plenty of space. Avoid complex nested table layouts and image maps.
- Carefully assess the cultural impact of all graphic images.

6. OTHER ISSUES
6.1 Quality Assurance
Quality is of prime importance in the Web. If the users do not like what is on the screen it would only take them a few seconds to leave the site and never return. Some of the worst problems identified in localised sites are:
- text still in English;
- spelling mistakes;
- grammatical mistakes;
- wrong terminology (above all in highly technical and specialised sites).

It is also important to know your audience and use the right “flavour”. For example, if your main market is Mexico (the Spanish-speaking country with the highest Internet penetration) you need to use the right terms. It is not good enough to have an Iberian Spanish text, you need to go local.

The best solution to keep costs down is to implement a general version, once its functionality is finished the text can be amended to adapt to specific markets.

There are four types of QA:
- Linguistic QA
  It checks the translation accuracy, the consistency of the text and its style. It involves the creation and maintenance of terminology and translation memory databases, the development of linguistic style guides as well as editing and proof reading strategies.
• Technical QA
  It consists of the verification of the readiness of the files for localisation, i.e. the application runs well on a localised version of the OS and the compilation of files, testing of individual files and testing of the built product (verification of code integrity, link integrity, tag correctness, string lengths, consistency of linked text, hot-key conflicts and out-of-order variables).

• Visual QA takes place on the running of the application. Screen layouts and dynamically generated dialogs can be checked at this stage.

• Functional QA verifies the correct operation of the localised site in different networks, with different databases, browsers and with third-party products. (23)

6.2. Updates
The main attraction of the Web is its currency. Yesterday’s news is old news. This has an impact on the maintenance of web sites, it is essential to carry out updates at least once per month to keep engaging the users. This is a serious hurdle to overcome in a multilingual site. Up to now the rule was to get all the updates done in the English site, manually extract the content from the HTML pages, localise it and update the foreign language versions once per week or, more commonly, once per month. Translating a web site is not a single “project” as such, but a continuous flow of changes. It is necessary to develop workflow systems to automate the transfer of files between client, project manager, translator, proof reader and back to the client for final approval.

Different projects are being developed at present to overcome this tedious job and the corresponding delay in updating. One of the most interesting approaches is that of Microsoft (24), who are developing a new series of workflow tools to streamline the process. When an editor in the U.S. makes a change in a database field, it will be marked and notification e-mails will be sent to the localisation offices. In this way the content can be updated proactively and much quicker. The changes in the American site can be reviewed and adapted only if they are deemed relevant. Additionally it will not be necessary to code the pages manually, the database-driven system will enable the automatic building of pages in their own templates.

Another approach could be to create “virtual teams” of translators. When new information is added, the text is transferred to the first translator in the team; if the job is rejected, it will then be sent to a second translator and so on until someone can do it. Once translated, the text is transferred to a proof reader and back to the client for approval. Once approved it can be uploaded in the multilingual database for publication. (25)

6.3. Testing
This is essential to ensure quality. It involves checking that all menus, drop-downs, hotkeys, dialog boxes, links, images, warnings, errors messages, etc. are correctly sized and placed and comply with cultural norms.

It is always a good idea to write test cases focusing on the smallest units of a project. They will help to uncover problems and can be reused in subsequent developments of your site or on other projects. These are some of the approaches adopted in the industry:

6.3.1. Build Verification Test (BVT): (5)
The BVT originates in the software world and is used to check if the build process has been successful. The BVT consists of a subset (normally 40%) of test cases and will focus on functionality rather than language.

6.3.2. Pseudo-localisation:
The process of exercising the site’s user interface, localisability, and site stability before localisation. This is done by quickly editing all of the strings in a project to:
- Include some extended characters.
- Increase the length of the terms and paragraphs.
- Ensure that the design is flexible for all of the terms to be translated.
- Test string truncation.
- Check if all **strings** are accessible to localisers, if all the keyboard shortcuts can be localised, if the characters are displaying correctly in HTML, in all the controls/elements of the web site and in and out of a database. (5)

6.3.3. Pilot Project (5)
This process involves running a **localisation** project for a particular language before starting the same process for other languages. Typically, a pilot project will run a few weeks ahead of the other languages.

As the pilot project is the first time a site is localised by a human being, it will allow the localiser to find some specific issues that may have been overlooked in the **internationalisation** phase. Although a project pilot is a time consuming task, it will greatly reduce the time spent in localising for other languages since most of the main problems will be discovered and sorted out at pilot stage.

The pilot project also allows the **localisation** process to be tested, to ensure all steps make sense and are achievable.

\[
\text{TotalBugs} = \text{CoreBug} \times \text{Languages} \\
\text{TotalBugs} = \text{Engineering Hours} = \text{AddToProjectCost} (5)
\]

From a technical point of view it is advisable to choose a language for the pilot project that will present character coding and character length problems, such as Japanese or Arabic. The more complex the language, the more problems can be sorted at the pilot stage.

From a marketing point of view, it is better to choose a language that has strategic importance. The languages most frequently used for pilots are Japanese and German. Both of these languages use international characters, require more space and command very important markets.

6.4. Recommendations
• A good QA test will ensure the success of your site in your target market.

• Implement a workflow process to handle regular updates.

• Ensure you have an experienced team that includes linguists, regional experts and programmers.

• Write test cases and carry out as many verifications as possible.

7. LOCALISATION PROCESS
The following is a summary of the most common practices currently used by most companies in order to localise their pages. This **localisation** is sometimes carried out in-house, but generally it is done by companies specialising in **localisation** and **globalisation**. Either approach is valid, but it is important to be aware of the higher costs involved in localising sites in-house. When several languages are involved, it will be necessary to hire experts that are native speakers of those languages plus all the technical support. This approach is only valid when the **localisation** is done in one or two languages, the company is a big international corporation or the development of international sites is the core of their business plans. Otherwise, it is recommended to contact a specialist third party company that will have the expertise and experience necessary to carry out the job efficiently. Once the site has been launched, it will be necessary to employ in-house webmasters with some knowledge of the target languages. Updates could be outsourced from translation agencies and uploaded in-house.

7.1. Analysing the target market and the current English site.
Before **localisation** starts it is essential to know the target market. Although this is not, strictly speaking, the job of the localisers, they should be aware of a series of issues such as:

• who the audience is;
• what they expect from the site;
• what does the company expects to achieve;
• who are the competitors in the target market;
• any special legal requirements;
• any special cultural, religious or sociological nuances (sometimes it may be worth to hire and anthropologist if the culture is very different to Western Europe);
• bandwidth requirements and fees for Internet use;
• domain names requirements;
• data protection legislation and copyright issues.

The marketing and legal departments together with the experts from the target country should produce a full specification covering all these areas.

Once this is done, the existing site must be critically assessed. Its technical complexity, download times, content layout, code issues should all be scrutinised. Before starting the localisation process the site should be properly internationalised to make future adaptations to other languages easier.

7.2. Internationalisation

The best time to internationalise a site is at the time of development for a number of reasons:

• errors are found earlier;
• errors will be fixed by the same people that writes the code;
• developers will learn how to internationalise a product and be aware of locale dependent items.

Unfortunately this approach is not always feasible as localisation is usually considered once the English site has been launched and proven to be successful.

When working in a Windows environment, the Win32 NLS APIs may be used as they provide an excellent tool to change the locale information. Some of the main items that have to be considered are:

• Date format (long and short);
• Time format;
• Money and anything related to it: taxes, etc.;
• Number formats;
• Address formats (having a zip field hard-coded);
• Font names, size and decoration.

(see appendix C to see the NLS data fields for Spanish).

Graphics need to be analysed, where possible replacing bitmaps with embedded text for plain graphic images or, alternatively, offering the original layered versions.

All kinds of symbols, colours and images must have their relevance to other countries assessed.

At this point the User Interface files, the content and the code should be separated in different resource files. Currently Server Side Includes are being used to insert content in the layout, this facilitates the task of updating content without disrupting the graphic design of the page. SSIs are also used to automatically insert common items such as banners, navigation bars, headers and footers.

Once the resource files are created it is easy to see the scope of the project, the number of files, the word count (essential to calculate translation costs and timings), etc. It is also possible to analyse the complexities of the code and try to solve problems such as the ones created by the use of concatenated strings, sorting order algorithms, dynamic data, hard coded information, etc. Internationalisation bugs will also be identified and rectified.

The structure of the entire site should be planned before starting the localisation process. One of the main issues here is to decide how to combine the different language versions. The most
common method is to store each language in the same location, as this helps to synchronise updates across all languages at once. Typically this type of site will have a link to each different language on its home page so a single web address can be used globally. A different approach is to store each language site in its own specific country, this slows down the maintenance but speeds up download times locally. A third approach consists of having different sites for different countries. This situation occurs when either the domain names are different when translated, the content is too different culturally or different design versions are used.

The final objective is to set up a global site infrastructure which will be centrally managed by a single technology team with localisable modules to be translated and adapted to any language.

### 7.3. Localisation

Once the site has been internationalised, it is ready to be localised. The first steps will be to send the static text to the translators and localise the textual information. Since translators will not necessarily be familiar with HTML it is advisable to use some kind of filter to lock the tags so only the text to be localised will be highlighted.

The future for the localisation of Type 1 sites is already established, both in HTML and XML: the use of Protected tag editors together with Translation packages will streamline the process. However, Type 2 and 3 sites will need to have new applications developed to facilitate accessibility to coded localisable content.

Not everything will need to be localised, and some pages will require more work than others. All pages should be closely analysed and prioritised according to their language requirements (from simple translation to cultural research and copywriting) and their functional complexity.

The use of CAT tools is indispensable to effectively localise software and web sites. There are three types of CAT tools:

- **Translation Memory**: A database which stores translated sentences.

- **Terminology Tools**: Bilingual glossaries of translated terms or phrases. Most translation memory tools include terminology management applications linked to the translation memory through automatic terminology lookup.

- **Software Localisation Tools**: Used to translate graphical user interfaces of software applications.

A standard tool in the translation market is the TRADOS™ package. There are more tools in the market offering similar features but TRADOS™ is the standard tool in the translation industry and is completely integrated with Microsoft Office applications. TRADOS™ has a feature called TagEditor, which lock the tags so the translators can not delete or alter them. It also builds up a Translation Memory™ that stores and utilises often-repeated phrases. This program assures language consistency throughout the product and helps with updates. The vocabulary kept in the Translation Memory™ is used to “leverage” into the different text versions, which is very useful when updating content.

TRADOS™ process:
The main problem with using TRADOS TM is that it only works with static HTML pages.

Once the translation has been completed, it is necessary to proof read all the pages to ensure the best possible quality. These high levels of quality should be maintained by monitoring the translated site, identifying the sections that will regularly change.

Once the text has been approved four main problem areas appear:

1. support for international characters;
2. maximum length of a localisable resource;
3. resources that should not be localised and break the functionality;
4. resources that should be localised but are not enabled for localisation.

These technical problems arise when the page content is dynamic rather than static. In this case the localisers need to have at least a minimum level of technical knowledge to locate localisable strings.

A possible solution to avoid these problems is to use Localisation Kits. These are used to provide instructions to the localisers, testers and engineers and should be provided before the localisation begins. Localisers need to know what to localise in order to distinguish localisable strings from functional ones. ASP files normally contain large amounts of scripting, with the localisable strings embedded in the code. These kits help the translators but also give a useful overview of the project for programmers, localisers and project managers.

Once the UI, the content and the code have been localised they have to be re-integrated into the site. How this is carried out will depend on how the content and code have been extracted. The most important issue however, is to make sure that the site structure facilitates easy integration of content, and that adequate procedures exist for acceptance testing prior to the site going live. (26)

Once the site has been technically, linguistically and culturally tested and has been launched, it has to be maintained and updated. The ideal solution is to have a webmaster who speaks the target language to carry out the day-to-day running of the site. In order to carry out updates, a system adapted to the size and type of site will have to be implemented. There are no standards
in the market to deal with the localised sites maintenance and updates, and there are thousands of different tools and approaches. Some of these are analysed in the next section.

8. Localisation Tools
In order to automate the localisation process, a number of tools are currently available in the market. This is a brief summary of some of the different localisation packages currently in use or still being developed. This summary is based on the author's personal criteria and is not a comprehensive guide to all of the tools available on the market:

This program is being developed at present and it will allow content creators to store multiple versions of a document, in different languages, behind one URL. The client and server software is being designed to automatically detect which language the user prefers and then select the appropriate document. (27)

8.2. Language Automation INC
This company first translates HTML from English into other languages, ensuring graphics and content are culturally acceptable. They subsequently link and script the pages to facilitate navigation. Finally they install WebPlexer™ software (developed by themselves) to automatically select the correct language on-the-fly, as customers enter the site. (28) It is a combination of integrated software modules and customisable translation services that work together to simplify the development, operation, translation and maintenance of a multilingual web site.

8.3. ITPs Approach
ITP is a major localisation agency belonging to SDL International, a leading globalisation solutions provider.

They first dismantle the UI, its on-line help elements, supporting documentation, and any other components such as splash screens, readme files, etc. into their constituent elements. The translatable text is then translated, and the UI, help files and documentation re-engineered to work correctly in the target language. (29)

They have developed their own software packages that are quite popular in the localisation industry. The S-Tagger™ and ReSET™ (by ITP). ReSET is an ITP-developed application that combines in one package all the engineering tools required for the localisation of resource-based (RC-based) software.

8.4. Visual Source Safe (VSS): This software is used to make files belonging to the base code read-only to help checking a localised file against the original source. This program facilitates the difficult task of localising code, above all, when the localiser is not very technically able.

8.5. SIM-SHIP
Simultaneous Shipping consists of launching all the versions (the original English together with the foreign languages) at the same time. Although this is a difficult task it is not impossible but it will require a more strict approach.

In order to achieve Sim-Ship it is necessary to supply files in their entirety. The source code and support materials must be “frozen”. The product, help and documentation files must be correctly internationalised. Strict, proper project management is also needed and, finally, translation memory tools must be used to streamline the process.

8.6. Localisation during development
This is the approach adopted by Microsoft. The benefits of this method is that it removes errors early in the process, ensures that people familiar with the code fix the problem (instead of having someone else fix the errors) and educate developers so globalisation bugs are not as likely to be introduced later in the project or in future projects. (5)

From a marketing point of view it also ensures that their products are advertised at the same time in more than 25 countries. Their software products are also localised in this way.
On the other hand, its requirements are more strict. It is vital to be fully aware of the stage that development has reached, how much code is “frozen” and that major overhauls will not happen without warning. (26)

9. Conclusions
The US and UK markets are no longer the only ones making use of the Internet. The growth of its usage in other countries has posed new challenges and demanded new solutions to offering content in other languages. The process involves not only translation but cultural adaptation and a series of technical hurdles that have to be overcome. The fact that most English sites were developed without any hint that they would have to be adapted to other languages in the future makes this process even more cumbersome and difficult. A new way of designing and creating sites must be adopted keeping in mind all the international issues that would be affected if the site was to be localised into another language. The localisation process itself is still developing and new tools and approaches appear every day. A parallel could be drawn between the current situation and the state of software development for PCs in the early 1980’s, when there was a proliferation of tools, platforms and technologies. (25) Therefore each localisation project is at present unique. In order to successfully localise a web site in different languages a multi-layered approach must be taken, using different tools and methodologies with strong underlying workflow processes.
A localisation case study

(For confidentiality reasons the name of the company has been changed to timeforparty.co.uk. This is a fictitious name)
1. Introduction
Timeforparty.co.uk is the UK’s leading interactive special occasions web site. The company has approximately 55-60 employees (including freelancers and temporary staff) and has experienced an incredible growth in the past year. It was created in 1998 and, as well as their UK site, they have a French and a German site. They also intend to expand into other European countries in the near future. The last version of the site, v.3, included new channels such as parties and special occasions gift shops which has increased their e-commerce operations.

The aim of the project was to localise the UK v.3 site for the Spanish market. The main idea behind Timeforparty’s European expansion is “keep it local”. It is important that the users feel the site is a real Spanish site and not a translated product, offering relevant and useful content together with a set of planning tools to help in the organisation of special events. Due to a lack of resources the whole localisation process has to be managed and carried out by the author. This site will therefore have to be considered a pilot version. The resource and budget allocation that would have been necessary in a real, large-scale project did not have to be applied in this case. Also, some of the recommendations for localising code such as creating resource files with localisable strings, could not be carried out due to lack of technical support. The Quality Assurance and test cases that would have been implemented in a real situation were also limited to general functional tests in Explorer and Netscape. The final product is a complete Spanish version of the English site with limited functionality. Although its back end functionality did not need any localisation since it does not affect what the users see, it could not be reused. This is due to the fact that the site could not be hosted at the company’s office, which involved a change of environment. To develop full functionality would have meant a whole re-engineering of its code structure, which is out of the scope of this project. However, if the site is eventually launched and hosted in Timeforparty’s live server, the scripts can be re-used. Finally a suggested strategy for a more effective localisation process is also offered.

The project only covered a selection of files and sections from the original English site for two reasons. First of all due to time and resource limitations constraints and, secondly, even if the site was going to be immediately launched, initially it would only contain the main sections and grow gradually as new business relationships were developed and the users became aware of the site.

2. The website
Timeforparty.co.uk is based on a multilayered architecture consisting of a Presentation Layer using Microsoft Internet Information Server (IIS) running on Windows NT 4 and Active Server Pages (ASP) technology; a Business Layer using MTS/Site Server; and a bottom layer where all data is stored in a relational database using SQL Server 7.

The data from the users is collected by SQL and stored on the database. IIS runs the entire operation on the Internet. ASP handles the information coming in from the forms to the SQL Server database, retrieves the data from the SQL Server, and displays the information in an HTML format.

The original site was obviously conceived and designed to be displayed only in English. Most instructions are hard coded, all graphics are bitmaps with overlaid text and it was not possible to get the original layered Photoshop files. The text is encoded using the ISO-8859-1 Latin 1 standard, the locale identifier (LCID) used in Windows for internal identification of regional and local settings was set to 2057, the default for the UK. As the French and German versions were developed independently from the original English site, localisation methods have never been applied before.

This case study will consist of an analysis of the methodology applied to localise Timeforparty.co.uk, paying attention at the issues mentioned in the first section. Some suggestions are made as to how the original English site could be more efficiently internationalised to help future localisations in other languages.
3. Methodology applied
3.1. Assessing the market and analysing the English site

Before launching a site in a foreign country is essential to know the target market and who your competitors are. In traditional commerce this is solved through a competitor and market analysis but in e-commerce, apart from those traditional reports, it is also necessary to have a look at the penetration and use of both computers and the Internet. Different web sites were consulted to assess the impact and use of the Internet in Spain and the level of online sales at present (see appendix C).

The special occasion market in Spain was also analysed using online resources. Due to lack of funding the author could not use more valuable resources offered by market researchers such as Dun&Bradstreet or onesource.com. The sites of all the Spanish competitors were also studied. Using the main search engines all the special occasion sites in Spain were found and inspected. Only 13 sites were found and in only one of them regular updates were taking place. All of them had very few suppliers and no signs of traffic (no community services, very few personal announcements, etc.) were found. This indicated that the online special occasions market is only emerging in Spain. Without a strong competence it looks like a launch in Spain could be a success.

After this market assessment the current English site was carefully inspected, its functionality, navigation and content. Within the time scale it was impossible to localise the whole site so a selection of pages was made. It was also decided that initially the site would not have any e-commerce elements as the main aim is to create an online presence. To implement an online shop would require the application of an e-business package to deal with the transactions and payments. The German and French sites were also examined to discover the strategies employed in them. These foreign versions were developed independently from the UK site and problems were resolved on a case-by-case basis by the developers. Unfortunately the French and German site follow the v.2 design, which is significantly different to v.3, the version adopted for the Spanish site.

The first technical step was to create a staging site within the server initially mirroring the English site. Timeforparty’s server hosts all four sites, English, French, German and Spanish, this does not allow any changes in the locale configuration of the server.

This is the file directory on the staging server:

![Figure 3.1.1. Timeforparty’s Staging Server](image)

These four sites are also different both in content and number of pages, depending on its degree of development and the traditions of each particular country. Therefore this is a peculiar localisation environment. It is not the same site on different languages that can be selected from a common
home page nor identical versions of the same site on different servers. This also has repercussions on updates, as an updated item on the English site does not necessarily mean the localised versions need to be updated as well. This architecture, if controlled from the UK as intended, will demand an in-house team of editors, webmasters and ASP developers in these languages to run the sites. Figures 3.2 and 3.3 show the slight differences between the English and the Spanish sites due to the different stages of development they are in, and the variations in content.

Figure 3.1.2. UK site file structure
One of the first issues to solve was the name of the site. The word Timeforparty translated into Spanish as Tiempodefiesta. As it was too long it was decided to leave it as fiesta. Additionally, to use the Spanish .es domain it is necessary that the company has a registered office in Spain. However, possibly because of this legal requirement, most sites in Spain are .com so finally the domain fiesta.com was chosen. This domain had previously been bought by Timeforparty.co.uk.

### 3.2. Identifying the different elements

All the elements of the site were divided into:

- User Interface
- Code
- Content

#### 3.2.1. User interface

**3.2.1.1. Graphics**

First of all every graphic image used in the site was localised. Since all of them were bitmaps with overlaid text, extensive reworking had to be carried out in Photoshop. Image Ready was used for animated Gifs to extract the text and re-insert it. These files were kept as .PSN graphics with layers for future localisations. Fortunately there were no problems with text expansion. Even when Spanish is longer than English, the words used for the channels were not excessively long, however it is worth mentioning that the text length issue was not taken into consideration when designing the site. This problem was very acute when designing the German site as German has very long words that often had to be truncated.

Photographs also had to be adapted accordingly. Since photographs involve copyright issues, there are not many on the site. The author only used one image from the French site, as it was also adequate for the Spanish market. The other pictures that appear on the site should be sent by the users.

The banners on this site are based on graphics, so once they are ready they need to be assembled. These banners together with the navigation bars and footers are saved as simple tables with no `<html>` or `<body>` tags to be used as SSIs. This system also facilitates the updates, if a new channel is added a single change on the navigation SSI updates all pages on the site.
3.2.1.2 Style Sheets and Tables
The site uses linked CSS that define the fonts, colours, sizes, etc. The layout is achieved through tables. Since some of these tables are nested they caused some problems when creating the templates. In some instances these tables were automatically nested by Dreamweaver (the software package used to code the pages in HTML). This program has a tendency to create arbitrary layout tags in some cases. One of the design rules at Timeforparty.co.uk is to avoid nested tables as much as possible. Most tables were simplified trying to keep them within just two levels. Since the SSIs were created from scratch with the localised graphics, the old pages could not be used as templates and new ones had to be created.

The site uses standard sans-serif fonts, Arial and Helvetica. These fonts are available in most alphabets. If the site was going to be localised in Asian, Indic or bi-directional languages available fonts would need to be researched in order to find the closest match.

3.2.1.3 Navigation
This was kept as similar to the English version as possible since it is not cultural dependant and is very well designed in the original site. The fact that only a reduced version of the original site was being localised reduced the possibility of navigation problems. The only issues examined were the hyperlinks, making sure they were connected to the right page, their names were consistent and were adequate to the reduced content version.

3.2.2 Code
3.2.2.1 Character Encoding
The character encoding of the site was not a problem since Spanish can be covered by the ISO-8859-1 Latin-1. This was the encoding used. Although ideally Unicode or UTF-8 should be used to achieve perfect internationalisation, this will have implications at server end that could demand re-engineering. UTF-8 can also cause problems at the front end as not all browsers are yet able to cope well with this encoding. If non Western European languages were to be considered the issue of character encoding would have to be carefully examined. All pages were correctly labelled on the meta tag with the right encoding.

The special characters of the Spanish language, i.e., á, é, í, ó, ú, ü, `, ¡, ¿, were coded in HTML using ampersand expressions. These were automatically created by Dreamweaver, for example, é was &eacute;. The URL did not pose any special technical problem since none of the special characters were included.

3.2.2.2 HTML
One of the first steps was to change the LCID in the Global.asa file from 2057 (UK) to 3082 (Spanish - modern) to correctly identify locale dependant information such as numbers or dates.

```<SCRIPT LANGUAGE="VBScript" RUNAT="Server">
Sub Session_OnStart
    Session.LCID=3082
End Sub
</SCRIPT>
```

The localisation of the static HTML pages was made using Dreamweaver 3.0. First of all the common parts of the pages such as banners, footers and navigation bars were converted into Server Side Includes and templates were created for each section. Since Dreamweaver displays tags and code strings in different colours it was very clear which parts needed translation. This was carried out directly on the ASP file, in a real life project the tags would be locked and the files sent to a translator.

Operating Systems control some default buttons and alert messages. If an English OS system is used to carry out the localisation, the localisers and testers must be aware of this, otherwise they may consider these buttons and messages in English as mistakes. For example, when the contents of a form are going to be sent by mail, a warning message alerting the user appears, this message will be on the language of the user’s OS. Also the default button “File” in a form will appear as “Browse” in an English OS and as “Buscar” in a Spanish one. Ideally the localisation process should be carried out on a computer with an OS in the target language to check
everything is working fine, and that hot keys and any other function keys or commands are consistent with the OS in the target language.

### 3.2.2.3. Code

In a real project, localisable **strings** would have to be identified and saved in **resource files** that could be reused in future foreign versions. In this case the files were manually examined and localised. In order to extract **strings** and recall them from the original page, significant re-engineering work has to be carried out by a professional programmer, which was impossible in this case.

The code to localise can be divided into three different areas:

- **Javascript**
  Javascript did not present much problem, the site does not use this language very often and most of the time it is for simple environment variables such as dates or validations. The author used resources freely available in the web to get the right scripts and adapt them to her needs. One particular site that was useful was www.htmlgoodis.com, it has a good repertoire of scripts for the Spanish language. Most of the scripts were re-used from the original site adapting them to the new server environment and variables.

- **ASP**
  Active Server Pages is a server-side scripting used to create dynamic, interactive applications that do not consume enormous amounts of processing time and can turn static pages into interactive pages with live content (30). HTML, other scripts, reusable **Active X** server components and both VBScript and JScript can be used with ASP. Since it works at the server end, it rarely affects what the users see, i.e., the items that needs **localisation**. In this case study most of the ASP code could be reused in other language versions. However, since the site has to be hosted in a different environment outside Timeforparty, the pages had to be stripped of any code to avoid generating ASP errors. Apart from the ASP errors, if the code is not removed the pages cannot be viewed in the browser.

  Since all the different sites (French, German, English and Spanish) are kept on the same server it is not possible to use the Windows NLS APIs to change the **locale**. Therefore it is important to specify the right locale identifier (LCID) in the Session object. For example, on the site there is a box showing a survey asking users their opinion about a party topic. The ASP code gets the user’s input and calculates the percentage of yes or no votes. The total figure is represented using the format 000.00%.

  In different languages decimals are represented in different ways, e.g.

  - English  5,000.25
  - Spanish  5 000,25
  - French   5 000,25

  When working in a Windows environment, this can be solved by simply adding the right locale on the global.asa and on the files. E.g.:

  ```
  <% 
  session.LCID=3082
  %> 
  ```

  This LCID code indicates ASP how to render locale information in a correct way.

- **Dynamically generated content**
  Dynamically generated content is database driven which means a connection to a back end database is necessary. Most of the tables used in the English site are used by the French and German sites, the same could apply to the Spanish site. The user name and password tables are the only items that are country-dependant. The party information is entered by the users, presumably in their language. When retrieving the information they will see exactly what they have
previously entered in the languages they used. Unless there are character encoding issues, this area should not be a problem. For this reason, only one table with the user name and password would need to be created on the database, although if the site was going to be properly internationalised, each country would have their own database. This system will also help to manage content and updates in a more efficient way.

In this case study a simple prototype was carried out with one table created in Access to allow users to log in as an example of what should be carried out on a larger scale. On the real site most of the ASP scripts used to connect to the database can be reused since they do not contain any string visible to the user. The code simply opens the connection to the database and extracts the required fields which are then placed on an HTML document. The language used to query the database is standard SQL.

The current sites use Lightweight Directory Access Protocol (LDAP) technology to handle user registrations. LDAP is an Internet standard for directory services that run over TCP-IP. Each combination of user name and password is given a unique identifier allowing access to the personal information previously entered by the user. It is not simply a form or a database, but a specialised server for directories that contain information that is rarely modified, such as user names. (31) LDAP works between the IIS and the underlying database.

3.2.2.4. Database connectivity

The architecture of the original site is very complex and out of the scope of this project. The users register with the site introducing their personal information and the information about their party. Whenever they log in with their user name and password, the site issues a session cookie that maintains their access rights for as long as they are on the site or until they sign out.

For this project a small prototype was developed using Microsoft Access to create a database that will allow the log in of users. In this way the prototype can be compared with the original site, as the procedure for accessing data is almost identical in MS Access and SQL server, except for some database specific behaviour.

In this prototype the users can register with the site and log in to access their personal account and planning tools. All the original ASP code was removed from all the pages to make them accessible although it also eliminated their original functionality.

First of all a simple database was created with just one table, tblUsers:

![Prototype database table](image)

Fig. 3.2.2.4-1 Prototype database table

This table can be used by the users to store personal information about themselves. The Userlogin and Userpassword fields will be used as identification to allow login and user access to this information.

Once the database is ready it has to be connected to the site. Windows 98 provides a simple interface to do this via the Open Database Connectivity (ODBC). ODBC provides an interface that allows applications to access data from different data sources and it is widely used to access relational data. ODBC allows connection to Microsoft Access, Microsoft SQL Server, dBase, Oracle and DB2 amongst others. (30)
Windows Control Panel offers a user-friendly ODBC Data Source Administrator. First of all a new System data source must be added, in this case a Microsoft Access Driver (*.mdb) was added. A Data Source Name has to be given in the System DSN, in this case it was called test, then the database was selected in C://webshare/wwwroot/test.mdb.

Now the ASP has to be written to:

- Get records from the form

  ```
  <% 
  USERNAME=Request.Form("USERNAME") 
  USERLOGIN=Request.Form("USERLOGIN") 
  USERPASSWORD=Request.Form("USERPASSWORD") 
  ROLE=Request.Form("ROLE") 
  ADDRESS1=Request.Form("ADDRESS1") 
  ADDRESS2=Request.Form("ADDRESS2") 
  ADDRESS3=Request.Form("ADDRESS3") 
  COUNTY=Request.Form("COUNTY") 
  POSTCODE=Request.Form("POSTCODE") 
  COUNTRY=Request.Form("COUNTRY") 
  %>
  ```

- Open the connection to the database

  ```
  <% 
  Dim objConn, objRS, strQ, strConn, stroutput 
  Set objConn = Server.CreateObject("ADODB.Connection") 
  strConn = "DSN=test;" 
  objConn.Open strConn 
  %>
  ```

- Store the values

  ```
  <% 
  strQ = "INSERT INTO tblUser (USERNAME, USERLOGIN, USERPASSWORD, ROLE, ADDRESS1, ADDRESS2, ADDRESS3, COUNTY, POSTCODE, COUNTRY) Values " 
  ```
strQ = strQ & "("&USERNAME&", "&USERLOGIN&", "&USERPASSWORD&", "&ROLE&", 
"&ADDRESS1&", "&ADDRESS2&", "&ADDRESS3&", "&COUNTY&", "&POSTCODE&", 
"&COUNTRY&")"
objConn.Execute strQ
%

• Return a dynamically generated thank you page.
  <% 
  Set objRS = objConn.Execute("SELECT * FROM tblUser") 
  %>

  <html>
  <head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
  </head>
  <body bgcolor="#ffffff" link="#330066" vlink="#330066" alink="#330066" text="#330066"
  topmargin="3" leftmargin="3" marginwidth="3" marginheight="3">
  (Rest of the code for the layout code)

  <%=fieldname%> is invoked whenever it is necessary to show the content of a database field.

• Close the connection.
  <% 
  <objRS.close : objConn.close 
  Set objRS = Nothing : Set objConn = Nothing 
  %>

The form address settings had to be localised, the drop down menus had to be translated and rendered in the correct alphabetical order. These menus were taken from other Spanish sites since they are standard lists.

The database should be further developed to allow the creation of party profiles and the use of the planning tools and the message boards. Timeforparty already has the database designed and it will just need to create a further connection to the Spanish site which will basically involve a similar process as the one previously outlined. However, this process will be more complex as LDAP technology and an e-commerce application is involved. Once the connection is ready the users will create their content in their own language.

The registration process can be checked by clicking on "Regístrate" on the home page. Once the user has been registered it is possible to log on to the site.

Once the users have been registered, they can enter their e-mail address on the first box and their password on the second. Then the ASP code will check on the database to find any record that matches those two fields.

• First it will open the connection to the database:
  Dim objConn, objRS, strQ, strConn, strUserLogin, strUserPassword, strUserName
  Set objConn = Server.CreateObject("ADODB.Connection")
  strConn = "DSN=test;"
  objConn.Open strConn

• Then it will create a Record Set to go through the records to try to find a match:
  Set objRS = Server.CreateObject("ADODB.Recordset")
  Set objRS.ActiveConnection = objConn
-- default SQL
strQ = "SELECT tblUser.* FROM tblUser WHERE tblUser.userlogin='" & strUserLogin & "' AND 
tbluser.userpassword = '" & strUserPassword & "'

objRS.Open strQ

• If there are no matches the users will be sent to an error page, otherwise they will get access
to the planning tools:
  If objRS.eof then
      Response.Redirect "my_party /sign_up/error.asp"
  else
      Response.Redirect "my_party/default.asp"
  End If
%

The planning tools are based on this process. The users enter all the different information and then
can retrieve it when they want. On this prototype the pages are stripped of any functionality and
only the layouts can be seen. For a fully functioning site, an entire database system will have to
be designed and developed in the new hosting environment.

3.2.2.5. Other technical issues
• Alphabetical sorting
  The sorting of lists, for example the country lists in the forms, was carried out manually while
  localising. Some of the lists were taken from Spanish sites so no further work was required. It is
  worth noting that this aspect is very often overlooked in localised sites, as the author has observed
  several times in different web sites.

• Composite strings
  Although the site was not internationalised, no cases of composite strings or embedded error
  messages were found.

• Active X
  The site contains Active X elements but none of them contained any embedded text.

• Forms
  Since the character encoding was not a problem, forms did not present many difficulties apart
  from translating the error and thank you messages and adapting them to the new server
  environment.

• Search engines
  In order to submit the site to the search engines, key words should be included in the metatags.
  This has not been done in this case study as the pages will not be submitted due to confidentiality
  reasons. Words with accented characters or the letter “ñ” should be avoided as some search
  engines do not recognise those glyphs.

Example of Meta Tag in the German site for Timeforparty.co.uk:

<META name="DESCRIPTION" content=" Hochzeitsseite, Hochzeitsplanung, Gast, Brautkleider,
  Brautmode, Brautstrauß, Hochzeitskleid, Heiratsantrag, Heiratsanzeige, Hochzeitsfeier,
  Hochzeitsgeschenke, Geschenke für besondere Anlässe">

<META name="KEYWORDS" content="Hochzeit, hochzeit, Brautkleid, brautkleid, Brautkleider,
  brautkleider, brautmode, brautmoden, Brautstrauß, brautstrauß, Blumen, Brautpaar, Brautvater,
  Brautschuhe, brautschuhe, Hochzeitskleid, Flitterwochen, flitterwochen, Feiern, feier,
  Heiratsanzeige, heiratsanzeige, Hochzeitstorte, hochzeitstorte, braut, Braut, Verlobung, verlobung,
  verlobt, verloben, Hochzeitstisch, Blumen, Trauung, Hochzeitsfeier, Rede, Ehe, Heirat,
  Hochzeitstorte, Hochzeitfoto, Standesamt, Trauzeuge, planen, Einladung, Hochzeitsseiten,
  Diamanten, Bräutigam, Kalender, Budget, Rat, Hochzeiten, Bücher, Tipps, Hochzeitskarten
  hochzeitskarten, traumhochzeit, traumhochzeit, Hochzeitsreden, hochzeitsreden, Heiraten,
  heiraten, Heirat, heirat, Hochzeitsplanung, Hochzeitsplaner, Budget, Gast, Checkliste, Klatsch und

N. Corte Fernández, Electronic Publishing 1999/2000 40
It is very important to add as many keywords and descriptions as possible as this will increase the amount of traffic reaching the site. All areas covered by the site should be covered, it is also advisable to add lower and upper case versions in case the search engine is case sensitive.

### 3.2.2.6 Hosting

The site could not be hosted at Timeforparty.co.uk, so the services of a web hosting company, www.hostek.com, were used. They offer space on NT Servers with ASP and database support. The files were sent by FTP and ODBC connection could be managed through a special Control Panel set up on their web site. In order to access the database, its new location has to be indicated and its directory has to be given a new set of permissions. Once the site is ready the Domain Name Registrar switch has to be initiated. This simply consists of giving the Name Server information to the register company, in this case, www.register.com.

### 3.2.3. Content

One of the biggest difficulties with timeforparty.co.uk is the nature of its content which is highly dependent on the culture of the target language. Traditions greatly vary from country to country and one of the requirements of this site is to keep a local “look and feel”. The content requires extensive cultural research and in many instances a simple translation is not possible at all. This involves a large amount of copy writing, i.e., creating content from scratch.

This fact also affects the file structure as new files covering new topics relevant to Spain were added while others had to be deleted. Obviously hyperlinks needed to be checked to ensure they were pointing to the right pages. Due to time constraints only two channels of the English site were localised: special occasions and parties. Flowcharts were used to map the contents of the site and its links, these charts had to be updated several times as the site developed.

As previously indicated, the translation was carried out directly in Dreamweaver 3.0. HTML templates were created with the SSIs and the tags were locked to facilitate translation. In a real life situation the files created would be sent for translation. If the translator used TRADOS or any other translation memory tool, they would use the facilities they provide for locking tags and identifying localisable strings in code. Code localisation should ideally be carried out in co-operation with a localisation engineer.

In this case the author did all the translations directly on Dreamweaver, checking links and localising any code with embedded text at the same time. The main issue is to be careful not to delete any tag or accidentally alter any piece of code.

### 4. Integrating all components

In order to put all the elements together, templates were created in HTML. The common components of each channel were included as SSI, most of these elements were navigation bars and banners. Once the templates were created the static text and images were dropped in them. Front-end scripts such as Javascript were also localised at the same time.

Once the static elements were correctly localised the site functionality was tested to discover which elements are visible at the user end and therefore need localisation. At this point the site still had not been connected to the database, therefore some of the functionality was tested on the English site. The ASP code was examined to find embedded strings and these were localised in the file.

### 5. Testing

Testing was carried out using Netscape Navigator 4.7 and Explorer 5. This testing was carried out to validate the functionality and navigation of the site. There was no time to carry out the...
necessary linguistic and cultural tests. It is worth noting that Netscape is more strict on the
tagging of the document and less tolerable of nested tables, some pages that were fine in Explorer
had to be modified to be seen with Netscape.

6. SUGGESTIONS
In order to efficiently and quickly localise the site in different languages and to facilitate updates,
the current site needs to be fully internationalised. A core set of relevant channels need as
compulsory for all countries to keep consistency amongst the different sites. Areas such as
traditional occasions and celebrations and legal requirements should be kept separately as they
are strongly influenced by the culture of each country, and therefore it will be impossible to
standardise them for all target countries.

6.1 Layout
• Graphics should not have any text overlaid. If this is not possible, the original layered version
should be stored in a central repository for localising the text layer. Resource graphic files will
be created, for example:

![Fig. 6.1. Suggested Graphic Storage](image)

All graphics and tables should have plenty of free space to accommodate longer languages. Table
nesting should also be kept as a minimum, ideally no more than two levels of depth avoiding
nesting within table cells. Netscape Navigator 4 contains a bug that prevents nested tables from
displaying properly, although Netscape 3 or 5 handle nested tables well. Therefore it is
recommended to be strict on this aspect and avoid nesting tables on cells.

• The site’s style sheet should list all fonts used making sure these fonts exist in the target
language.
• Layout elements have to be completely separated from the functionality of the site since the
aesthetics of a site should be customised individually for each country as appropriate.

6.2. Character encoding and scripts
• The character encoding should be changed to Unicode or UTF-8. Since the end-user
applications are not quite ready yet for UTF-8 it would be preferably to use ISO-8859-1 Latin
when publishing in Western European languages and UTF-8 internally to facilitate further
 expansion into other languages. The HTTP protocol should always include the charset label to
indicate what encoding is being used.
• Forms should be correctly labelled for the server and the browser to identify the encoding.
When the form is used for requesting geographical information, as in the party profile and User
information in Timeforparty, all locale dependant information such as addresses, post codes,
lists of regions and countries should be saved as SSI. For example, in the case of Spain, the
default country should be Spain, the administrative areas should be divided by provinces, and
the post code should consist of 5 figures, etc. In the case of Timeforparty the information is

N. Corte Fernández, Electronic Publishing 1999/2000 42
entered by the user in their own language and stored in a database. When the user retrieves that information they get exactly what they entered. Therefore, apart from the server correctly interpreting the character encoding, there is not much localisation work at the back end.

- Although the database could be shared by the different languages providing they all use the same character encoding, a better solution would be to keep the different languages in different databases to facilitate updates and administration.
- The best solution for dealing with the code would be to identify and extract all the text strings in the source code, translate them into the target language and place them into resource files. In this way there would be one core binary and multiple international DLLs for each country. The translated software would then be tested and the screen display edited.
- Target language specific information must not be hard coded, should be grouped in resource files to facilitate localisation. In the same way country specific information that appears in the HTML static content, such as prices, taxes or promotions, should be grouped either at the beginning or the end of the document to avoid mistakes and facilitate amendments. Alternatively, this information could all be grouped in an individual page linked to the main page. The main page would then display a list of links to the different languages either by textual links or by using icons such as flags.
- The information in the meta tags should be as extensive as possible to allow access by the main search engines.
- Drop down menus and all kind of listings should be correctly sorted in the target language alphabet. This should be done by the localiser while localising the code. Standard lists such as countries and provinces could be done beforehand and kept in resource files.

6.3. Content

- Timeforparty’s content is very culture dependent and the tone is quite informal and friendly. In order to achieve the same results in the target language it would be better to copy write instead of merely translating the information.
- If external translators are to be employed the project manager should have an exact word count, as translation prices are based on the number of words. A decision should be made about how much content needs to be translated and how much needs to be copy written. Finally, proof reading and a check of the layout should be done once the page has been assembled.

6.4. Updates and maintenance

- A workflow system should be envisaged to administrate the updates and day-to-day running of the site. A configuration management system should be used to keep track of each of the language versions of the source code.
- It is advisable to devise localisation kits when different engineers, translators, proof readers, etc. are going to be involved in the same project. The localisation kit should clearly specify all source files built in proper directory trees, explanations of current developments, indication of content, volume and modifications expected over the time, a list of tools used to develop the original site, etc. The more detailed the document is, the less likely it is for errors to appear.

7. RECOMMENDED SOFTWARE

In order to automate and streamline the process it would be advisable to employ some specialised software to ease and speed up the procedures. The following packages have been chosen by the author after researching several applications available in the market.

7.1. TRADOS

Trados is the perfect solution to manage the translation workflow and leverage content, assuring consistency and team work. Trados offers a translation memory that stores every translation previously made checking new text against the segments already translated. The translator will therefore only need to go over the document editing any mistake and giving the text the right style.

Trados is ideal for the translation of static HTML since it allows the locking of tags and translation directly on the HTML document, avoiding the possibility of any accidental alteration of the code. Its TagEditor also has a special Internet Explorer preview mode. Once the document has been translated it can also verify that all the tagging is correct. The user interface can also be displayed in several languages to help translators.
It also offers a terminology database, Multiterm, that automatically searches for translated terms
during translation. The Translation Memory and Multiterm allow concurrent use by different
translators, so a team of translators can be employed to work on the site without compromising
style and consistency. Each user can be allowed different access level for security reasons and to
avoid accidental deletions or unauthorised access.

Trados also supports SGML and XML, Unicode and, therefore, all languages available in Windows
95, 98, 2000 and NT, including Central Europe, Cyrillic, and bidirectional languages such as
Arabic and Hebrew with all their fonts.

(Information obtained in Trados web site at www.trados.com) (32)

7.2. WebPlexer
The WebPlexer system is a combination of integrated software modules and customisable
translation services that work together to simplify the development, operation and maintenance of
multilingual web sites. The system has been developed by Language Automation Plc.

It builds a multilingual site automatically without necessarily replicating the original site, which
perfectly fits with the international situation of Timeforparty. WebPlexer runs on the server side of a
web connection.

WebPlexer is cross-platform (both Unix and NT) and works with all popular Web servers (including
Microsoft's IIS/ISAPI Web server), and therefore could be easily applied in Timeforparty's technical
environment. Since WebPlexer does not use ISAPI, NSAPI or CGI interfaces, it will not conflict
with any existing applications or modules. In fact, WebPlexer is designed as a downstream
application, i.e. an application which runs outside of the Web server environment, and is invoked
after the operation of serving a document has been completed. It will therefore work seamlessly
with any back end applications, databases, application servers, CGI processes, etc.

WebPlexer offers different modules:

- WebPlexer's Visitor module.
  This automatically finds out the language and country of a Web site visitor and indicates to the
  Web server which localised version to send to the user's browser. It supports content in all
  languages and encodings, and selects automatically the font to use in the browser. It also
  provides an enhanced Web server log with a detailed breakdown of country and language for each
  visitor of the site.

- WebPlexer's Developer module
  This allows multiple languages to be served from a single site. It builds a master site that is
  language independent. The language content is placed in a WebPlexer database where it can be
  easily managed and maintained, making the addition of languages and pages very easy and cost-
  effective. Developer supports both static and dynamic web sites and can be used with existing
  backend databases.

- WebPlexer's Workflow Manager
  This allows an ease and cost-effective maintenance. It identifies language content that is out of
date and facilitates translation and update through a user-definable workflow process.

Workflow Manager offers a Management Console that detects when a document in the master
language has been updated and notifies the manager which corresponding documents in the
other languages require translation. The documents are then send through a workflow and are
dynamically routed to individual translators and proof readers. It features a version control system
that allows leveraging of previous translations.
8. Testing and quality assurance

Finally, testing and quality assurance routines should also be implemented. The first test will be on the original site to check it has been properly internationalised, all the functionality works and any bugs that may appear are eliminated. This first test is crucial due to the fact that if there are any problems with the original site, they will multiply in the different language versions.

The quality assurance tests should look at the linguistic, technical, visual and functional aspects of the site. It is advisable to have an in-country review once the localised site is ready to make sure it does not conflict with the country’s OS, that everything works and for a final cross cultural check.

In this particular case study it would be advisable to carry out a localisation pilot project to find out the most likely major problems and based on that build a series of test cases to apply in subsequent localisations. This Spanish prototype has already found several issues which could form the basis of a preliminary test case.

Managers should be designated to approve texts, agree on updates and decide if these updates are relevant for all the sites or for only some of them. Since some changes and updates may imply technical issues, a system of financial approvals should also exist to allow the allocation of budget and resources.

9. Conclusions

The first step before creating more language versions would be to internationalise the English site. At present, problems with foreign versions are carried out on a case-by-case basis and there is no updating or version control. Although the task of internationalisation is time consuming and costly, its benefits would be noticeable in the long term: ease of use; quicker production of localised versions; developer’s time savings; efficient use of resources; effective version and updating management; and less technical problems during localisation. In order to automate the process different software packages can be used. Trados and WebPlexer offer a good combination to optimise the localisation process. Finally a thorough program of tests and quality assurance methods should be implemented to make sure the users are 100% satisfied with the site and accept it as if it had been created in their own language from scratch.
References:

7. DAVIS, M. “Forms of Unicode”.
8. EBBEN S. and MARSHALL G. “Designing a Globalized and Localizable Web Site”.
18. “Writing multilingual user interface applications”.
20. RADOSEVICH, L. “Going global overnight”.
24. VIESSE, A. “Localization: Done right, it’s part of the spec, not an afterthought”. http://www.microsoft.com/backstage/column_3.htm (accessed on 20/6/00)


32. www.trados.com

33. www.lai.com
Bibliography:

Printed sources

Internet sources
- http://www.webmonkey.com
- http://www.multilingualwebmaster.com
- http://www.fxtrans.com
- http://www.kudos.com
- http://www.aimc.es
- http://www.babel.alis.com
- http://www.unicode.org
- http://oss.software.ibm.com
- http://www.multilingual.com
- http://www.localization-institute.org

Magazines
Multilingual Computing & Technology
ITI Bulletin

Newsgroups and newsletters
www3.w3.org/international
Microsoft Developer and This Week
Transfree
APPENDIX A

Named entities for ISO 8859-1 characters

The named character entities in this section produce characters whose numeric equivalents should already be supported by conforming HTML 2.0 user agents. Thus, the named character entity &divide; is a more convenient form than &247; for obtaining the division sign (÷).

Character 65533 (FFFD hexadecimal) is the last valid character in UCS-2. 65534 (FFFE hexadecimal) is unassigned and reserved as the byte-swapped version of ZERO WIDTH NON-BREAKING SPACE for byte-order detection purposes. 65535 (FFFF hexadecimal) is unassigned.

The list of characters

<!-- Portions © International Organization for Standardization 1986
Permission to copy in any form is granted for use with conforming SGML systems and applications as defined in ISO 8879, provided this notice is included in all copies.
-->
<!-- Character entity set. Typical invocation:
<!ENTITY % HTMLlat1 PUBLIC "-//W3C//ENTITIES Full Latin 1//EN//HTML">
%HTMLlat1;
-->
<!ENTITY nbsp  CDATA "&#160;" -- no-break space -->
<!ENTITY iexcl CDATA "&#161;" -- inverted exclamation mark -->
<!ENTITY cent  CDATA "&#162;" -- cent sign -->
<!ENTITY pound CDATA "&#163;" -- pound sterling sign -->
<!ENTITY curren CDATA "&#164;" -- general currency sign -->
<!ENTITY yen    CDATA "&#165;" -- yen sign -->
<!ENTITY brvbar CDATA "&#166;" -- broken (vertical) bar -->
<!ENTITY sect   CDATA "&#167;" -- section sign -->
<!ENTITY uuml CDATA "&#168;" -- umlaut (dieresis) -->
<!ENTITY copy CDATA "&#169;" -- copyright sign -->
<!ENTITY ordf CDATA "&#170;" -- ordinal indicator, feminine -->
<!ENTITY laquo CDATA "&#171;" -- angle quotation mark, left -->
<!ENTITY not    CDATA "&#172;" -- not sign -->
<!ENTITY shy    CDATA "&#173;" -- soft hyphen -->
<!ENTITY reg    CDATA "&#174;" -- registered sign -->
<!ENTITY macr CDATA "&#175;" -- macron -->
<!ENTITY deg CDATA "&#176;" -- degree sign -->
<!ENTITY plusmn CDATA "&#177;" -- plus-or-minus sign -->
<!ENTITY sup2 CDATA "&#178;" -- superscript two -->
<!ENTITY sup3 CDATA "&#179;" -- superscript three -->
<!ENTITY acute CDATA "&#180;" -- acute accent -->
<!ENTITY micro CDATA "&#181;" -- micro sign -->
<!ENTITY para CDATA "&#182;" -- pilcrow (paragraph sign) -->
<!ENTITY middot CDATA "&#183;" -- middle dot -->
<!ENTITY cedil CDATA "&#184;" -- cedilla -->
<!ENTITY sup1 CDATA "&#185;" -- superscript one -->
<!ENTITY ordm CDATA "&#186;" -- ordinal indicator, masculine -->
<!ENTITY raquo CDATA "&#187;" -- angle quotation mark, right -->
<!ENTITY frac14 CDATA "&#188;" -- fraction one-quarter -->
<!ENTITY frac12 CDATA "&#189;" -- fraction one-half -->
<!ENTITY frac34 CDATA "&#190;" -- fraction three-quarters -->
<!ENTITY iquest CDATA "&#191;" -- inverted question mark -->
<!ENTITY Agrave CDATA "&#192;" -- capital A, grave accent -->
<!ENTITY Aacute CDATA "&#193;" -- capital A, acute accent -->

N. Corte Fernández, Electronic Publishing 1999/2000 49
Named entities for symbols, mathematical symbols, and Greek letters

The named character entities in this section produce characters that may be represented by glyphs in the widely available Adobe Symbol font, including Greek characters, various bracketing symbols, and a selection of mathematical operators such as gradient, product, and summation symbols.

When to use Greek entities. This entity set contains all the letters used in modern Greek. However, it does not include Greek punctuation, precomposed accented characters nor the non-spacing accents (tonos, dialytika) required to compose them. There are no archaic letters, Coptic-unique letters, or precomposed letters for Polytic Greek. The entities defined here are not intended for the representation of modern Greek text and would not be an efficient representation; rather, they are intended for occasional Greek letters used in technical and mathematical works.

The list of characters

<!ENTITY thorn CDATA "&#254;" -- small thorn, Icelandic -->
<!ENTITY yuml CDATA "&#255;" -- small y, dieresis or umlaut mark -->

Named entities for symbols, mathematical symbols, and Greek letters

The named character entities in this section produce characters that may be represented by glyphs in the widely available Adobe Symbol font, including Greek characters, various bracketing symbols, and a selection of mathematical operators such as gradient, product, and summation symbols.

When to use Greek entities. This entity set contains all the letters used in modern Greek. However, it does not include Greek punctuation, precomposed accented characters nor the non-spacing accents (tonos, dialytika) required to compose them. There are no archaic letters, Coptic-unique letters, or precomposed letters for Polytic Greek. The entities defined here are not intended for the representation of modern Greek text and would not be an efficient representation; rather, they are intended for occasional Greek letters used in technical and mathematical works.

The list of characters

<!ENTITY % HTMLsymbol PUBLIC "-//W3C//ENTITIES Symbolic//EN//HTML" %HTMLsymbol; -->

<!-- Portions © International Organization for Standardization 1986: Permission to copy in any form is granted for use with conforming SGML systems and applications as defined in ISO 8879, provided this notice is included in all copies. -->

<!-- Relevant ISO entity set is given unless names are newly introduced. New names (i.e., not in ISO 8879 list) do not clash with any existing ISO 8879 entity names. ISO 10646 character numbers are given for each character, in hex. CDATA values are decimal conversions of the ISO 10646 values and refer to the document character set. Names are Unicode 2.0 names. -->

<!-- Latin Extended-B -->
<!ENTITY fnof CDATA "&#402;" -- latin small f with hook, =function, =florin, u+0192 ISOtech -->

<!-- Greek -->
<!ENTITY Alpha CDATA "&#913;" -- greek capital letter alpha, u+0391 -->
<!ENTITY Beta CDATA "&#914;" -- greek capital letter beta, u+0392 -->
<!ENTITY Gamma CDATA "&#915;" -- greek capital letter gamma, u+0393 ISOgrk3 -->
<!ENTITY Delta CDATA "&#916;" -- greek capital letter delta, u+0394 ISOgrk3 -->
<!ENTITY Epsilon CDATA "&#917;" -- greek capital letter epsilon, u+0395 -->
<!ENTITY Zeta CDATA "&#918;" -- greek capital letter zeta, u+0396 -->
<!ENTITY Eta CDATA "&#919;" -- greek capital letter eta, u+0397 -->
<!ENTITY Theta CDATA "&#920;" -- greek capital letter theta, u+0398 ISOgrk3 -->
<!ENTITY Iota CDATA "&#921;" -- greek capital letter iota, u+0399 -->
<!ENTITY Kappa CDATA "&#922;" -- greek capital letter kappa, u+039A -->
<!ENTITY Lambda CDATA "&#923;" -- greek capital letter lambda, u+039B ISOgrk3 -->
<!ENTITY Mu CDATA "&#924;" -- greek capital letter mu, u+039C -->
<!ENTITY Nu CDATA "&#925;" -- greek capital letter nu, u+039D -->
<!ENTITY Xi CDATA "&#926;" -- greek capital letter xi, u+039E ISOgrk3 -->
<!ENTITY Omicron CDATA "&#927;" -- greek capital letter omicron, u+039F -->

N. Corte Fernández, Electronic Publishing 1999/2000 51
Named entities for markup-significant and internationalization characters

The named character entities in this section are for escaping markup-significant characters (these are the same as those in HTML 2.0 and 3.2), for denoting spaces and dashes. Other characters in this section apply to internationalization issues such as the disambiguation of bidirectional.

Entities have also been added for the remaining characters occurring in CP-1252 which do not occur in the HTML Lat1 or HTML Symbol entity sets. These all occur in the 128 to 159 range within the cp-1252 charset. These entities permit the characters to be denoted in a platform-independent manner.

The list of characters

<!-- Special characters for HTML -->

<!-- Character entity set. Typical invocation: -->
<!ENTITY % HTMLspecial PUBLIC "-//W3C//ENTITIES Special//EN//HTML" %HTMLspecial; -->

<!-- Portions © International Organization for Standardization 1986: Permission to copy in any form is granted for use with conforming SGML systems and applications as defined in ISO 8879, provided this notice is included in all copies. -->

<!-- Relevant ISO entity set is given unless names are newly introduced. New names (i.e., not in ISO 8879 list) do not clash with any existing ISO 8879 entity names. ISO 10646 character numbers are given for each character, in hex. CDATA values are decimal conversions of the ISO 10646 values and refer to the document -->
character set. Names are Unicode 2.0 names.

<!-- C0 Controls and Basic Latin -->
<!ENTITY quot CDATA "&amp;quot; -- quotation mark, =apl quote, u+0022 ISOnum -->
<!ENTITY amp CDATA "&amp;" -- ampersand, u+0026 ISOnum -->
<!ENTITY lt CDATA "&lt;" -- less-than sign, u+003C ISOnum -->
<!ENTITY gt CDATA "&gt;" -- greater-than sign, u+003E ISOnum -->

<!-- Latin Extended-A -->
<!ENTITY OElig CDATA "&amp;#338;" -- latin capital ligature oe, u+0152 ISOlat2 -->
<!ENTITY oelig CDATA "&amp;#339;" -- latin small ligature oe, u+0153 ISOlat2 -->
<!-- ligature is a misnomer, this is a separate character in some languages -->
<!ENTITY Scaron CDATA "&amp;#352;" -- latin capital letter s with caron, u+0160 ISOlat2 -->
<!ENTITY scaron CDATA "&amp;#353;" -- latin small letter s with caron, u+0161 ISOlat2 -->
<!ENTITY Yuml CDATA "&amp;#376;" -- latin capital letter y with diaeresis, u+0178 ISOlat2 -->

<!-- Spacing Modifier Letters -->
<!ENTITY circ CDATA "&amp;#710;" -- modifier letter circumflex accent, u+02C6 ISOpub -->
<!ENTITY tilde CDATA "&amp;#732;" -- small tilde, u+02DC ISOdia -->

<!-- General Punctuation -->
<!ENTITY ensp CDATA "&amp;#8194;" -- en space, u+2002 ISOpub -->
<!ENTITY emsp CDATA "&amp;#8195;" -- em space, u+2003 ISOpub -->
<!ENTITY thinsp CDATA "&amp;#8201;" -- thin space, u+2009 ISOpub -->
<!ENTITY zwnj CDATA "&amp;#8204;" -- zero width non-joiner, u+200C NEW RFC 2070 -->
<!ENTITY zwj CDATA "&amp;#8205;" -- zero width joiner, u+200D NEW RFC 2070 -->
<!ENTITY lrm CDATA "&amp;#8206;" -- left-to-right mark, u+200E NEW RFC 2070 -->
<!ENTITY rlm CDATA "&amp;#8207;" -- right-to-left mark, u+200F NEW RFC 2070 -->
<!ENTITY ndash CDATA "&amp;#8211;" -- en dash, u+2013 ISOpub -->
<!ENTITY mdash CDATA "&amp;#8212;" -- em dash, u+2014 ISOpub -->
<!ENTITY lsquo CDATA "&amp;#8216;" -- left single quotation mark, u+2018 ISOnum -->
<!ENTITY rsquo CDATA "&amp;#8217;" -- right single quotation mark, u+2019 ISOnum -->
<!ENTITY sbquo CDATA "&amp;#8218;" -- single low-9 quotation mark, u+201A NEW -->
<!ENTITY ldquo CDATA "&amp;#8219;" -- right double quotation mark, u+201C ISOnum -->
<!ENTITY rdquo CDATA "&amp;#8220;" -- left double quotation mark, u+201D ISOnum -->
<!ENTITY bquo CDATA "&amp;#8222;" -- double low-9 quotation mark, u+201E NEW -->
<!ENTITY dagger CDATA "&amp;#8224;" -- dagger, u+2020 ISOpub -->
<!ENTITY Daggar CDATA "&amp;#8225;" -- double dagger, u+2021 ISOpub -->
<!ENTITY permil CDATA "&amp;#8240;" -- per mille sign, u+2030 ISOtech -->
<!ENTITY laquo CDATA "&amp;#8249;" -- single left-pointing angle quotation mark, u+2039 ISO proposed -->
<!-- laquo is proposed but not yet ISO standardised -->
<!ENTITY rsquo CDATA "&amp;#8250;" -- single right-pointing angle quotation mark, u+203A ISO proposed -->
<!-- rsquo is proposed but not yet ISO standardised -->
Appendix B
Spanish country and language locale information
Country and language specifications

Country Code: 34
English name of country: Spain
Abbreviated country name (ISO Standard 3166): ES
Native name of country: España
Main Language: Spanish
Native language name: Español
Abbreviated language name: Esp

General Numeric Status:
Positive numeric pattern: 1.000 (no sign string is used)
Negative numeric pattern: -1.000
Thousand separator: period (.)
Decimal separator: comma, ex: 1,23
List separator: ;
Numeric grouping: Sizes for each group of digits to the left of the decimal; sizes are separated by semicolons. If the last value is 0, the preceding value is repeated. To group thousands, specify "3;0".
List of the ten native digits: 0123456789
Position of sign string in positive values: No sign string is used. Ex. ESP 1.234
Position of sign string in negative values: Immediately precedes the monetary symbol, ex. -1.234
Decimal digits: 0,00
String used for positive sign: No positive sign used
String used for negative sign: dash, ex. –1.234
Metric measurement system: yes
Spacing before measurement abbreviation: yes, single space ex: 5 m
Spacing before symbols %, etc: No, ex. 55%

Monetary/Currency Settings:
Currency name: Peseta
International monetary symbol: ESP
Monetary decimal separator: Comma
Currency decimal digits: 0
Monetary thousands separator: period (.)
Currency symbol, after the number: pts.

Date/Calendar Formatting Options
Which week of the year is considered first: Week number 1 contains the first Thursday of January
Which day of the week is considered first: Monday
Short date format: dd/mm/yy
Short date order: DMY
Long date format: dddd de mmmm de(l) yyyy
Long date order: DMY
Medium Date
Short Date
Number of digits for century in short date format: 2
Leading zeros in month fields for long date format: n/a
Month Information:
1. enero (January)
2. febrero (February)
3. marzo (March)
4. abril (April)
5. mayo (May)
6. junio (June)
7. julio (July)
8. agosto (August)
9. septiembre (September)
10. octubre (October)
11. noviembre (November)
12. diciembre (December)
Month names abbreviated:
1. ene
2. feb
3. mar
4. abr
5. may
6. jun
7. jul
8. aug
9. sep
10. oct
11. nov
12. dec
Day Information:
lunes (Monday)
martes (Tuesday)
miércoles (Wednesday)
jueves (Thursday)
viernes (Friday)
sábado (Saturday)
domingo (Sunday)
Day names abbreviated:
lun
mar
mié
jue
vie
sáb
dom
Time Setting:
Time format: hh.mm.ss, ex. 09.59
24-hour format: Yes, ex. 22:45
<table>
<thead>
<tr>
<th>Time Display</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>am/pm:</td>
<td>AM/PM</td>
</tr>
<tr>
<td>Long Time</td>
<td>GMT-07:00 H:mm:ss 1:47:08 GMT-07:00</td>
</tr>
<tr>
<td>Medium Time</td>
<td>H:mm:ss 1:47:08</td>
</tr>
<tr>
<td>Short Time</td>
<td>H:mm 1:47</td>
</tr>
</tbody>
</table>

2001 LRC Best Thesis Award Winner
APPENDIX C

AIMC’s report on the Internet Usage in Spain.

Access and use in the last 30 days.
Last Access
Place of Access
Services Used
Profile by sex, age and social class
Internet usage by provinces
## APPENDIX D

### Language Codes: ISO 639, Microsoft and Macintosh

<table>
<thead>
<tr>
<th>Language</th>
<th>ISO Code</th>
<th>Win Code</th>
<th>Mac Name</th>
<th>Mac Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abkhazian</td>
<td>ab</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Afar</td>
<td>aa</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Afrikaans</td>
<td>af</td>
<td>0x0036</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Albanian</td>
<td>sq</td>
<td>0x001c</td>
<td>langAlbanian</td>
<td>36</td>
</tr>
<tr>
<td>Amharic</td>
<td>am</td>
<td></td>
<td>langAmharic</td>
<td>85</td>
</tr>
<tr>
<td>Arabic</td>
<td>ar</td>
<td>0x0001</td>
<td>langArabic</td>
<td>12</td>
</tr>
<tr>
<td>Armenian</td>
<td>hy</td>
<td></td>
<td>langArmenian</td>
<td>51</td>
</tr>
<tr>
<td>Assamese</td>
<td>as</td>
<td></td>
<td>langAssamese</td>
<td>68</td>
</tr>
<tr>
<td>Aymara</td>
<td>ay</td>
<td></td>
<td>langAymara</td>
<td>134</td>
</tr>
<tr>
<td>Azerbaijani</td>
<td>az</td>
<td></td>
<td>langAzerbaijani(Latin),</td>
<td>49(L), 50(A)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>langAzerbaijanAr(Arabic)</td>
<td></td>
</tr>
<tr>
<td>Bashkir</td>
<td>ba</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basque</td>
<td>eu</td>
<td>0x002d</td>
<td>langBasque</td>
<td>129</td>
</tr>
<tr>
<td>Bengali (Bangla)</td>
<td>bn</td>
<td></td>
<td>langBengali</td>
<td>67</td>
</tr>
<tr>
<td>Bhutani</td>
<td>dz</td>
<td></td>
<td>langDzongkha</td>
<td>137</td>
</tr>
<tr>
<td>Bihari</td>
<td>bh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bislama</td>
<td>bi</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Breton</td>
<td>br</td>
<td></td>
<td>langBreton</td>
<td>142</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>bg</td>
<td>0x0002</td>
<td>langBulgarian</td>
<td>44</td>
</tr>
<tr>
<td>Burmese</td>
<td>my</td>
<td></td>
<td>langBurmese</td>
<td>77</td>
</tr>
<tr>
<td>Byelorussian</td>
<td>be</td>
<td>0x0023</td>
<td>langByelorussian</td>
<td>46</td>
</tr>
<tr>
<td>Cambodian</td>
<td>km</td>
<td></td>
<td>langKhmmer</td>
<td>78</td>
</tr>
<tr>
<td>Catalan</td>
<td>ca</td>
<td>0x0003</td>
<td>langCatalan</td>
<td>130</td>
</tr>
<tr>
<td>Chewa</td>
<td></td>
<td></td>
<td>langChewa</td>
<td>92</td>
</tr>
<tr>
<td>Chinese</td>
<td>zh</td>
<td>0x0004</td>
<td>langTradChinese,</td>
<td>19(T), 33(S)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>langSimpChinese</td>
<td></td>
</tr>
<tr>
<td>Corsican</td>
<td>co</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Croatian</td>
<td>hr</td>
<td>0x001a</td>
<td>langCroatian</td>
<td>18</td>
</tr>
<tr>
<td>Czech</td>
<td>cs</td>
<td>0x0005</td>
<td>langCzech</td>
<td>38</td>
</tr>
<tr>
<td>Danish</td>
<td>da</td>
<td>0x0006</td>
<td>langDanish</td>
<td>7</td>
</tr>
<tr>
<td>Dutch</td>
<td>nl</td>
<td>0x0013</td>
<td>langDutch</td>
<td>4</td>
</tr>
<tr>
<td>English</td>
<td>en</td>
<td>0x0009</td>
<td>langEnglish</td>
<td>0</td>
</tr>
<tr>
<td>Esperanto</td>
<td>eo</td>
<td></td>
<td>langEsperanto</td>
<td>94</td>
</tr>
<tr>
<td>Estonian</td>
<td>et</td>
<td>0x0025</td>
<td>langEstonian</td>
<td>27</td>
</tr>
<tr>
<td>Faeroese</td>
<td>fo</td>
<td>0x0038</td>
<td>langFaeroese</td>
<td>30</td>
</tr>
<tr>
<td>Farsi</td>
<td>fa</td>
<td>0x0029</td>
<td>langFarsi, langPersian</td>
<td>31</td>
</tr>
<tr>
<td>Fijian</td>
<td>fj</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finnish</td>
<td>fi</td>
<td>0x000b</td>
<td>langFinnish</td>
<td>13</td>
</tr>
<tr>
<td>Flemish</td>
<td></td>
<td></td>
<td>langFlemish</td>
<td>34</td>
</tr>
<tr>
<td>French</td>
<td>fr</td>
<td>0x000c</td>
<td>langFrench</td>
<td>1</td>
</tr>
<tr>
<td>Frisian</td>
<td>fy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Galician</td>
<td>gl</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Galla</td>
<td></td>
<td></td>
<td>langGalla</td>
<td>87</td>
</tr>
<tr>
<td>Georgian</td>
<td>ka</td>
<td></td>
<td>langGeorgian</td>
<td>52</td>
</tr>
<tr>
<td>German</td>
<td>de</td>
<td>0x0007</td>
<td>langGerman</td>
<td>2</td>
</tr>
<tr>
<td>Language</td>
<td>Code</td>
<td>Native Name</td>
<td>Native Language</td>
<td>ID</td>
</tr>
<tr>
<td>------------------</td>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
<td>-----</td>
</tr>
<tr>
<td>Greek</td>
<td>el</td>
<td></td>
<td>langGreek</td>
<td>14</td>
</tr>
<tr>
<td>Greenlandic</td>
<td>kl</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Guarani</td>
<td>gn</td>
<td></td>
<td>langGuarani</td>
<td>133</td>
</tr>
<tr>
<td>Gujarati</td>
<td>gu</td>
<td></td>
<td>langGujarati</td>
<td>69</td>
</tr>
<tr>
<td>Hausa</td>
<td>ha</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hebrew</td>
<td>iw, he</td>
<td>0x000d</td>
<td>langHebrew</td>
<td>10</td>
</tr>
<tr>
<td>Hungarian</td>
<td>hu</td>
<td></td>
<td>langHungarian</td>
<td>26</td>
</tr>
<tr>
<td>Icelandic</td>
<td>is</td>
<td></td>
<td>langIcelandic</td>
<td>15</td>
</tr>
<tr>
<td>Indonesian</td>
<td>in, id</td>
<td>0x0021</td>
<td>langIndonesian</td>
<td>81</td>
</tr>
<tr>
<td>Interlingua</td>
<td>ia</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interlingue</td>
<td>ie</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inuktitut</td>
<td>iu</td>
<td></td>
<td>langInuktitut</td>
<td>143</td>
</tr>
<tr>
<td>Inupiaq</td>
<td>ik</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Irish</td>
<td>ga</td>
<td></td>
<td>langIrish</td>
<td>35</td>
</tr>
<tr>
<td>Italian</td>
<td>it</td>
<td></td>
<td>langItalian</td>
<td>3</td>
</tr>
<tr>
<td>Japanese</td>
<td>ja</td>
<td></td>
<td>langJapanese</td>
<td>11</td>
</tr>
<tr>
<td>Javanese</td>
<td>jw</td>
<td></td>
<td>langJavaneseRom</td>
<td>138</td>
</tr>
<tr>
<td>Kannada</td>
<td>kn</td>
<td></td>
<td>langKannada</td>
<td>73</td>
</tr>
<tr>
<td>Kashmiri</td>
<td>ks</td>
<td></td>
<td>langKashmir</td>
<td>61</td>
</tr>
<tr>
<td>Kazakh</td>
<td>kk</td>
<td></td>
<td>langKazakh</td>
<td>48</td>
</tr>
<tr>
<td>Kinyarwanda</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kirghiz</td>
<td>ky</td>
<td></td>
<td>langKirghiz</td>
<td>54</td>
</tr>
<tr>
<td>Kirundi</td>
<td>m</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Korean</td>
<td>ko</td>
<td></td>
<td>langKorean</td>
<td>23</td>
</tr>
<tr>
<td>Kurdish</td>
<td>ku</td>
<td></td>
<td>langKurdish</td>
<td>60</td>
</tr>
<tr>
<td>Laotian</td>
<td>lo</td>
<td></td>
<td>langLao</td>
<td>79</td>
</tr>
<tr>
<td>Lappish</td>
<td></td>
<td></td>
<td>langLappish, langSaamisk</td>
<td>29</td>
</tr>
<tr>
<td>Latin</td>
<td>la</td>
<td></td>
<td>langLatin</td>
<td>131</td>
</tr>
<tr>
<td>Latvian (Lettish)</td>
<td>lv</td>
<td>0x0026</td>
<td>langLatvian</td>
<td>28</td>
</tr>
<tr>
<td>Lingala</td>
<td>ln</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lithuanian</td>
<td>lt</td>
<td></td>
<td>langLithuanian</td>
<td>24</td>
</tr>
<tr>
<td>Macedonian</td>
<td>mk</td>
<td></td>
<td>langMacedonian</td>
<td>43</td>
</tr>
<tr>
<td>Malagasy</td>
<td>mg</td>
<td></td>
<td>langMalagasy</td>
<td>93</td>
</tr>
<tr>
<td>Malay</td>
<td>ms</td>
<td>0x003e</td>
<td>langMalayRoman(Latin), langMalayArabic(Arabic)</td>
<td>83(L), 84(A)</td>
</tr>
<tr>
<td>Malayalam</td>
<td>ml</td>
<td></td>
<td>langMalayalam</td>
<td>72</td>
</tr>
<tr>
<td>Maltese</td>
<td>mt</td>
<td></td>
<td>langMaltese</td>
<td>16</td>
</tr>
<tr>
<td>Manx Gaelic</td>
<td>gv*</td>
<td></td>
<td>langGailick</td>
<td>141</td>
</tr>
<tr>
<td>Maori</td>
<td>mi</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Marathi</td>
<td>mr</td>
<td></td>
<td>langMarathi</td>
<td>66</td>
</tr>
<tr>
<td>Moldavian</td>
<td>mo</td>
<td></td>
<td>langMoldavian</td>
<td>53</td>
</tr>
<tr>
<td>Mongolian</td>
<td>mn</td>
<td></td>
<td>langMongolian(Mongolian), langMongolianCyr(Cyrillic)</td>
<td>57(M), 58(C)</td>
</tr>
<tr>
<td>Nauru</td>
<td>na</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nepali</td>
<td>he</td>
<td></td>
<td>langNepali</td>
<td>64</td>
</tr>
<tr>
<td>Norwegian</td>
<td>no</td>
<td>0x0014</td>
<td>langNorwegian</td>
<td>9</td>
</tr>
<tr>
<td>Occitan</td>
<td>oc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oriya</td>
<td>or</td>
<td></td>
<td>langOriya</td>
<td>71</td>
</tr>
<tr>
<td>Oromo (Afan)</td>
<td>om</td>
<td></td>
<td>langOromo</td>
<td>87</td>
</tr>
<tr>
<td>Pashto (Pushto)</td>
<td>ps</td>
<td></td>
<td>langPashto</td>
<td>59</td>
</tr>
<tr>
<td>Language</td>
<td>ISO 639-1</td>
<td>ISO 639-2</td>
<td>ISO 639-3</td>
<td>Language Name</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------</td>
<td>------------</td>
<td>-----------</td>
<td>----------------</td>
</tr>
<tr>
<td>Polish</td>
<td>pl</td>
<td>0x0015</td>
<td>langPolish</td>
<td>25</td>
</tr>
<tr>
<td>Portuguese</td>
<td>pt</td>
<td>0x0016</td>
<td>langPortuguese</td>
<td>9</td>
</tr>
<tr>
<td>Punjabi</td>
<td>pa</td>
<td></td>
<td>langPunjabi</td>
<td>70</td>
</tr>
<tr>
<td>Quechua</td>
<td>qu</td>
<td></td>
<td>langQuechua</td>
<td>132</td>
</tr>
<tr>
<td>Rhaeto-Romance</td>
<td>rm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Romanian</td>
<td>ro</td>
<td>0x0018</td>
<td>langRomanian</td>
<td>97</td>
</tr>
<tr>
<td>Ruanda</td>
<td></td>
<td></td>
<td>langRuanda</td>
<td>90</td>
</tr>
<tr>
<td>Rundi</td>
<td></td>
<td></td>
<td>langRundi</td>
<td>91</td>
</tr>
<tr>
<td>Russian</td>
<td>ru</td>
<td>0x0019</td>
<td>langRussian</td>
<td>32</td>
</tr>
<tr>
<td>Samoan</td>
<td>sm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sangro</td>
<td>sg</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sanskrit</td>
<td>sa</td>
<td></td>
<td>langSanskrit</td>
<td>65</td>
</tr>
<tr>
<td>Scots Gaelic</td>
<td>gd</td>
<td></td>
<td>langGaidhlig</td>
<td>140</td>
</tr>
<tr>
<td>Serbian</td>
<td>sr</td>
<td>0x001a</td>
<td>langSerbian</td>
<td>42</td>
</tr>
<tr>
<td>Serbo-Croatian</td>
<td>sh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sesotho</td>
<td>st</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Setswana</td>
<td>tn</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shona</td>
<td>sn</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sindhi</td>
<td>sd</td>
<td></td>
<td>langSindhi</td>
<td>62</td>
</tr>
<tr>
<td>Singhalese</td>
<td>si</td>
<td></td>
<td>langSinhalese</td>
<td>76</td>
</tr>
<tr>
<td>Siswati</td>
<td>ss</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slovak</td>
<td>sk</td>
<td>0x001b</td>
<td>langSlovak</td>
<td>39</td>
</tr>
<tr>
<td>Slovenian</td>
<td>sl</td>
<td>0x0024</td>
<td>langSlovenian</td>
<td>40</td>
</tr>
<tr>
<td>Somali</td>
<td>so</td>
<td></td>
<td>langSomali</td>
<td>88</td>
</tr>
<tr>
<td>Spanish</td>
<td>es</td>
<td>0x000a</td>
<td>langSpanish</td>
<td>6</td>
</tr>
<tr>
<td>Sundanese</td>
<td>su</td>
<td></td>
<td>langSundaneseRom</td>
<td>139</td>
</tr>
<tr>
<td>Swahili</td>
<td>sw</td>
<td>0x0041</td>
<td>langSwahili</td>
<td>89</td>
</tr>
<tr>
<td>Swedish</td>
<td>sv</td>
<td>0x001d</td>
<td>langSwedish</td>
<td>5</td>
</tr>
<tr>
<td>Tagalog</td>
<td>tl</td>
<td></td>
<td>langTagalog</td>
<td>82</td>
</tr>
<tr>
<td>Tajik</td>
<td>tg</td>
<td></td>
<td>langTajiki</td>
<td>55</td>
</tr>
<tr>
<td>Tamil</td>
<td>ta</td>
<td></td>
<td>langTamil</td>
<td>74</td>
</tr>
<tr>
<td>Tatar</td>
<td>tt</td>
<td></td>
<td>langTatar</td>
<td>135</td>
</tr>
<tr>
<td>Telugu</td>
<td>te</td>
<td></td>
<td>langTelugu</td>
<td>75</td>
</tr>
<tr>
<td>Thai</td>
<td>th</td>
<td>0x001e</td>
<td>langThai</td>
<td>22</td>
</tr>
<tr>
<td>Tibetan</td>
<td>bo</td>
<td></td>
<td>langTibetan</td>
<td>63</td>
</tr>
<tr>
<td>Tigrinya</td>
<td>ti</td>
<td></td>
<td>langTigrinya</td>
<td>86</td>
</tr>
<tr>
<td>Tonga</td>
<td>to</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tsonga</td>
<td>ts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Turkish</td>
<td>tr</td>
<td>0x001f</td>
<td>langTurkish</td>
<td>17</td>
</tr>
<tr>
<td>Turkmen</td>
<td>tk</td>
<td></td>
<td>langTurkmen</td>
<td>56</td>
</tr>
<tr>
<td>Twi</td>
<td>tw</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uighur</td>
<td>ug</td>
<td></td>
<td>langUighur</td>
<td>136</td>
</tr>
<tr>
<td>Ukrainian</td>
<td>uk</td>
<td>0x0022</td>
<td>langUkrainian</td>
<td>45</td>
</tr>
<tr>
<td>Urdu</td>
<td>ur</td>
<td>0x0020</td>
<td>langUrdu</td>
<td>20</td>
</tr>
<tr>
<td>Uzbek</td>
<td>uz</td>
<td></td>
<td>langUzbek</td>
<td>47</td>
</tr>
<tr>
<td>Vietnamese</td>
<td>vi</td>
<td>0x002a</td>
<td>langVietnamese</td>
<td>80</td>
</tr>
<tr>
<td>Volapük</td>
<td>vo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Welsh</td>
<td>cy</td>
<td></td>
<td>langWelsh</td>
<td>128</td>
</tr>
<tr>
<td>Wolof</td>
<td>wo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Xhosa</td>
<td>xh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>Letter 1</td>
<td>Letter 2</td>
<td>Proposed</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>---------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>Yiddish</td>
<td>i, yi</td>
<td>langYiddish</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>Yoruba</td>
<td>yo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zulu</td>
<td>zu</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- proposed
## APPENDIX E

<table>
<thead>
<tr>
<th>Locale Description</th>
<th>Short String</th>
<th>Hex Value</th>
<th>Decimal Value</th>
<th>Locale Description</th>
<th>Short String</th>
<th>Hex Value</th>
<th>Decimal Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afrikaans</td>
<td>af</td>
<td>0x0436</td>
<td>1078</td>
<td>Hindi</td>
<td>hi</td>
<td>0x0439</td>
<td>1081</td>
</tr>
<tr>
<td>Albanian</td>
<td>sq</td>
<td>0x041C</td>
<td>1052</td>
<td>Hungarian</td>
<td>hu</td>
<td>0x040E</td>
<td>1038</td>
</tr>
<tr>
<td>Arabic - U.A.E.</td>
<td>ar-ae</td>
<td>0x3801</td>
<td>14337</td>
<td>Icelandic</td>
<td>is</td>
<td>0x040F</td>
<td>1039</td>
</tr>
<tr>
<td>Arabic - Bahrain</td>
<td></td>
<td>0x3C01</td>
<td>15361</td>
<td>Indonesian</td>
<td>in</td>
<td>0x0421</td>
<td>1057</td>
</tr>
<tr>
<td>Arabic - Algeria</td>
<td>ar-dz</td>
<td>0x1401</td>
<td>5121</td>
<td>Italian - Standard</td>
<td>it</td>
<td>0x0410</td>
<td>1040</td>
</tr>
<tr>
<td>Arabic - Egypt</td>
<td>ar-eg</td>
<td>0x0C01</td>
<td>3073</td>
<td>Italian - Switzerland</td>
<td>it-ch</td>
<td>0x0810</td>
<td>2064</td>
</tr>
<tr>
<td>Arabic - Iraq</td>
<td>ar-iq</td>
<td>0x0801</td>
<td>2049</td>
<td>Japanese</td>
<td>ja</td>
<td>0x0411</td>
<td>1041</td>
</tr>
<tr>
<td>Arabic - Jordan</td>
<td>ar-jo</td>
<td>0x2C01</td>
<td>11265</td>
<td>Korean</td>
<td>ko</td>
<td>0x0412</td>
<td>1042</td>
</tr>
<tr>
<td>Arabic - Kuwait</td>
<td>ar-kw</td>
<td>0x3401</td>
<td>13313</td>
<td>Latvian</td>
<td>lv</td>
<td>0x0426</td>
<td>1062</td>
</tr>
<tr>
<td>Arabic - Lebanon</td>
<td></td>
<td>0x3001</td>
<td>12289</td>
<td>Lithuanian</td>
<td>lt</td>
<td>0x0427</td>
<td>1063</td>
</tr>
<tr>
<td>Arabic - Libya</td>
<td>ar-ly</td>
<td>0x1001</td>
<td>4097</td>
<td>Macedonian</td>
<td>mk</td>
<td>0x042F</td>
<td>1071</td>
</tr>
<tr>
<td>Arabic - Morocco</td>
<td>ar-ma</td>
<td>0x1801</td>
<td>6145</td>
<td>Malay - Malaysia</td>
<td>ms</td>
<td>0x043E</td>
<td>1086</td>
</tr>
<tr>
<td>Arabic - Oman</td>
<td>ar-om</td>
<td>0x2001</td>
<td>8193</td>
<td>Maltese</td>
<td>mt</td>
<td>0x043A</td>
<td>1082</td>
</tr>
<tr>
<td>Arabic - Qatar</td>
<td>ar-ga</td>
<td>0x4001</td>
<td>16385</td>
<td>Norwegian - Bokmål</td>
<td>no</td>
<td>0x0414</td>
<td>1044</td>
</tr>
<tr>
<td>Arabic - Saudiia</td>
<td>ar-sa</td>
<td>0x0401</td>
<td>1025</td>
<td>Polish</td>
<td>pl</td>
<td>0x0415</td>
<td>1045</td>
</tr>
<tr>
<td>Arabic - Syria</td>
<td>syr</td>
<td>0x2801</td>
<td>10241</td>
<td>Portuguese - Standard</td>
<td>pt</td>
<td>0x0816</td>
<td>2070</td>
</tr>
<tr>
<td>Arabic - Tunisia</td>
<td>ar-tn</td>
<td>0x1C01</td>
<td>7169</td>
<td>Portuguese - Brazil</td>
<td>pt-br</td>
<td>0x0416</td>
<td>1046</td>
</tr>
<tr>
<td>Arabic - Yemen</td>
<td>ar-ye</td>
<td>0x2401</td>
<td>3217</td>
<td>Raeto-Romance</td>
<td>rm</td>
<td>0x0417</td>
<td>1047</td>
</tr>
<tr>
<td>Basque</td>
<td>eu</td>
<td>0x042D</td>
<td>1069</td>
<td>Romanian</td>
<td>ro</td>
<td>0x0418</td>
<td>1048</td>
</tr>
<tr>
<td>Belarusian</td>
<td>be</td>
<td>0x0423</td>
<td>1059</td>
<td>Romanian - Moldova</td>
<td>ro-mo</td>
<td>0x0818</td>
<td>2072</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>bg</td>
<td>0x0402</td>
<td>1026</td>
<td>Russian</td>
<td>ru</td>
<td>0x0419</td>
<td>1049</td>
</tr>
<tr>
<td>Catalan</td>
<td>ca</td>
<td>0x0403</td>
<td>1027</td>
<td>Russian - Moldova</td>
<td>ru-mo</td>
<td>0x0819</td>
<td>2073</td>
</tr>
<tr>
<td>Chinese - PRC</td>
<td>zh-cn</td>
<td>0x0804</td>
<td>2052</td>
<td>Serbian</td>
<td>sr</td>
<td>0x0C1A</td>
<td>3098</td>
</tr>
<tr>
<td>Chinese - Hong Kong</td>
<td>zh-hk</td>
<td>0x0C04</td>
<td>3076</td>
<td>Setsuana</td>
<td>sn</td>
<td>0x0432</td>
<td>1074</td>
</tr>
<tr>
<td>Chinese - Singapore</td>
<td>zh-sg</td>
<td>0x1004</td>
<td>4100</td>
<td>Slovak</td>
<td>sk</td>
<td>0x041B</td>
<td>1051</td>
</tr>
<tr>
<td>Chinese - Taiwan</td>
<td>zh-tw</td>
<td>0x0404</td>
<td>1028</td>
<td>Sorbian</td>
<td>st</td>
<td>0x042E</td>
<td>1070</td>
</tr>
<tr>
<td>Croatian</td>
<td>hr</td>
<td>0x041A</td>
<td>1050</td>
<td>Spanish - Standard</td>
<td>es</td>
<td>0x040A</td>
<td>1034</td>
</tr>
<tr>
<td>Czech</td>
<td>cs</td>
<td>0x0405</td>
<td>1029</td>
<td>Spanish - Argentina</td>
<td>es-ar</td>
<td>0x2C0A</td>
<td>11274</td>
</tr>
<tr>
<td>Danish</td>
<td>da</td>
<td>0x0406</td>
<td>1030</td>
<td>Spanish - Bolivia</td>
<td>es-bo</td>
<td>0x400A</td>
<td>16394</td>
</tr>
<tr>
<td>Dutch</td>
<td>nl</td>
<td>0x0413</td>
<td>1043</td>
<td>Spanish - Chile</td>
<td>es-cl</td>
<td>0x340A</td>
<td>13322</td>
</tr>
<tr>
<td>Dutch - Belgium</td>
<td>nl-be</td>
<td>0x0813</td>
<td>2067</td>
<td>Spanish - Colombia</td>
<td>es-co</td>
<td>0x240A</td>
<td>9226</td>
</tr>
<tr>
<td>Dutch - Australia</td>
<td>en-au</td>
<td>0x0C09</td>
<td>3081</td>
<td>Spanish - Costa</td>
<td>es-cr</td>
<td>0x140A</td>
<td>5130</td>
</tr>
<tr>
<td>English - Belize</td>
<td>en-bz</td>
<td>0x2809</td>
<td>10249</td>
<td>Spanish - Republic</td>
<td>es-do</td>
<td>0x1C0A</td>
<td>7178</td>
</tr>
<tr>
<td>English - Canada</td>
<td>en-ca</td>
<td>0x1009</td>
<td>4105</td>
<td>Spanish - Ecuador</td>
<td>es-ec</td>
<td>0x300A</td>
<td>12298</td>
</tr>
<tr>
<td>English - Ireland</td>
<td>en-ie</td>
<td>0x1809</td>
<td>6153</td>
<td>Spanish - Guatemala</td>
<td>es-gt</td>
<td>0x100A</td>
<td>4106</td>
</tr>
<tr>
<td>English - Jamaica</td>
<td>en-jm</td>
<td>0x2009</td>
<td>8201</td>
<td>Spanish - Honduras</td>
<td>es-hn</td>
<td>0x480A</td>
<td>18442</td>
</tr>
<tr>
<td>English - New Zealand</td>
<td>en-nz</td>
<td>0x1409</td>
<td>5129</td>
<td>Spanish - Mexico</td>
<td>es-mx</td>
<td>0x080A</td>
<td>2058</td>
</tr>
<tr>
<td>English - South America</td>
<td>en-za</td>
<td>0x1C09</td>
<td>7177</td>
<td>Spanish - Nicaragua</td>
<td>es-ni</td>
<td>0x4C0A</td>
<td>19466</td>
</tr>
<tr>
<td>English - Trinidad</td>
<td>en-td</td>
<td>0x2C09</td>
<td>11273</td>
<td>Spanish -</td>
<td>es-pa</td>
<td>0x180A</td>
<td>6154</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Kingdom</th>
<th>Language - Country</th>
<th>Code</th>
<th>Bidi</th>
<th>ISO 639-1</th>
<th>Bidi</th>
<th>Bidi</th>
<th>Bidi</th>
</tr>
</thead>
<tbody>
<tr>
<td>English - United States</td>
<td>Spanish - Peru</td>
<td>es-pe</td>
<td>0x280A</td>
<td>10250</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Estonian</td>
<td>Spanish - Puerto Rico</td>
<td>es-pr</td>
<td>0x500A</td>
<td>20490</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Farsi</td>
<td>Spanish - Paraguay</td>
<td>es-py</td>
<td>0x3C0A</td>
<td>15370</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finnish</td>
<td>Spanish - El Salvador</td>
<td>es-sv</td>
<td>0x440A</td>
<td>17418</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Faeroese</td>
<td>Spanish - Uruguay</td>
<td>es-uy</td>
<td>0x380A</td>
<td>14346</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>French - Standard</td>
<td>Spanish - Venezuela</td>
<td>es-ve</td>
<td>0x200A</td>
<td>8202</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>French - Belgium</td>
<td>Estonian</td>
<td>et</td>
<td>0x0429</td>
<td>1061</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>French - Canada</td>
<td>Estonian</td>
<td>fa</td>
<td>0x0425</td>
<td>1036</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>French - Luxembourg</td>
<td>Estonian</td>
<td>fi</td>
<td>0x040B</td>
<td>1035</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>French - Switzerland</td>
<td>Estonian</td>
<td>fo</td>
<td>0x0438</td>
<td>1080</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>French - Belgium</td>
<td>Estonian</td>
<td>fr</td>
<td>0x040C</td>
<td>1036</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>French - Canada</td>
<td>Estonian</td>
<td>fr-be</td>
<td>0x080C</td>
<td>2060</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>French - Luxembourg</td>
<td>Estonian</td>
<td>fr-ca</td>
<td>0x0C0C</td>
<td>3084</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>French - Switzerland</td>
<td>Estonian</td>
<td>fr-ch</td>
<td>0x140C</td>
<td>5132</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>German - Standard</td>
<td>Estonian</td>
<td>fr-lu</td>
<td>0x100C</td>
<td>4108</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>German - Austria</td>
<td>Estonian</td>
<td>gd</td>
<td>0x043C</td>
<td>1084</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>German - Lichtenstein</td>
<td>Estonian</td>
<td>de</td>
<td>0x0407</td>
<td>1031</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>German - Luxembourg</td>
<td>Estonian</td>
<td>de-at</td>
<td>0x0C07</td>
<td>3079</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>German - Switzerland</td>
<td>Estonian</td>
<td>de-li</td>
<td>0x1407</td>
<td>5127</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Greek</td>
<td>Estonian</td>
<td>de-lu</td>
<td>0x1007</td>
<td>4103</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hebrew</td>
<td>Estonian</td>
<td>el</td>
<td>0x0408</td>
<td>1032</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hebrew</td>
<td>Estonian</td>
<td>he</td>
<td>0x040D</td>
<td>1037</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX F

GLOSSARY

ACP:
Acronym for the ANSI Codepage in use. Windows NT uses this codepage to convert to/from Unicode automatically.

Active X:
Microsoft technology that allows data to be shared among different applications.

ANSI (see Extended ASCII)

American Standard Code for Information Interchange (ASCII):
Coding scheme based on 7-bit byte allowing 128 different characters to be encoded.

Bidirectional languages:
Languages that generally read from right to left. However, some non-English text and numbers are read from left-to-right. These languages include Arabic, Hebrew, and Thai.

Bitmap (.bmp):
Graphic in which the colour of the pixels is defined by one or more bits.

Character:
The simplest element used to represent written languages.

Character encoding:
A one-to-one mapping from a set of characters into a set of numbers, used to represent text in software.

Computer Assisted Translation (CAT):
Computer technology applications that automates and assists the translation process by providing tools such as Translation Memory and Terminology databases.

Codepage:
Small piece of code in an application that tells your operating system where to look up particular characters. Ordered set of characters in which a numeric index is associated with each character. The first 128 characters of each code page are functionally the same and include all characters needed to type English text.

Composite String:
An error message or other text that is dynamically created and presented to the user in sentence form.

Copyfitting:
Process whereby localized printed materials have the same number of pages as the source material, and follow the set of rules in any documentation that defines things such as on which page each chapter starts, whether or not paragraphs are broken and so on.

Double-Byte:
A character defined with 2 bytes (16 bits) instead of 1 byte (8 bits).

Double-Byte Enabling (DBE):
Reengineering an original product’s source code to support the input, display, and manipulation of double-byte language character sets.

Dynamic Link Library (DLL):
A file that contains executable functions or data for applications.

Extended ASCII:
Coding scheme based on 8-bit byte allowing 256 characters to be encoded.

**Font:**
A collection of glyphs for a displaying text in a particular typeface.

**Globalisation:**
Process of designing a product so that it can be sold anywhere in the world with minimum changes.

**Glyph:**
A graphical representation of a character.

**Internationalisation:**
Process of engineering a product so it can be easily and efficiently localised.

**Leverage:**
Building current translation projects on those previously finished. Using one translation for repeated sections of text.

**Locale:**
A set of user-preference information related to the user’s language and sub-language. The user locale can be different from the default system locale.

**Localisation:**
Way of integrating the whole of a website or any other software product cohesively into the language and culture of the target group. Conversion of the User Interface and supporting materials of a product from its original state to fit the language and cultural nuances of a target locale.

**Master language:**
It refers to the language of the core operating system components. The master language can not be removed.

**Multi-lingual User Interface (MUI):**
Codename for a separate release of Windows 2000 called Windows 2000 Multilanguage Version. In this version of Windows 2000, the UI language can be changed according to the preferences of individual users. The user can then select the UI language or set it by using Group Policy for Organizational Units. It also allows users of different languages to share the same workstation.

**Native Encoding:**
Individual standards for a given language or set of languages, such as ANSI for English.

**Quality Assurance:**
Process to ensure that the target document resembles as closely as possible the source document.

**Resource Files:**
Source files that contain information to be compiled into the program. They contain the parts of the application that are seen by the user.

**Sim-Ship (simultaneous shipment):**
Releasing all language versions of a particular product at the same time.

**Strings:**
Groupings of characters that are used in programs. They are often enclosed in single or double quotes.

**System Locale:**
Determines which script non-Unicode applications will support. It is the locale emulated by the system. To change the system it is necessary to reboot.
Thread Locale:
Locale of a given thread. Gets inherited upon creation from a User Locale and can be changed at run time to any valid locale (per thread). APIs can use this locale to format numbers, date, time…

Translation Memory:
A database of sentences and phrases and their translated versions, built from previous translations.

Unicode:
Attempt to rationalize the large number of codepage standards that exist for different character sets.

UCS (Universal Character Set):

UI Language:
Language in which the system displays its menus, help files, error messages and dialog boxes.

User Locale:
The user preferences for formatting dates, currencies, numbers, etc.

UTF-8:
Subset of Unicode commonly used as a code page and is used on the Web. Each 16-bit Unicode character is encoded as a sequence of one, two, or three 8-bit bytes, depending on the value of the character.