# Localization & Internationalization Testing

**Shanthi.AL:**Shanthi.Alagappan@cognizant.com

---

**Cognizant Technology Solutions India Private Ltd,**
**63/1&2, Old Mahabalipuram Road,**
**Navallur, Chennai**
**P.O 603 103**

---

**6th Annual International Software Testing Conference in India 2006**

**Delhi: February 17-18, 2006**

# 1. Abstract

Internationalization and Localization are subsets of Globalization. Internationalization “i18n” refers to the process of designing, developing and engineering the product that can be adaptable to various locales and regions without further any engineering changes. Today the world is a global village, the products are developed in one remote of the world, undergo globalization process, launched in multiple markets and used in different remotes of the world. As a consequence, the need of internationalization and localization process and testing requirement for the internationalized product is considerably increased.
This paper

- provides insight on internationalization and localization process
- signifies the importance of effective test plan and strategy
- briefs about the resource bundle and i18n test environment
- provides case study for i18n test data and test case design
- Over view of i18n and L10N testing
- Provides check lists for i18n test life cycle.

## 2 Introductions

### 2.1 Internationalization

Internationalization is the process where the code of the software is modified so that it is completely independent of any culture specific information. The hard –coded strings of the software will be pulled out and stored in external files that are called as resource bundles and these will load at runtime. I18n process typically involves the following tasks.

- Externalizing of strings, graphics, icons, texts etc.
- Selecting code page and defining code page conversions
- Modifying all the text manipulation functions to be aware of the code page.
- Changing the logic of all the formatting functions (Date, Time, Currency, Numeric, etc)
- Changing the Collation /sorting functions

**Note:** Code page is nothing but assigning a specific number to each character in a language in order to handle text. It is a mapping table of characters to its numeric value. **ASCII** is a good example of code page.

### 2.2 Common Culture Specific Information:

- Externalization of strings: No string should be hard wired to the code. It should be externalized to a resource file so that it can be translated to the required language and can be applied during run time.
- Date and Time Formatting: Month, Day, Year formats supported as to which comes first?
- Numeric and Currency formatting: Currency symbols and how grouping the Numbers differ in each country?
- Collation /Sorting Order: Sorting order will change depending upon the native language. Specific rules will be defined for sorting process, based on the code-page used.

### 2.3 What is a Locale?

A locale consists of basic components such as language, territory and code page. The main objective of "i18n" is to externalize all "cultural specific information" from the code which means this data is to be loaded at run time so that, the software will behave appropriately based on the locale set /installed to the client machine.

### 2.4 How to set the locale to client machine?

- Install the native language version of O/S in the client machine [Ex: Win2000 Japanese version] **OR**
- **For Windows 2000:**
  1. Go to Regional Options, by going to start menu ->Settings -> Control Panel -> Regional Options
  2. In the General Tab, under the "Settings for the current user" panel, choose the appropriate locale

### 2.5 Localization

It is the process of customizing the software product for each language that is to be supported. It is the aspect of development and testing relating to the translation of the software and its presentation to the end user. It includes translating the program, choosing the appropriate icons and graphics and other cultural considerations. It also may include the translation of help files and documentation. Localization is abbreviated as L10N, which means that 'L' and 'N' are separated by 10 characters.

## 3. Effective Test Strategy and Test Plan for i18n testing:

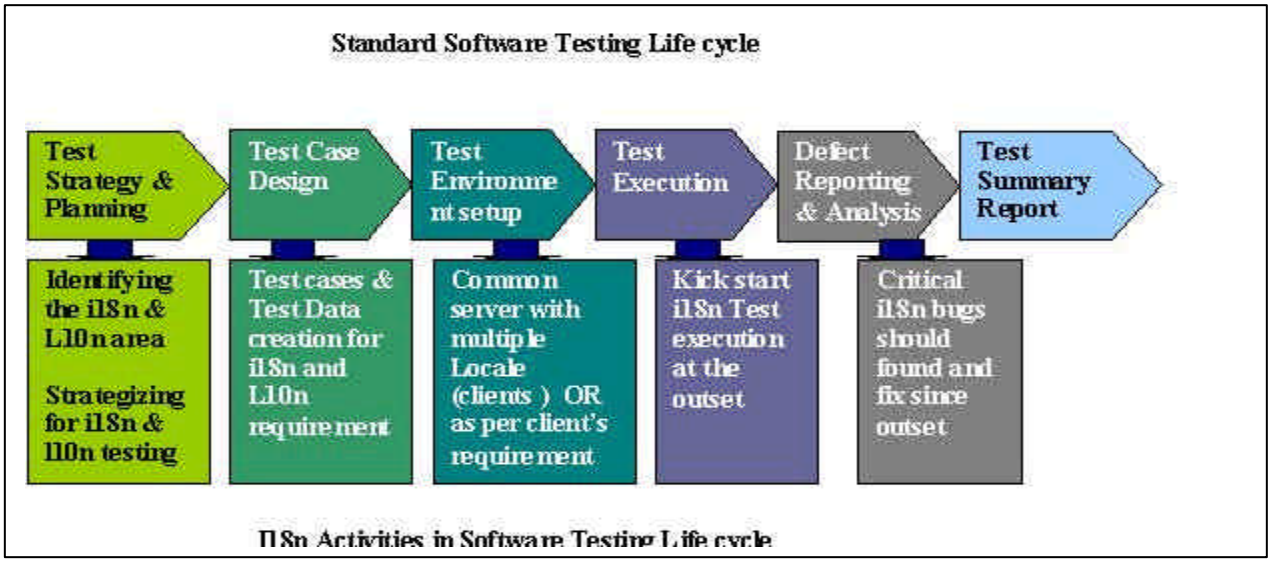
Simultaneous release is important in a world where, instant communication is needed worldwide. In order to maximize the market benefit, and to show of commitment to regional markets, clients would want to release their product to multiple markets, on time, simultaneously. In consideration of this, the test plan and strategy should focus on

- Kick start of I18n and L10N testing since day one, the day when the regular testing starts on base product (English)
- Defining a suitable strategy for i18n testing to pull out all i18n related critical bugs at the outset.
- Planning to fix, regress the bugs and its impact on i18n environment since outset.

As a result, a flawless product can be released to multiple markets, simultaneously.

The following picture illustrates the i18n activities that need to be done during software test life cycle process.

.



### 3.1 Test Approach would cover the following:

- Resource Bundle loading from server pertaining to the locale setting
- User Interface display pertaining to the locale setting
- Culture specific date, time, monetary formats, collate and sorting order, numeric,
- Customized colors and fonts on the web server (web –based application)
- Language specific character set render
- Technology specific i18n support validation
- Product stability by validating functionalities and business rules

## 4. Test case and Test data design:

### 4.1 Test case and test data needs to be created to

- Verify the Language specific translated string or resource files are loaded by the application, depending on the current language and locale settings (client O/S)
- Verify the Language specific translated strings rendered on user interface and error messages.
- Verify the culture specific Date /time, sort order, numeric and monetary formats, collate and sort are displayed according to locale.
- Verify the customized colors and fonts on the web server (Globalization)
- Verify the product stability by inputting various test data specific to required language.

**Case study:** I worked for i18n support web-based email product and involved in test data preparation. The test cases and test data are created for MIME to ensure if the various language (Japanese, Chinese, Korean, German, and French) char sets are encoded and decoded properly when sending email on the specified language.
To go further in detail, we would just go through the MIME in brief…

**MIME:** The Multi-purpose Internet Mail Extensions (MIME) is the Internet standard for sending e-mail messages that contain non US-ASCII character sets, enriched text, GIF images, and other types of files such as multimedia including audio and video. The Distinct MIME ActiveX control allows you to easily build applications that have MIME encoding and decoding capabilities. It can be used to build a stand-alone encoder, or used in conjunction with the IMAP4, SMTP, POP2/POP3 and NNTP ActiveX controls to send, receive and post encoded e-mail or news messages. The MIME ActiveX control supports RFC 822 (plain messages), MIME conformant Base 64 and Quoted printable, Binhex, as well as uuencode and uudecode. It can encode or decode any of these file formats.
(*Courtesy: Multipart Internet Mail Extensions Protocol ActiveX control for Microsoft windows. Version 5.2)*

I here by give a sample for Japanese test data to test the MIME part with various char sets for receiving messages (receives messages in the application)
MIME is an internet standard for email messages which, plays key role for i18n products. Our product Supports both Iso-2022-jp and UTF-8 char sets. MIME encoding and decoding char set for Japanese text is ISO-2022-jp and UTF-8 and the MIME format are Quoted Printable and Base 64.
That is MIME would use a char set ISO-2022-JP or UTF-8 (feature to set the char set at front –end of the Application) using the format QP or Base-64 to render (encode and decode) the Japanese text messages that sent via mail server.

The following are the test data for the above scenario (picture 4)


## 5. I18n Test Environment Setup:

To get in to setting up test environment, we would just know about the resource bundle?

### 5.1 What's Resource bundle? http://java.sun.com/developer/TechTips/1998/tt0521.html

A resource bundle contains locale-specific objects, for example strings representing messages to be displayed in the application. The idea is to load a specific bundle of resources, based on a particular locale. The following are the sample resource bundle properties files.

```
# German greeting files (greet_de.properties)

  Morn=Guten Morgan

  # English greeting file (greet_en.properties)
```

Morn=Good morning

The lower case ‘de”,’ja’ and ‘en” represents the language

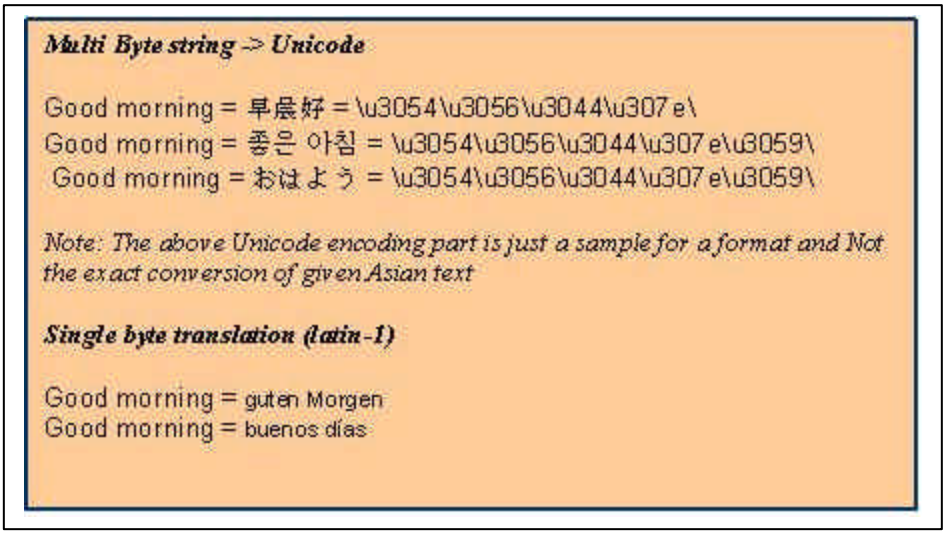
### 5.2 How the program understands the language?

Program or code will understand only the Latin-1 and Unicode encoded characters.
The following activities should be followed to let the program understand the language properties files.

- All the extracted strings should be locally translated.( Required Asian and western languages)
- Ensure if the strings that are translated in to Western European languages contain single byte characters (German , French)
- Ensure if the strings that are translated in to Asian languages contain double byte characters (Japanese , Chinese and Korean)
- Latin-1 covers most of the Western European languages and hence the western European translated strings need not to make any conversion as the program can understand Latin-1.
- Double byte translated strings should be converted in to Unicode encoded characters
- Java development kit version 1.1(all above versions) contains a conversion tool
- Native2ascii which, is used for converting the multi bytes in to Unicode.
- The program will have a code to access the resource bundle which would understand based on the local
- name “de” ,”ja” represented to properties file

Note: JDK must be installed in the test environment for the conversion.

Picture 5 shows the sample of Multi byte and Unicode notations




### 5.3 I18n Installer:

The installer package will contain the resource bundle, batch file (for conversion). While executing the installer package, the batch would run internally, the multi byte strings will be converted in to Unicode notation and finally, the converted and Latin-1 files (server files, client files and link files) will be placed in to appropriate folders. Some installers would automatically place the resource files in proper folder and some installers, the user need to place the files in proper folders. This depends on the program written in the installer.

The application will be installed on the common (base –English) server. After installation, the resource files are placed in appropriate folders in the common server. When the user accesses the application from different locales

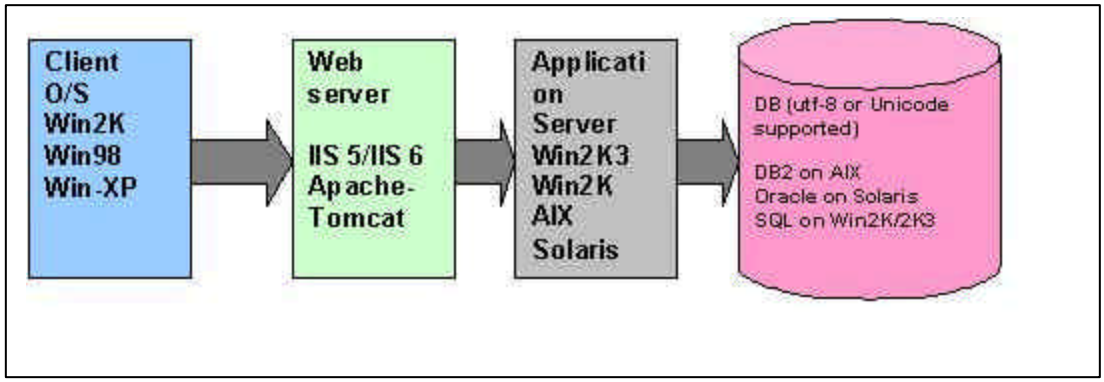
(client O/S),the corresponding resource files will be extracted from server side and required client properties files will be loaded in appropriate folders at client locale.

Eventually the application will be displayed pertaining to the locale (client O/S)

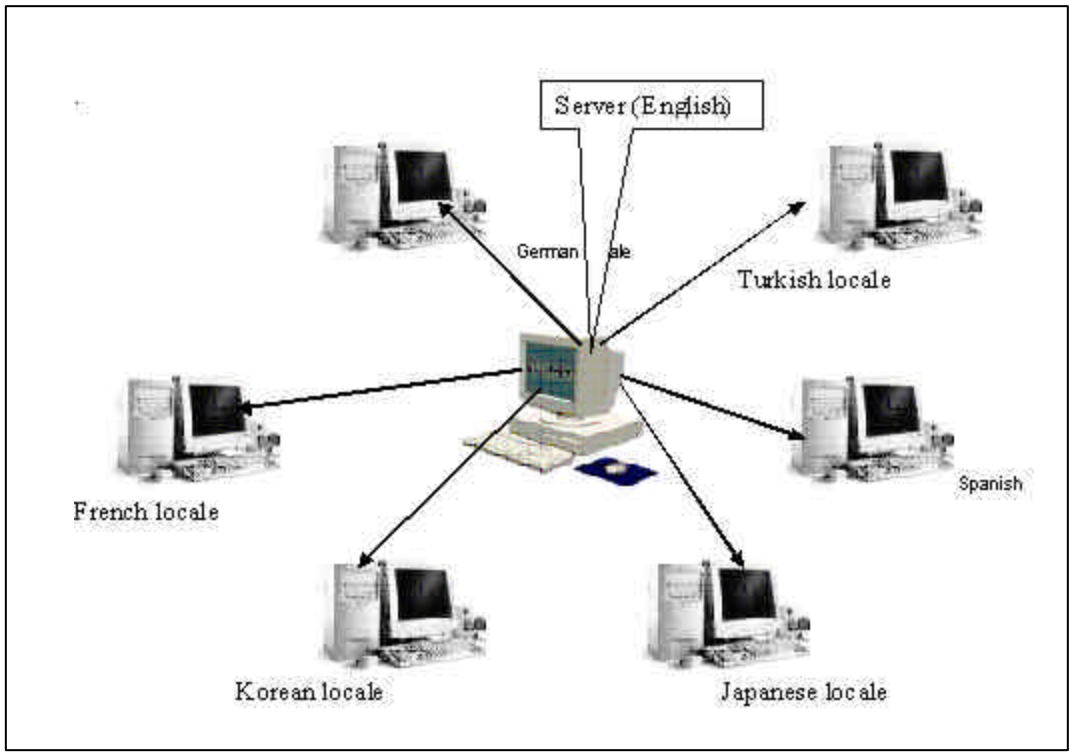
### 5.4 Test Environment:

The test bed can be set based on the client's requirements. A dedicated server, data base and clients versions are used in the specific language for setting up a test environment. OR a common server with data base can be set up with multiple clients. Please refer the following pictures

Picture 5.4a illustrates a web based i18n test environment




The following picture 5.4b illustrates a sample i18n test bed set up




**Reference:** Please refer the check list provided in section 5, to ensure the important points during setting up the test environment

# 3 I18N and L10N –Testing

## 3.1 I18N –Testing

The focus of i18n testing is as follows:

- **Compatibility testing:** Testing the product for language compatibility, this includes testing the product behavior in identifying and initializing from its language environment and its ability to customize to that environment.
- **Functionality testing:** Functionality testing is the core area of i18n testing .It typically includes running the whole functionality regression test on different language environments and exercising the interface with native language strings. It involves verifying the culture specific information such as date/time display.
- **User interface Validation:** To check for visual problems such as text truncation or overlap, graphics issues or other visual problems.
- **Interoperability testing**: It ensures that the software interacts properly with targeted platforms, operating systems, applications (and versions) and so on.
- **Usability testing:** It evaluates the ease of use of the system *(optional)*
- **Installation testing:** Testing to ensure if the product installation messages are displayed in a corresponding language when installing the application on a dedicated server

## 3.2 L10N Testing

Localization testing is a language verification testing mainly focusing on the appropriateness of the translation in the following items.

- GUI context
- Online help files
- Error messages, Dialog boxes
- Tutorials/Readme files
- Documents such as User manual, Installation guide, Release notes etc,

Testing also involves in checking the GUI layout, and making sure nothing is truncated in the UI and Correctness and consistency of the Error messages.

***Note**: L10N testing is typically done by the native speaker of that specific language*

# 4 I18N testing - Check List

| SN | Description | Tick | | | Remarks |
|---|---|---|---|---|---|
| | | **Yes** | **No** | **NA** | |
| **a) Environment Setup** | | | | | |
| 1.1 | Ensure the configuration [hardware and software] details from the Test Plan /Source for the i18n testing environment | | | | |

| | | | | | |
|---|---|---|---|---|---|
| 1.2 | Ensure the required native language/English version of O/S is installed in Server boxes? | | | | |
| 1.3 | Ensure the required Service Pack of native language version applied on O/S [For Windows] | | | | |
| 1.4 | Ensure that the required version of IIS /Apache –Tomcat is running on the server box where the Web Server is being installed? | | | | |
| 1.5 | Ensure that the correct version of Oracle client/DB2 client/SQL client [or] JDBC driver is installed on the Server box? | | | | |
| 1.6 | Ensure that the Database, which is going to be used, is Unicode/UTF-8 supported? | | | | Database should be able to accept all the language characters and retrieve the same. It should be Unicode compatible |
| 1.7 | Set the environmental variable for the local language, in the App server? [Optional Oracle Database] | | | | For example, Japanese installations should set the NLS_LANG Environment variable value to Japanese_Japan.JA16SJIS. |
| 1.8 | Ensure that the required locale is being installed on the client machine. | | | | |
| 1.9 | Ensure that the required service pack version is installed on the client machine [optional] | | | | |
| **b) I18n Testing** | | | | | |
| 2.1 | Have you applied the Resource bundles to a specified path for Application Server and Web server option? | | | | |
| 2.2 | Check the number of language properties files are available in the resource bundles | | | | |
| 2.2 | Verify that the user interface of the whole application is displayed in native language strings corresponding to the client locale | | | | |
| 2.3 | Verify that the user interface is displayed by default in English, when accessing from different client locale environment. | | | | Access the application from the French locale client where the properties files are not available in the resource |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | bundle |
| 2.4 | Verify that the user interface screens of the product are displayed in the other supported languages [language properties files from the resource bundle] corresponding to the relevant locale is installed on the client machine. | | | | |
| 2.5 | Validate the data handling capability of the application using mixture of native language character set, accented characters, ASCII characters and special characters | | | | Example: use the combination of Hiragana, katakana, Kanji, alphanumeric, special characters, accented characters etc for data validation in Japanese environment. For other environments use the applicable parameters |
| | Check if the validated inputs are rendered as original in the UI | | | | |
| 2.6 | Verify if the collation / sorting operations are functioning based on the client locale. | | | | |
| 2.7 | Verify if the filtering and searching operations are functioning based on the client locale. | | | | |
| 2.8 | Address order display differs from language to language. For example in Japanese the order will be postal code, state, city and name. For English name, city, state, and postal code in the order of display. | | | | |
| **c) Locale Testing** | | | | | |
| 3.1 | Verify that the Date and Time format displayed across the application is based on the client locale<br>Verify if the validation of date values handled are using double –byte numbers. | | | | Ex: Japanese date format is yyyy / mm / dd |
| 3.2 | Verify if the Currencies and calculations are using double-byte numbers. | | | | |
| 3.3 | Verify if the number formatting in the application for telephone numbers and pin codes, etc is based on the client locale. | | | | |

| | | | | | |
|---|---|---|---|---|---|
| 3.4 | Verify if the cursor / prompt is aligning on the right side of the text fields in the UI, corresponding to the right to left locale installed on the client | | | | Ex: Arabic, Hebrew |
| **d) Functions** | | | | | |
| 4.1 | Are all the specified customer needs requirements being tested? | | | | |
| 4.2 | Are all the inputs / outputs specified for each function tested? | | | | |
| 4.3 | Are all the input / output boundary conditions tested? | | | | |
| 4.4 | Are all the functionalities tested using the native language inputs? | | | | For Ex: using multi-byte characters for login names, Password, object names, Japanese content attachments with Japanese name, etc. |
| 4.5 | Are all the default input / output values tested? | | | | |
| 4.6 | Have the invalid input values been tested? | | | | |
| 4.7 | Is all the security requirements specified for each function tested? | | | | |
| 4.8 | Are the entire database requirements specified for each function tested? | | | | |
| 4.9 | Are all the events and states of each function specified tested? | | | | |
| **e) Localization** | | | | | |
| 5.1 | Are all user interfaces specified tested? | | | | |
| 5.2 | Do all the user interface screens display in the native language corresponding to the client locale? | | | | |
| 5.3 | Verify that there are no junk characters present in the strings on the screens | | | | |
| 5.4 | Verify that the native language characters are not getting overlapped | | | | |
| 5.5 | Confirm if the icons, images, graphics displayed in the UI are appearing correctly and are not broken | | | | |

| | | | | | |
|---|---|---|---|---|---|
| 5.6 | Confirm if the contents in the UI screens are displayed in the standard / traditional fonts. | | | | |
| 5.7 | Confirm if the tool tips / window status / title / alert messages displayed are based on the client locale. | | | | |
| 5.8 | Testing Mnemonics and short cut keys | | | | |
| 5.9 | Confirm if the help files are displayed in the native language corresponding to the client locale. | | | | If the help files are not translated in the required language, it should display the default English files |

## Reference:


http://takeoba.cool.ne.jp/java/native2ascii.htm
http://java.sun.com/developer/TechTips/1998/tt0521.html.
http://en.wikipedia.org/wiki/Shift-JIS
http://czyborra.com/charsets/iso8859.html#ISO-8859-1


## Author:


**Shanthi Alagappan** is a Senior Associate at Cognizant technology Solutions, Chennai, India. She holds Bachelor degree in mathematics from University of Madras, Chennai. She is Certified Software Engineer from Quality Assurance Institute, Florida, and USA. Shanthi has six years experience in various facets of QA activities on e-CRM products, Supply chain management, multi media and health information systems. She is proficient in Japanese language, and she is Level-2 JLPT (Japanese Level Proficiency Test) certified from Government of Japan. She has been working for Japanese clients where she performed Internationalization QA activities, security and Non-Functional testing in Japanese environment.