GlobalLogic®

# Unraveling the Mystery of i18n Testing

Shivani Goyal
Specialist Test Engineer, Quality Assurance

Viniti Garg
Senior Consultant, Quality Assurance

When it comes to testing an application that will be launched worldwide, i18n testing plays a key role. However, multilingual applications require comprehensive testing that can be very difficult to achieve. This white paper provides recommendations on critical data values for i18n testing.

www.globallogic.com

Shivani Goyal, Specialist Test Engineer, Quality Assurance
Viniti Garg, Senior Consultant, Quality Assurance

# Table of Contents

Shivani Goyal, Specialist Test Engineer, Quality Assurance
Viniti Garg, Senior Consultant, Quality Assurance

## Introduction

Internationalization testing is the process of validating a global application so that it can easily be conformed to local languages and cultures. This process is also called i18n testing because there are 18 characters between I and N in internationalization. Going forward, internationalization testing will be referred to as i18n testing in this white paper.

## Why i18n Testing?

The universal market has significantly evolved over the past decade, with nearly every enterprise serving customers around the globe. To increase international market share, there is a growing consensus to migrate the monolingual applications from legacy character sets to Unicode and to extend support for multilingual data, formats, and user interfaces. As such, i18n testing has become an integral part of software testing.

The goal of internationalization is to make sure that an application's code can handle Unicode support without causing data loss or data integrity issues. It ensures that the application being tested works uniformly across multiple regions and is relevant to a global audience. However, i18n testing can be challenging because an application's content and UI can be multilingual in terms of text, currencies, date, time, and number formats. To address such complexities, testers must select test data that encompass all aspects of i18n testing.

This white paper will identify how testers can choose the right test data to effectively and thoroughly test software intended for global markets.

## Foundation of i18n Testing

### Locale

The locale defines the conventions for language, country, and any special cultural preferences of a user. A user must be associated with a specific locale. It is crucial for any application with internationalization support to identify multiple locales and then behave according to the selected locale. The following formats can be changed for a locale:

- Collation
- Number format
- Date-time format
- Currency
- Calendar

In Windows, the locale can be set via Control Panel > Clock, Language, Region, and Language Options. On POSIX platforms such as UNIX, Linux, and others, the locale can be set via an environment variable "LANG" in the following format: [language[_territory][.codeset][@ modifier]].

### Character Encoding

Character encoding is mapping characters and symbols of a language into binary format, and it applies to every text resource (e.g., flat files, URLs, documents, databases, etc.). The default encoding for an application is usually determined by the defined locale. To test the encoding support, you should select test data so that it explores the boundaries of the encoding and is relevant to that encoding. For example, to test the encoding scheme ISO 8859-1, you should use accented letters along with ASCII letters.

To ensure consistency in encoding representation, the Unicode Standard is used industry-wide. It is a fixed-width encoding scheme for all characters used in the world's major written languages and can be leveraged for all text elements, such as letters, Japanese Hiragana characters that represent syllables, and Chinese ideographs that represent full words or concepts.

The Unicode Standard defines three encoding forms: UTF-8, UTF-16, and UTF-32. All three encoding forms encode the same common character repertoire and can be efficiently transformed into one another without loss of data.

- **UTF-8** is used for HTML and similar protocols. UTF-8 is a way of transforming all Unicode characters into a variable-length encoding of bytes. Since the Unicode characters corresponding to the familiar ASCII set have the same byte values as ASCII, Unicode characters that are transformed into UTF-8 can be used with most existing software without requiring extensive software rewrites.

- **UTF-16** is used in many environments that need to balance efficient access to characters with the economical use of storage. It is reasonably compact, and all the heavily-used characters fit into a single 16-bit code unit while the other characters are accessible via pairs of 16-bit code units.

- **UTF-32** is useful when a fixed-width, single code unit access to characters is desired and memory space is not a concern. When using UTF-32, each Unicode character is encoded in a single 32-bit code unit.

All three encoding forms need up to 4 bytes (or 32-bits) of data for each character.

## Time Zone

A time zone is a region that has a uniform standard time for legal, commercial, and social purposes. A time offset is an amount of time subtracted from or added to UTC to get the current civil time.

Daylight saving time (DST), or summer time, is the practice of advancing clocks during summer months by one hour so that light extends into the evening hours. In any particular time zone, residents either observe standard time all year round (e.g., Russia, South Africa), or they observe standard time in winter and daylight time in summer.

## Localization

Enterprises may want to provide users with a fully localized UI for major components like browsers, operating systems, and databases. Localization (sometimes shortened to "l10n") is the process of adapting a product or service to align with a particular language, culture, and desired local "look-and-feel."

In addition to providing idiomatic language translation, localizing a service or product requires developers to consider details such as time zones, money, national holidays, local color sensitivities, product or service names, gender roles, and geographic examples. A successfully localized service or product is one that appears to have actually been developed within the local culture.

## Collation

Collation controls the way string values are sorted. The rules vary with locale because different natural languages sort words differently. For example, a traditional Spanish collation places words beginning with "ch" at the end of a list of words beginning with "c".

Various collations are available for each language. Certain languages like French, Spanish, and German sort the same set of characters in different orders. Since there are specific attributes applicable to different collations, it is important to test each attribute with the relevant datasets from different languages.

## Guidelines for Test Data Preparation

Building a correct and comprehensive dataset gives you a head start for i18n testing, which can be complicated due to the variations between different cultures and languages. To help you prepare the test data in a systematic and focused manner, we suggest following the below guidelines.

### Strength

This attribute defines the comparison strength in order to sort/group the strings. To test this attribute, you should choose a language with accented characters and/or upper and lower case characters. This way, you can verify that both accent and case differences are being considered. Strength accepts the following values:

- **Primary** is a case-insensitive and accent-insensitive locale setting. It ignores case differences ("a" versus "A") and accent differences ("à" versus "â").
- **Secondary** is a case-insensitive locale setting. It ignores case differences ("a" versus "A") but considers accent differences (treats "à" different from "â").
- **Tertiary** allows both case differences and accent differences to be considered.
- **Quaternary** is used for Hiragana characters (e.g., "あ" < "ア" for this setting, which is "あ" = "ア" otherwise).

In the following example, we used a dataset from French to illustrate the sorting/grouping in Primary/Secondary/Tertiary comparison strengths.

Sorting Order (Ascending)

| Dataset | Primary | Secondary | Tertiary |
|---|---|---|---|
| À | A | A | a |
| à | a | a | A |
| Â | À | À | à |
| â | Â | à | À |
| A | à | Â | â |
| a | â | â | Â |
| æ | Æ | Æ | æ |
| Æ | æ | æ | Æ |

Grouping

| Dataset | Primary | Secondary | Tertiary |
|---|---|---|---|
| À | A | A | a |
| à | a | a | A |
| Â | À | À | à |
| â | Â | à | À |
| A | à | Â | â |
| a | â | â | Â |
| Æ | Æ | Æ | æ |
| æ | æ | æ | Æ |

## Storage

Since most languages use an alphabet with a limited set of text symbols, punctuation marks, and special characters (e.g., English, Italian), one byte per character usually suffices. One byte is enough to distinguish every possible character in a language and is known as a single-byte character. For languages that are more complex and require two or more bytes to represent them (e.g., Chinese, Japanese, Korean), multibyte characters are utilized.

When testing, you must include datasets from both single byte and multi-byte character sets. You should also provide data input that overruns the allocated storage capacity, even if the number of characters matches the specified length. Below is an example of this process.

| Text to be Stored | Number of Characters | Bytes of Storage Required | String(4) | String(5) |
|---|---|---|---|---|
| “role” | 4 | 4 | Pass | Pass |
| “rôle” | 4 | 5 | Overrun | Pass |

## Case

The locale setting and collation may allow case differences to be ignored or flipped. Uppercase letters may be sorted before lowercase letters, or vice versa. For example, lowercase letters are usually sorted before uppercase letters in English. Latvian letters are the exact opposite. As such, you should select languages with upper/lowercase characters to test this criterion (e.g., English, French, Russian). Below is an example of this process using characters from English.

Case-insensitive locale setting: “a” = “A”
Case-sensitive locale setting: “a” < “A”

Similarly, setting the collation order to UpperFirst will result in uppercase characters being sorted before lowercase characters, and vice versa. This setting is useful for locales that have already supported ordering but require different order of cases. Below is an example of the sorting order process (ascending).

| Dataset | UpperFirst | LowerFirst |
|---|---|---|
| USA | USA | usa |
| Usa | Usa | uSa |
| uSa | uSa | Usa |
| usa | usa | USA |

## Accents

Accented letters may be treated as minor variants of the unaccented letter, or they may be treated as distinct letters. For example:

- “é” in French may be treated equivalent to “e” depending on the locale setting.
- “Å” in Danish is treated as a separate letter that sorts just after “Z”.

Some French dictionary ordering traditions sort accents in reverse order, from the end of the string. For example, the “backwards” setting allows for the following sorting:

- Normal accent ordering: cote < coté < côte < côté
- Backward accent ordering: cote < côte < coté < côté

In the above example, the word “côte” sorts before “coté” because the acute accent on the final “e” is more significant than the circumflex on the “o”.

Another example:

- Normal accent ordering: americas < americàs < americÂs < àmericas < âmericas
- Backward accent ordering: americas < àmericas < âmericas < americàs < americÂs

Shivani Goyal, Specialist Test Engineer, Quality Assurance
Viniti Garg, Senior Consultant, Quality Assurance

## Date and Time

Date and time formats are important considerations for testing global applications.

Date Formats
Date formatting is not constant throughout the world. Although each date basically displays the day, month, and year – and time displays the hour, minutes, and seconds – their presentation order and separators do vary. In fact, there may even be differences between regions within the same country. To help illustrate this concept, below are comparisons of two basic date formats between US-English, Mexican-Spanish, and Japanese languages.

This variance in formatting can cause bugs if not tested carefully. For example, depending on which country you are in, the date specified as 07/04/01 could mean:

- US-English: July 4th, 2001
- Mexican-Spanish: April 7th, 2001
- Japanese: April 1st, 2007

| Date | US-English | Mexican-Spanish | Japanese | Note the Difference |
|---|---|---|---|---|
| SHORT DATE: | | | | |
| (10/12/95) | month/day/year<br>10/12/95 | day/month/year<br>12/10/95 | year/month/day<br>95/10/12 | The order of the month, day, and year varies |
| LONG DATE: | | | | |
| (Tuesday, October 12, 1995) | Tuesday, October 12, 1995 | martes 12 de octubre de 1995 | 1995年10月12日 | (1) The names of the months and days of the week are different.<br><br>(2) In Spanish, the day comes before the month, everything is lowercase, and the article “de” has been added.<br><br>(3) In Japanese, the day of the week is not displayed, and translations for day, month, and year act as separators. |

Time Formats
As with date formats, time formats also vary from one culture to another and should be tested for the following considerations:

**12-hour or 24-hour clock**
- The 24-hour clock is mostly used in European and Asian regions, while the 12-hour A.M./P.M. model is used in the United States.
- A.M./P.M. can be written in the local language.
- In a few locales, A.M./P.M. comes before the time.

**Separator for hours, minutes, and seconds**
- The colon (:) is the most commonly used character to separate hours, minutes, and seconds.
- Some Asian languages use ideographic characters as a separator.
- A few locales require “h”, “m”, and “s” for time displays.

**Time zones**
- Not testing for time zone support can be a source of runtime errors.
- The time zone is most commonly displayed using GMT (Greenwich Mean Time) or UTC (Coordinated Universal Time) as the base, followed by the time zone, with a positive or negative offset in hours and minutes. For example, the time zone for India is UTC +5:30, and the time zone for Eastern Time (US and Canada) is UTC -5:00.
- Time zones can also be displayed using names for the local zones. However, not all countries use local names, and the abbreviations to represent time zones are not unique.
- Daylight saving time is not used in all countries, and where DST is implemented, it does not start and end on the same day every year.

## Monetary and Numeric Formats

Symbols for monetary and numeric formats are defined by locales. For example, the U.S. uses a dot as a separator while France uses a comma. Therefore $1,234 can have a different meaning in different countries.

Monetary Formats
Currency formatting should be tested considering the following attributes:

**Currency symbol and placement**
Currency can be represented by a pre-defined symbol like the US dollar sign ($) or a combination of letters (USD). This symbol can also be placed either before or after the amount.

**Decimal and thousands separators**
Both decimal and thousands separators can be a dot or comma, depending on the locale. And while most locales are consistent in how they use separators across currencies and numbers, there are some exceptions. For example, Switzerland uses a period as a decimal separator for the Swiss franc (Sfr. 507.15) and a comma for numbers (507,15).

**Negative amounts**
Different locales depict negative amounts of money using different symbols and symbol placements. Below is an example of how some countries represent negative amounts.

| Country | Format | Description |
|---|---|---|
| US | ($507.15) | Use of parentheses |
| France | -507,15 F | Negative sign, amount, currency symbol; use of “,” as thousands separator |
| UK | -£ 507.15 | Negative sign, currency symbol, amount; use of “.” as thousands separator |
| Denmark | kr-507,15 | Negative sign between currency symbol and amount |
| Netherlands | €507,15- | Currency symbol, amount, negative sign |

Number Formats
As with currency, it is important to consider the variances in numeric value formats between locales. Below are some examples of how numbers can be presented in different locales.

**Thousands separator**

| Country | Number Format | Description |
|---|---|---|
| US | 6,115 | Use of comma |
| Germany | 6.115 | Use of period |

**Decimal separator**

| Country | Number Format | Description |
|---|---|---|
| US | 6,115.5 | Use of period |
| Germany | 6.115,5 | Use of comma |

**Negative numbers**

| Number Format | Description |
|---|---|
| -125 | Negative sign at the beginning of number |
| 125- | Negative sign at the end of number |
| (125) | Use of parentheses |
| 125 | Number displayed in red color |

**Percentages**

| Number Format | Description |
|---|---|
| 95% | Number, percentage sign |
| 95 % | Number, space, percentage sign |
| %95 | Percentage sign, number |
| 95 pct | Number, percent abbreviation |

## Calendar Support

While most English-speaking countries use the Gregorian Calendar, applications that support i18n should also consider other global calendars (e.g., Japanese, Thai Buddha, Hijri, Hebrew lunar, Persian, Taiwan). To test calendar support for different locales, you should consider the following factors:

- Calendars may have different year values. For example, the Gregorian year 2000 is the 12th year in the Japanese Heisei era and the year 1421 in the Hijri calendar.
- January 1st may or may not be the first day of the year.
- The length of the year and months may vary depending on the calendar.
- There are different ways to consider leap years.
- Even within the same calendar, the first day of the week might not be the same. For example, some European countries that use the Gregorian calendar start the week on Monday while the US starts the week on Sunday.

## Special Cases to Consider for i18n Testing

Some special cases that need to be considered while testing include:

- The sorting of letters A-Z may differ in non-English languages. For example, in Lithuanian, “i” < “y” < “k”.
- Sorting orders may vary even in the same language. For example, in German dictionaries, “öf” < “of” while in German phonebooks “of” < “öf”.
- In German, upper(‘ß’) → ‘SS’
- Combinations of letters can be considered as a single letter. For example, in traditional Spanish, “ch” is a single letter and “c” < “ch” <”d”.
- A letter can be considered as if it were two letters. For example, in German phonebooks, “ä” is treated as “ae”.
- The Thai language reverses the order of certain letters.
- Unaccented letters that are distinct in one language can be the same in another. For example, the letters “v” and “w” are two different letters in English but are considered variant forms of the same letter in Swedish.

## Conclusion

While there is a significant demand in the market for end-to-end global application support, the internationalization process is challenging across multiple levels. The biggest hurdle is testing these applications since hundreds of conventions and languages are used worldwide. Selecting the best test data is critical to testing such applications. The information available for i18n testing is expansive, so focusing on the right points is key to accelerating testing and bridging the gaps.

## References

http://userguide.icu-project.org/
http://www.unicode.org/standard/principles.html
http://en.wikipedia.org/wiki/Main_Page
https://msdn.microsoft.com/en-us/goglobal/bb688121

## About the Authors

Shivani Goyal is a Specialist Test Engineer of Quality Assurance at GlobalLogic's Noida facility. With over 14 years in the IT industry, Shivani's focus areas include testing different aspects of database, big data, analytics, and i18n. She is currently exploring new areas of testing in the field of data warehousing.

Viniti Garg is a Senior Consultant of Quality Assurance at GlobalLogic's Noida facility. She has over 10 years of experience in software product testing and is interested in datawarehouse testing, including SDK, optimizer, i18n, security, and cloud testing.

**About GlobalLogic Inc.**
GlobalLogic is a full-lifecycle product development services leader that combines deep domain expertise and cross-industry experience to connect makers with markets worldwide.Using insight gained from working on innovative products and disruptive technologies, we collaborate with customers to show them how strategic research and development can become a tool for managing their future. We build partnerships with market-defining business and technology leaders who want to make amazing products, discover new revenue opportunities, and accelerate time to market.

For more information, visit www.globallogic.com