# Unicode and Windows XP

*Cathy Wissink*
*Program Manager, Windows Globalization*
*Microsoft Corporation*

## *1. Introduction*

Windows XP, released in October 2001, builds on the pervasive Unicode support in Windows NT 4.0 and Windows 2000. Leveraging the single worldwide source in Windows 2000, the Windows Globalization Team was able to efficiently and quickly develop new international features for Windows XP, in spite of a short delta between releases (less than 18 months). Furthermore, the single worldwide source in Windows XP allowed for full globalization in all versions of the product, independent of localization or MUI (Multilingual User Interface) support. This paper will present an overview of globalization support in Windows from the perspective of Unicode, specifically:

- A description of the evolution of Windows on the NT platform from the perspective of Unicode support (namely, that of a single world-wide binary);
- A discussion of the international features of the next release of Windows and how Unicode played a role in the development of those features;
- A brief sketch of future directions of Windows from a globalization perspective.

## *2. Unicode and the Evolution of Windows NT*[1]

Support of Unicode on the Windows NT platform began with the development of Windows NT 3.1 in the early 1990s, when the decision was made to include full Unicode support from the onset. Basing the Windows NT kernel on Unicode was a radical departure from the code page-based approach used in the 16-bit platforms Windows 3.0 and 3.1. The implementation of Unicode laid the foundation for a fully-globalized operating system, although the realization of that goal remained a few revisions away.

Despite the underlying Unicode support on Windows NT 3.1, code page support continued to be necessary for many of the higher-level applications and components included in the system, explaining the pervasive use of the "A" [ANSI] versions of the Win32 APIs rather than the "W" ["wide" or Unicode] versions. (The term "ANSI" as used to signify Windows code pages is a historical reference, but is nowadays a misnomer that continues to persist in the Windows community. The source of this

[1] It should be emphasized here that the scope of this historical discussion is limited primarily to the Windows NT platform (Windows NT 3.1, Windows NT 4.0, Windows 2000). The 16-bit versions of Windows (Windows 95, Windows 98, Windows ME) have a limited amount of Unicode support compared to the Windows NT platform.

comes from the fact that the Windows code page 1252 was originally based on an ANSI draft, which became ISO Standard 8859-1. However, in adding code points to the range reserved for control codes in the ISO standard, the Windows code page 1252 and subsequent Windows code pages originally based on the ISO 8859-x series deviated from ISO. To this day, it is not uncommon to have the development community, both within and outside of Microsoft, confuse the 8859-1 code page with Windows 1252, as well as see "ANSI" or "A" used to signify Windows code page support.)

Windows NT 4.0, released in July 1996, demonstrated further progress towards globalization with the migration of system components and applications to Unicode. However, the system fell short of full globalization since there were still three separate binaries required for different language versions:

- A binary for the European language releases;
- A binary for the East Asian language releases (Korean, Japanese and Chinese);
- A binary for the Middle Eastern and Southeast Asian language releases.

Three different binaries meant three different source trees, bug fixes, file changes and service pack releases. Changes to the code couldn't be made just once, but had to be made three different times, and since each source tree was owned and maintained by a different development team, it was difficult to ensure that changes made to one binary were also made to the others in the exact same manner. This complicated the development process considerably and hampered efforts to create a consistently globalized product.

These issues were addressed in Windows NT 5.0 (renamed Windows 2000) through the development of a single worldwide source code base from which each of the Windows 2000 binaries were compiled[2]. The move towards this single worldwide source was simplified by the fact that the existing Windows NT source files were based on Unicode; the representation of data was consistent across the code bases. (In comparison, this type of code merge was not possible on the Windows 16-bit platform, since the binaries were based on code pages and not Unicode. This was the case even on the last release of this platform, namely Windows ME.)

This single worldwide source provided many benefits to both the development team and the end user. The development team benefited from the streamlining of the development process. The files distributed with each version of the operating system were created by a single team in a single source tree, independent of localization. Development time was more efficiently spent on specific localization (string) bugs rather

[2] The distinction between a single worldwide source and single worldwide binary is an important one. The single Windows 2000 source was compiled into separate binaries for each localized version of the system. Derived from the same source, these localized versions share the exact same core functionality but were tailored for specific markets as deemed necessary. In practice, the only noticeable differences across versions were those of resources in the user interface (e.g., in strings and dialogs) and in non-Win32 binaries (e.g., DOS emulation).

than code fixes specific to the version, and the product could therefore be localized faster. The time spent localizing was cut from months (Windows NT 4.0) to weeks (Windows 2000) in many instances. For example, the Korean version of Windows NT 4.0 shipped four months after the US English version. However, the Korean version of Windows 2000 shipped four *weeks* after the US English version. (The localization effort continued to improve on Windows XP, in that *all localized versions* shipped within 100 days of shipping US English.) Of course, less time spent localizing each version translated into lower costs.

User benefits included the early release of localized versions of the system after the release of the English version; the ability to run multilingual applications on any language version of the system (a properly globalized Hebrew application can run on the English, Japanese or Multilingual versions of Windows 2000, for example); and the fact that the same international functionality is available on all versions of the product (with the exception being the UI languages, or multilingual resources, in the MultiLanguage version of the system).

This last benefit is particularly important. It means, for example, that a user has all 126 user locales (sets of cultural data specific to a region and language) available to them on any version of Windows 2000[3]; Uniscribe (the script rendering engine) is also available on all versions of the product. (This is considerably different behavior than what was available on the different binaries of Windows NT 4.0; only a subset of the complete locale set was available on each version.)

The full set of user locales was made available initially on Windows 2000. The functionality important to the worldwide community, such as proper formatting and display of text, culturally correct date and time formatting, collation, and calendaring is available on every version of the system, from Windows 2000 forward.

Windows XP, as the merge of the Windows NT platform with key elements of the Windows 9x experience, builds on the single world-wide source code of Windows 2000 to provide even richer international functionality; a more comprehensive implementation of Unicode, support for more locales and languages and an improved user experience. These new features are outlined below.

### *3. The International Features of Windows XP*

As noted in the previous section, comprehensive and consistent international functionality is available on all versions of Windows XP. This international functionality includes the following:

[3] See Appendix A for a complete list of user locales available in Windows XP; this is a superset of the user locales available on Windows 2000.

- National Language Support (NLS), including the locale model
- Uniscribe (the Windows Unicode Script Processor, a script rendering engine), including the OpenType Layout Services
- Windows Text Services Framework
- MUI (Multilingual User Interface technology; available with the Windows XP Multilingual User Interface Pack)

### *3. 1. National Language Support*

NLS includes the functionality that determines behavior of the system in culturally appropriate operations:

- Collation and linguistic casing;
- Retrieval of data particular to a culture (date and time formatting, currency and number information, number grouping, native digits, calendars, etc.);
- String conversions to and from Unicode (based on code pages, or system locales, particular to the culture).

The NLS data for a particular culture (generally a combination of language and region[4]) is collectively referred to as a *locale*. For many people in the field of internationalization, this is the best-known definition of a locale, since this was the original definition. However, the concept of a "locale" has been expanded in the last few revisions of Windows into a dizzying array of locales, most notably:

- User Locale
- System Locale
- Input Locale

A radical overhaul of the Regional and Language Options Control Panel UI in Windows XP has made the locale model less confusing to the user, but for clarification's sake, I list below the old "locale" Windows 2000 nomenclature, and the new name in the Windows XP UI:

- *User Locale (now referred to as "Standards and Formats" on Windows XP)*. This is the concept that most people understand as a "locale". The user locale represents the preferences of a user based on cultural expectations for calendar type, time formatting, collation, currency formatting, etc. This can be modified by the user within Regional and Language Options in the Control Panel. It is a user

[4] Although the language + region combination is sufficient to define most locales, there are a few locales which, for either legacy or other reasons, require at least one other parameter. In certain parts of the world (most notably in the post-USSR countries), certain countries have implemented an alternate writing system. As such, it is necessary to have multiple locales for the same language + country combination which include a writing system parameter (examples of this include Azeri, Mongolian and Uzbek). There are also legacy locales that include a sort method parameter (i.e., Spanish – Spain in both International Sort and Traditional Sort).

property, and can be changed at any time, with the system notifying processes immediately. Changes are effective at once. A user can also customize settings by clicking the "Customize…" button. This brings up a dialog [figure 2] where numbers, dates, currency and time can be tailored to the user's preference.

*Figure 1. Regional and Language Options Control Panel: Standards and Formats (User Locale)*

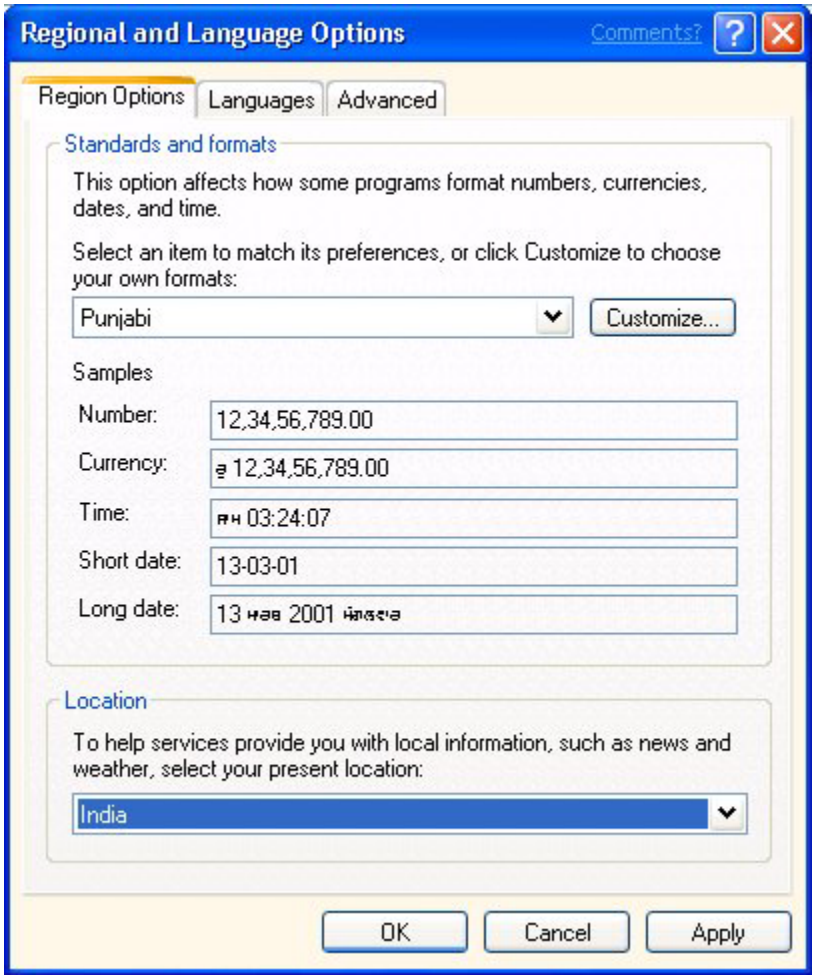
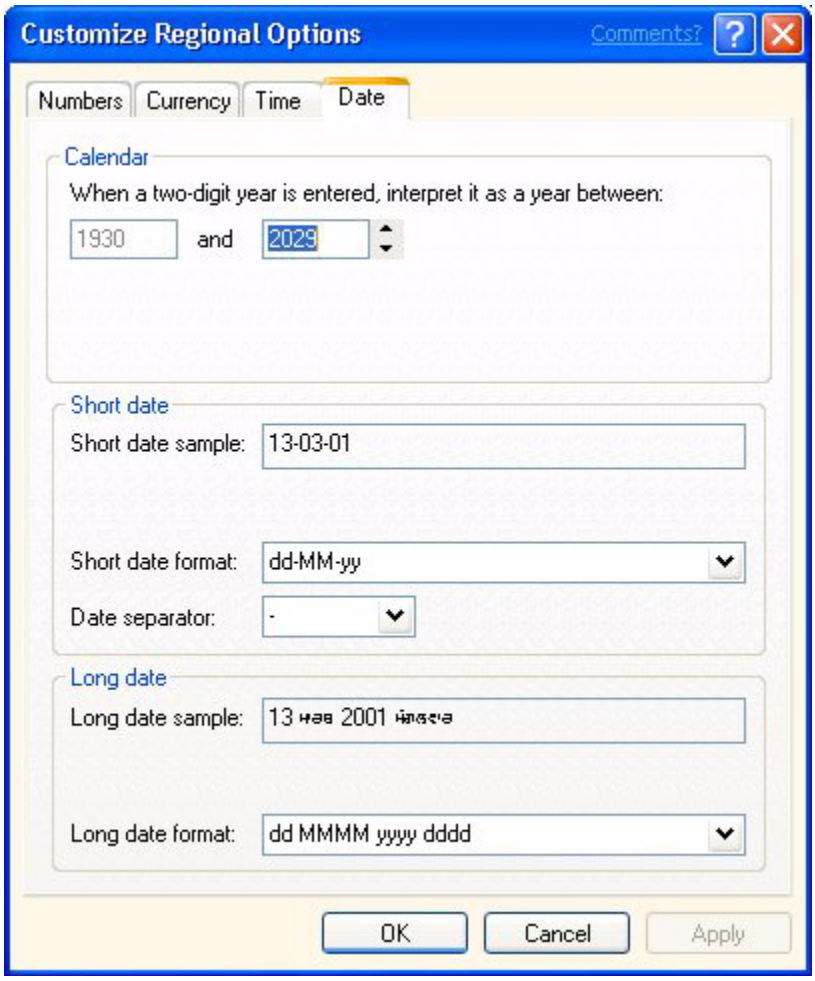


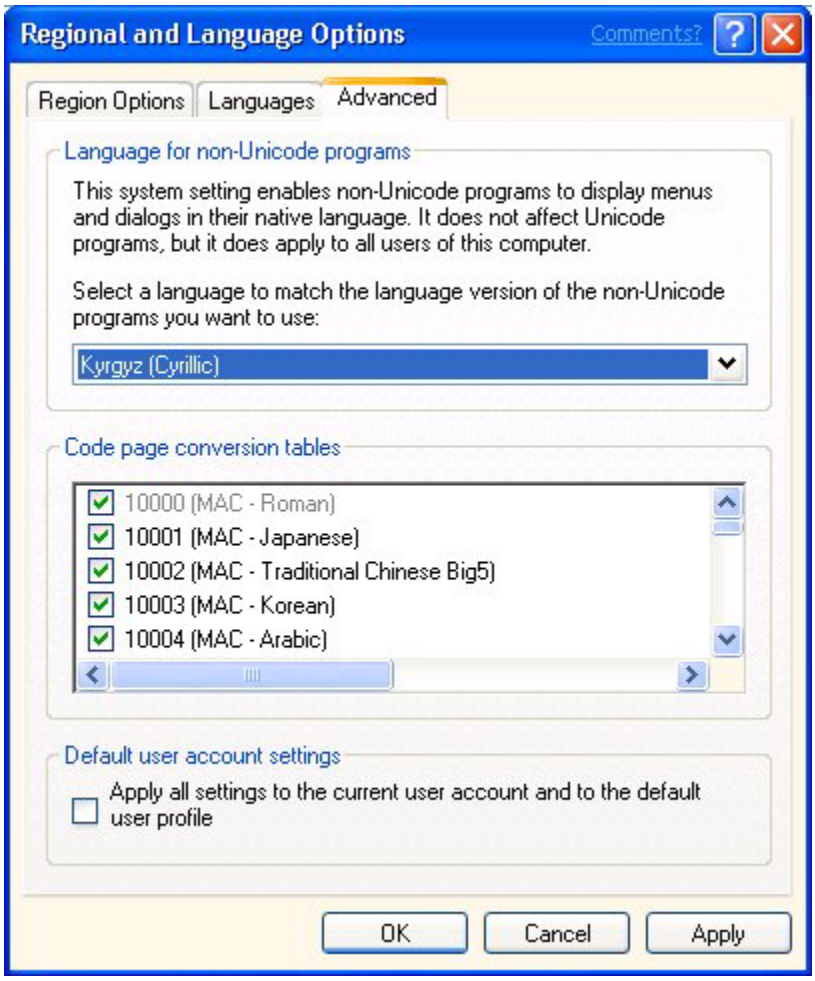
*Figure 2. Regional and Language Options Control Panel: Standards and Formats (User Locale)--Customize Regional Options*




- *System Locale (now referred to as "the Language for non-Unicode Programs" on Windows XP).* The term "system locale" as used in Windows 2000 was really a misnomer; the system locale is actually the set of underlying code pages used for a particular user locale. The system locale allows non-Unicode applications to run as they would on a system[5] set to their particular code page (and has no impact on Unicode applications whatsoever). However, it is a setting that requires administrator privileges on the system (since it is a system property; thus the name). Note that certain user locales (e.g., the Indic locales, Georgian, Armenian) do not have a corresponding system locale; they are Unicode-only locales and as such do not have any code page support.

[5] An example of this would be the 16-bit version of Windows appropriate to the application's market.

*Figure 3. Regional and Language Options Control Panel: Language for non-Unicode Programs (System Locale)*




- *Input Locale (now referred to as "Input Languages and Methods" on Windows XP).* The input locale includes two values: the locale ID particular to the language of input and the ID particular to the keyboard or IME. A user locale has a primary input locale associated with it, but the user can change the input locale at any time by clicking the "Details" button (which brings up the Text Services dialog [figure 5]). Input locales can be added or removed through the Regional and Language Options Control Panel, and switched through many different means.

*Figure 4. Regional and Language Options Control Panel: Input Languages and Methods (Input Locale)*

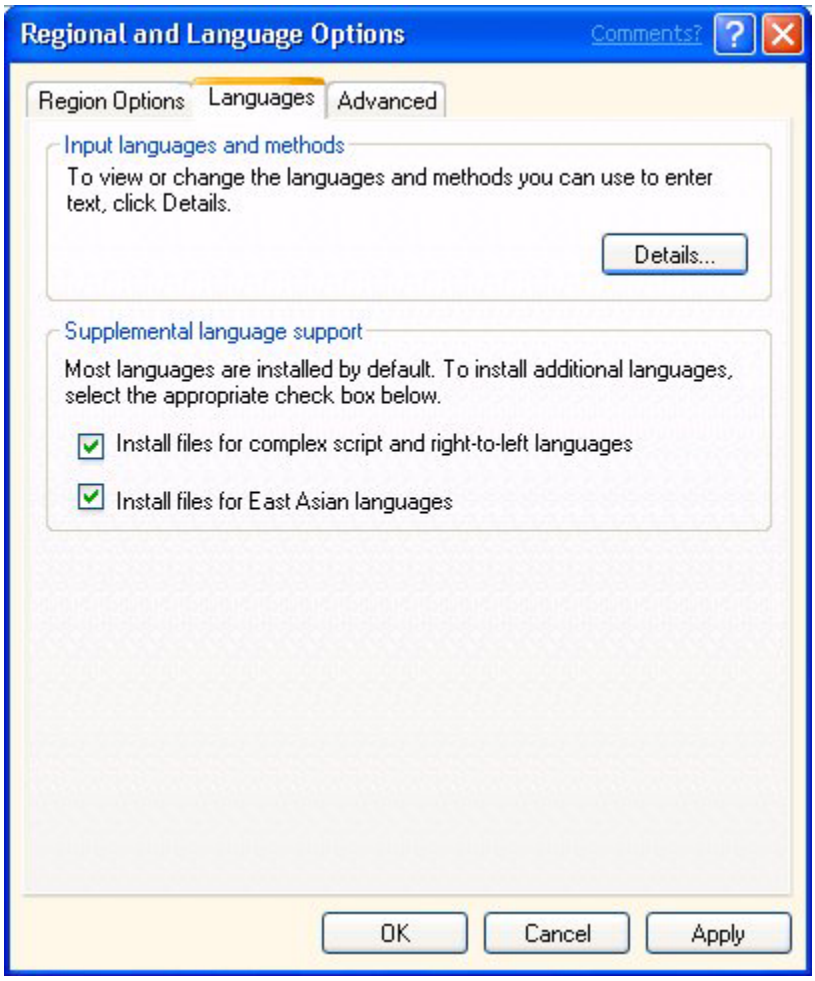
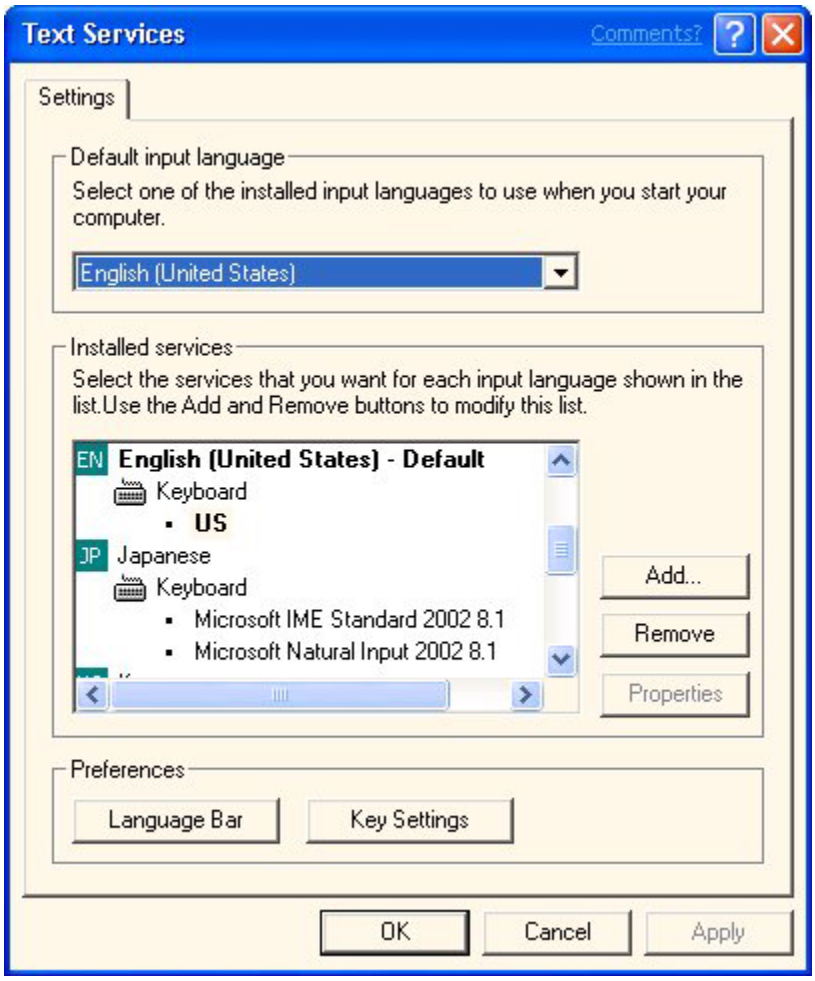


*Figure 5. Regional and Language Options Control Panel: Input Languages and Methods: Details Button – Text Services*




The system locale and input locale are set by default to match the user locale (in other words, the user locale determines which code pages and input methods are supported by default, although these can be changed if the user has appropriate permissions). In addition to the 126 user locales supported in Windows 2000, nine new user locales were added for Windows XP, namely: Punjabi, Gujarati, Telugu, Kannada, Kyrgyz, Mongolian (Cyrillic script), Galician, Divehi and Syriac[6]. In addition, an invariant locale was added. This locale is useful in those instances where locale-independent results are desired, but it should not be used for display. (See Appendix A for a complete list of user locales on Windows XP.)

[6] For each of the user locales added to Windows XP, input locale and system locale support was also added. The exception to this rule is those situations where the user locale is purely Unicode; in these cases, there is no system locale support (that is, underlying code page support). In addition, further improvements were made to existing locale data for Windows XP, including adding support for old Hangul collation in Korean, and improvements to the locale database and existing input locales.

Finally, with regards to updated NLS support, supplementary character or surrogate pair (aka "surrogate"[7]) support was turned on by default in Windows XP. (In comparison, it was necessary to use particular registry key values on Windows 2000 to get the same type of support.[8])

### *3.2. Uniscribe (Windows Unicode Script Processor)*

Uniscribe is the technology used to handle the layout, rendering and editing of complex scripts. A complex script is one that requires special processing in order to be displayed and edited; the characters are not simply displayed left to right in a relatively context-free manner (as is often the case with scripts such as Latin and Cyrillic). For example, the Hebrew and Arabic scripts have a right-to-left reading order rather than left-to-right. Scripts such as Devanagari and Arabic have characters that need to be shaped depending on the context in which they present themselves[9]. Vietnamese script has the characteristic of allowing multiple diacritics on a base Latin letter, and the placement of these diacritics is dependent on the particular diacritics used, the number of diacritics used, and the underlying base letter. With Indic scripts (e.g., Tamil, Telugu, Gujarati), it is often the case that the order in which characters are input is not the same order that characters are displayed. Thai has special word breaking needs.

Uniscribe was developed to handle rendering issues in these scripts that had special needs like character reordering, contextual glyph shaping, and specialized word breaking. This technology incorporates a collection of shaping engines, one for each complex script. Each shaping engine handles the rules particular to a certain script, working in conjunction with the graphics display subsystem for script display rules and the OpenType Layout Services for layout details. Uniscribe provides a single technology that can be used across versions of Windows XP; it is agnostic with regards to script handling.

One of the system components integrated into the Windows 2000 single worldwide binary, Uniscribe has been improved for Windows XP in the following areas:

- New script support, including Gujarati, Gurmukhi, Kannada, Telugu, Divehi, and Syriac;
- Improvements to resolution, layout and formatting made possible by the implementation of GDI+;

---

[7] Note that the term "surrogate" is no longer considered correct terminology for a UTF-16 extension character consisting of a high and low surrogate. See http://www.unicode.org/glossary/index.html for specific definitions.
[8] For the specifics on these registry values, please see the information in MSDN at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/intl/unicode_192r.asp.
[9] Analogous to this concept is cursive writing in Latin script. While a particular letter (e.g., b) might look a certain way if it is written by itself, it will look slightly different if written after a letter (and again, the shape may be slightly different depending on the letter it follows).

- Uniscribe is now always installed on the system independent of language preference;
- Font fallback support.

### *3.3. Windows Text Services Framework*

The Windows Text Services Framework, a simple and scalable system framework enabling natural language text services and advanced text input on the desktop and in applications, makes its debut in Windows XP. With the Text Services Framework, an application can implement a generic, device-neutral, language-neutral API and thus be enabled for a number of text services including speech, handwriting, and keyboard input methods. For more details about this framework, please see Kevin Gjerstad's paper on this topic in the proceedings from the September 2001 IUC.

### *3.4. MUI (Windows XP Multilingual User Interface Pack)*

MUI is the shorthand term often used for what is officially called the Windows XP Multilingual User Interface Pack; it is also used as the term to describe the underlying technology on this particular version of Windows XP. MUI allows the UI language of the system's menus, dialogs and Help files to be changed into one of 33 different languages[10] according to the preferences of the user. The MUI version of Windows XP was developed with the overall goal of reducing the cost of implementing and maintaining multilingual IT environments, keeping the following scenarios in mind:

- *Allowing companies to implement a single worldwide product* that can be tailored to a user's UI language needs; UI language is just another setting, rather than a completely different version of the product. For example, a corporation based in Singapore with subsidiary offices in New York and Paris could deploy the Windows XP MUI version in all three locations, rather than deploying the Chinese version in Singapore, the English version in New York and the French version in Paris. This reduces both time and cost in the rollout, maintenance and updates of the system.
- *Enabling administrators to better support their users* independently of the UI language a user is running. For example, support personnel can change the UI of a problematic machine to better diagnose issues in their own language, then reset the UI language back to that of the user; system personnel can use their own language to configure a machine, then change the language to one more appropriate to the user.
- *Allowing users to change the language of the UI* as it suits them, improving their productivity (a user is not stuck having to work with a non-preferred language). For example, each user can set her own UI language. MUI eliminates the need to

[10] There are 24 fully localized languages in MUI. In addition, there are 9 languages being released on MUI with Language Interface Packs (LIPs), which are partial localizations.

have dual-boot configurations or machines dedicated to a single localized language version.

The development of MUI proved much easier than expected, thanks to underlying Unicode support. Since Windows resources were already in Unicode prior to the development of MUI, there were no problems with underlying code page (system locale) dependencies in resources. Display of any localized text ceased to be an issue because of the single worldwide source code; all the international functionality was already in place to properly format and display text, independent of language.

The "magic" of MUI resides in the build process and resource loader. The system is built by copying the resource files from the localized builds (i.e., an automated build process strips the resources from the localized build and propagates them to a separate CD). Testers from the respective localization teams check for the same type of issues that might crop up on a localized build. The resource loader has been modified to check the user's UI language setting at runtime and automatically loads the resources appropriate for the selected language.

Much of the MUI development work between Windows 2000 and Windows XP was spent increasing the percentage of the system's UI that displays in the selected language; only 95% of the Windows 2000 MultiLanguage[11] version's UI was localized into each language (compared to 100% of the UI in each of the localized versions)[12]. This work involved removing UI strings from the registry and out of the kernel into resource files, as well as improvements to the Shell, Desktop and Console resource handling. In addition, all the Windows XP help files were enabled for UI language switching[13]. (Further improvements were made to MUI for the .NET Server release after Windows XP shipped, mostly in improving the setup experience.)

One important feature to understand about MUI (as opposed to the other international functionality described previously in this section) is that the MUI technology only exists on the Windows XP Multilingual User Interface Pack. However, all other international functionality (NLS, Uniscribe, Windows Text Services Framework) is available on *all* versions of Windows XP (standard English, all localized versions and the MUI version). In other words:

- MUI allows the user to change the *user interface language* (menus and dialogs). This functionality is only available on the MUI version of Windows 2000 and Windows XP.
- The ability to work with *language content* (text editing, formatting, printing; culturally-correct formatting of dates, times, currency, collation and linguistic

[11] Note the name change between Windows 2000 and Windows XP with regard to MUI.
[12] The effort expended to fully "MUI-ze" the UI increases as the amount of non-localized text decreases; it is a classic case of diminishing returns.
[13] On Windows 2000, help files were partially moved to the UI language for MUI, but many (win32hlp, GDI-based help) were not.

casing) is available on *all* versions of Windows 2000 and Windows XP, including the MUI version of Windows XP (and of course the localized versions).

Again, it is important to emphasize that all the international features described in this section would not be available to the extent they now are, were it not for the built-in support for Unicode originally added to the predecessors of Windows XP, culminating in the single worldwide source used for Windows 2000. By providing this type of Unicode support on the Windows NT platform, developers had an environment in which features for the global market like MUI could be incubated and successfully implemented, with remarkably little development time[14]. In addition, new ideas in international functionality can be developed with the assumption that Unicode support is available and that all future versions of the Windows platform will continue to improve that support.

## *4.0 Future Directions in Globalization on Windows*

The international functionality on Windows 2000 has been described as revolutionary; the changes to Windows XP international functionality were more along an evolutionary scale (which were still considerable considering the short development cycle). However, the question remains: what's next for globalization on Windows?

Our mantra in planning future development is: "English is just another language".

Further improvements to the MUI development process will marry localization and the MUI technology in future versions of Windows. New directions are also planned for NLS. Previously unsupported language and locale support is being researched; new international functionality we are working on will resolve some of the deficiencies inherent in the current locale model.

The overarching goal of our research, both in new plans and current implementation, is to make cultural and linguistic differences between versions transparent to the user. While we as implementers strive to work towards a single worldwide version of product that serves all users' needs independent of location or language, it is still imperative that the user feels the product serves needs unique to her language, culture or country. In other words, while Windows strives for a global reach, we aspire to make the user's experience local in scope. As we have found in our implementation, Unicode is the major reason both these goals are attainable, both today and in future releases of Windows.

---

[14] The development team was truly surprised by how little had to be changed to make MUI work: as noted earlier, Windows resources are stored as Unicode, and as such there were no dependencies on the resource's system locale to worry about. Secondly, with Uniscribe in place on all versions of the product, getting correct text formatting and display for all of the UI languages was not an issue.

*Appendix A: List of Supported User Locales in Windows XP*

| *LCID* | *Name of Locale* |
|---|---|
| 0436 | Afrikaans - South Africa |
| 041c | Albanian - Albania |
| 1401 | Arabic - Algeria |
| 3c01 | Arabic - Bahrain |
| 0c01 | Arabic - Egypt |
| 0801 | Arabic - Iraq |
| 2c01 | Arabic - Jordan |
| 3401 | Arabic - Kuwait |
| 3001 | Arabic - Lebanon |
| 1001 | Arabic - Libya |
| 1801 | Arabic - Morocco |
| 2001 | Arabic - Oman |
| 4001 | Arabic - Qatar |
| 0401 | Arabic - Saudi Arabia |
| 2801 | Arabic - Syria |
| 1c01 | Arabic - Tunisia |
| 3801 | Arabic - U.A.E. |
| 2401 | Arabic - Yemen |
| 042b | Armenian - Armenia |
| 082c | Azeri - Azerbaijan (Cyrillic) |
| 042c | Azeri - Azerbaijan (Latin) |
| 042d | Basque - Spain |
| 0423 | Belarusian - Belarus |
| 0402 | Bulgarian - Bulgaria |
| 0403 | Catalan - Spain |
| 0c04 | Chinese - Hong Kong SAR |
| 1404 | Chinese – Macau SAR |
| 0804 | Chinese - PRC |
| 1004 | Chinese - Singapore |
| 0404 | Chinese - Taiwan |
| 041a | Croatian - Croatia |
| 0405 | Czech - Czech Republic |
| 0406 | Danish - Denmark |
| 0465 | Divehi - Maldives |
| 0813 | Dutch - Belgium |
| 0413 | Dutch - Netherlands |
| 0c09 | English - Australia |
| 2809 | English - Belize |
| 1009 | English - Canada |
| 2409 | English - Caribbean |
| 1809 | English - Ireland |
| 2009 | English - Jamaica |
| 1409 | English - New Zealand |
| 3409 | English - Philippines |
| 1c09 | English - South Africa |
| 2c09 | English - Trinidad |
| 0809 | English - United Kingdom |
| 0409 | English - United States |
| 3009 | English - Zimbabwe |
| 0425 | Estonian - Estonia |
| 0438 | Faeroese - Faeroe Islands |
| 0429 | Farsi - Iran |
| 040b | Finnish - Finland |
| 080c | French - Belgium |
| 0c0c | French - Canada |
| 040c | French - France |
| 140c | French - Luxembourg |
| 180c | French - Monaco |
| 100c | French - Switzerland |
| 0456 | Galician - Spain |
| 0437 | Georgian - Georgia |
| 0c07 | German - Austria |
| 0407 | German - Germany |
| 1407 | German - Liechtenstein |
| 1007 | German - Luxembourg |
| 0807 | German - Switzerland |
| 0408 | Greek - Greece |
| 0447 | Gujarati - India |
| 040d | Hebrew - Israel |
| 0439 | Hindi - India |
| 040e | Hungarian - Hungary |
| 040f | Icelandic - Iceland |
| 0421 | Indonesian - Indonesia |
| 0410 | Italian - Italy |
| 0810 | Italian - Switzerland |
| 0411 | Japanese - Japan |
| 044b | Kannada - India |
| 043f | Kazakh - Kazakhstan |
| 0457 | Konkani - India |
| 0412 | Korean (Extended Wansung) - Korea |
| 0440 | Kyrgyz - Kyrgyzstan |
| 0426 | Latvian - Latvia |
| 0427 | Lithuanian - Lithuania |
| 042f | Macedonian – Macedonia (FYROM) |
| 083e | Malay - Brunei Darussalam |
| 043e | Malay - Malaysia |
| 044e | Marathi - India |
| 0450 | Mongolian (Cyrillic) - Mongolia |
| 0414 | Norwegian - Norway (Bokmål) |
| 0814 | Norwegian - Norway (Nynorsk) |
| 0415 | Polish - Poland |
| 0416 | Portuguese - Brazil |
| 0816 | Portuguese - Portugal |
| 0446 | Punjabi - India |
| 0418 | Romanian - Romania |
| 0419 | Russian - Russia |
| 044f | Sanskrit - India |
| 0c1a | Serbian - Serbia (Cyrillic) |
| 081a | Serbian - Serbia (Latin) |
| 041b | Slovak - Slovakia |
| 0424 | Slovenian - Slovenia |
| 2c0a | Spanish - Argentina |
| 400a | Spanish - Bolivia |
| 340a | Spanish - Chile |
| 240a | Spanish - Colombia |
| 140a | Spanish - Costa Rica |
| 1c0a | Spanish - Dominican Republic |
| 300a | Spanish - Ecuador |

| | |
|---|---|
| 440a | Spanish - El Salvador |
| 100a | Spanish - Guatemala |
| 480a | Spanish - Honduras |
| 080a | Spanish - Mexico |
| 4c0a | Spanish - Nicaragua |
| 180a | Spanish - Panama |
| 3c0a | Spanish - Paraguay |
| 280a | Spanish - Peru |
| 500a | Spanish - Puerto Rico |
| 0c0a | Spanish - Spain (International Sort) |
| 040a | Spanish - Spain (Traditional Sort) |
| 380a | Spanish - Uruguay |
| 200a | Spanish - Venezuela |
| 0441 | Swahili - Kenya |
| 081d | Swedish - Finland |
| 041d | Swedish - Sweden |
| 045a | Syriac - Syria |
| 0449 | Tamil - India |
| 0444 | Tatar - Tatarstan |
| 044a | Telugu - India |
| 041e | Thai - Thailand |
| 041f | Turkish - Turkey |
| 0422 | Ukrainian - Ukraine |
| 0420 | Urdu - Pakistan |
| 0843 | Uzbek - Uzbekistan (Cyrillic) |
| 0443 | Uzbek - Uzbekistan (Latin) |
| 042a | Vietnamese - Vietnam |