Globalizing Applications Part 1: Pre-Unicode Solutions

Michael Redford

Michael Redford studied German and Economics at Rutgers University (B.A.) and Theoretical Linguistics at the University of Constance (M.A.). He recently joined SAP AG as an Information Developer (Technology Development, Development Platforms — Internationalization). He can be reached at michael.redford@sap.com.

The Internet has changed the way people and businesses communicate, and has made it possible to access and transfer data anywhere, at any time.

At the moment, the majority of data passing through the Internet is in English, but as Internet access spreads and costs decrease, the online population will more closely mirror the overall world population. Estimates of population growth vary greatly, but demographic trends clearly show that the number of language speakers who use a non-Latin script will boom. More Chinese, Hindi, Arabic, Bengali, Russian, and Japanese users means more data in those languages will be traveling through the Internet. mySAP.com must therefore be polyglot to function in multilingual business settings and to function across heterogeneous IT landscapes.

The Internet and globalization place additional demands on developers, because they have to consider language issues when they write programs, and on system administrators, who have to maintain larger, global networks. But it also places demands on managers, who have to stay on top of increasingly far-flung business operations. This article, the first of a two-part series, is therefore addressed to all these SAP constituencies, because the need to understand the issues involved in internationalization is a byproduct of our increasing reliance on the Internet.

In this article, I give a general introduction to character code pages and present two internationalization solutions that SAP currently offers: blended code pages and Multi-Display/Multi-Processing (MDMP) code pages. The second article in this series, which will appear in the next edition of *SAP Professional Journal*, deals with Unicode, the

character encoding used in the World Wide Web, Java, XML, JavaScript, and everything else on the Internet. Code pages are only one aspect of internationalization, but they are the most critical part, because it is crucial that users can see, enter, and print all of the characters used in their language without those characters becoming garbled.

Terms

Internationalization:
The process of making a system or application independent of, or transparent to, natural language (e.g., a system that can support multiple logon languages).

Localization:
The process or result of modifying a system or application software to support a particular language environment (e.g., a system designed to support only one logon language). Localization differs from internationalization, which attempts to remove all references to language from a system or application.

Character Code Pages

A character is an abstract representation of a written object, such as:

- The letter A
- The symbol ♖
- The ideograph 龜
- The dingbat ☛

A character's appearance on a screen or on the printed page is referred to as a *glyph*, not a character. For example:

A *A* A *A*

They all look slightly different, but they all are renderings of the character A.

For the remainder of this article, I will place a forward slash before and after an element to indicate that it should be regarded as a character. For example, /A/ refers to the character A. Braces will be used to indicate glyphs. For example, [A] should be interpreted as the glyph A — i.e., the way in which the /A/ (the character A) would appear on a screen or on a printed page. Lastly, the symbols < > indicate a keyboard letter, so <A> should be interpreted as the keyboard letter A.

Each character has a unique *character code*, which distinguishes it from other characters, for example, /A/ = 0x41, /B/ = 0x42, etc. A *font* defines how a character is rendered into a glyph. The character code 0x41 can correspond to radically different glyphs: [A], [*A*], [✌], [✡], [✍], depending on the font. Despite these differences, the character code remains the same — change the font back to Times Roman and the [A] reappears.

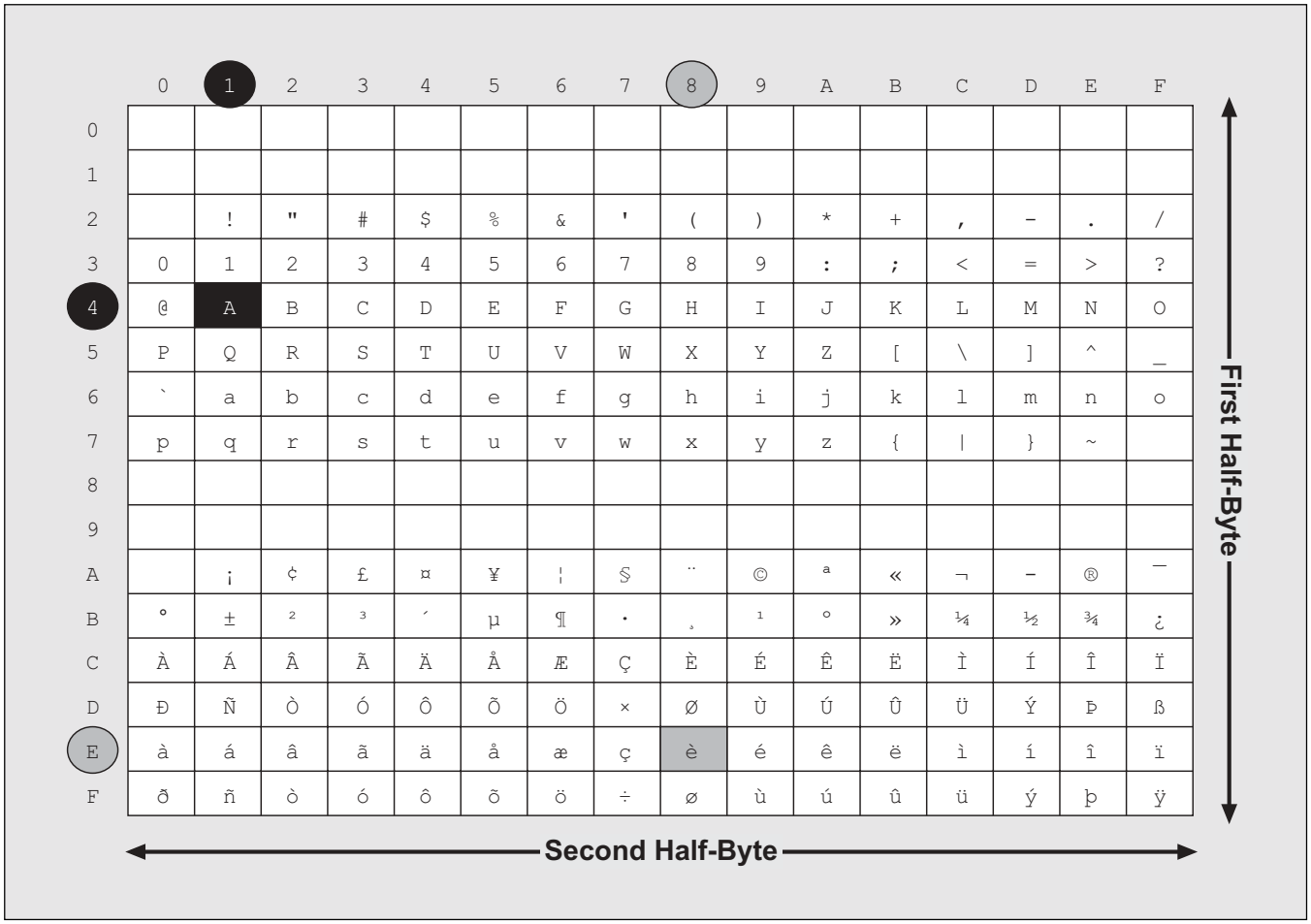
Character code values derive from the rows and columns of a *character code page*, such as the one shown in **Figure 1**. The most common character code pages have space for 256 characters (16 rows x 16 columns). Each space on the code page, or *code point*, is reserved for one character. A character code is simply the coordinate value for a given space in the matrix.

In the code page shown in Figure 1, the character /A/ has the code point 0x41 in hexadecimal notation, and the character /è/ has the code point 0xE8. The empty fields are reserved for nonprinting characters, such as `LINE FEED`, or they have not been assigned a character.

In theory, developers are free to design their own code page and assign values as they choose. For obvious reasons there are numerous standard code pages, such as the code page in Figure 1, which is one of the ISO8859 code pages. In all ISO8859 code pages, the upper portion of the code page is constant and contains 7-bit ASCII[1] characters. The lower

[1] American Standard Code for Information Interchange.

Figure 1 Code Page ISO8859-1 (Latin-1): A 1-Byte Code Page



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8																
9																
A		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

portion of the code page varies: the character at code point 0xE8 can be /è/, /č/, /θ/, or /ш/, *depending on the code page*. If the incorrect code page is selected, the wrong character will be used.

Most computer users are unaware that the keystrokes they make are converted into several different character code pages: When we hit a key on the keyboard of a Windows frontend, it is encoded in MS1252, a Microsoft code page, then converted to ISO8859-1 on the SAP application server, and then converted to Unicode (UTF-8) when it is sent via the Internet; at the other end it may be converted to EBCDIC[2] on an AS/400 and then the entire conversion odyssey can begin again. Therefore, not only must the relevant code pages be integrated within the three tiers of an R/3 landscape (database, application server, and frontend), but internal code pages must also be compatible with external code pages as well.

Thus internationalization involves juggling multiple code pages, because prior to Unicode, no single code page could provide the necessary cross-platform data format for all languages. For example, while the ISO8859-1 code page in Figure 1 is large enough to accommodate many of the world's languages, there is not enough room to represent all of the characters used in French, Russian, and Greek in a 1-byte code page. And Chinese, Japanese, and Korean each require their own 2-byte code page. (Even in multibyte code pages, the 7-bit ASCII

[2] IBM's Extended Binary Coded Decimal Interchange Code.

Figure 2 Standard Code Pages and mySAP.com-Supported Languages

Code Page	Supported Languages
ISO8859-1	Danish, Dutch, English, Finnish, French, German, Italian, Icelandic, Norwegian, Portuguese, Spanish, Swedish
ISO8859-2	Croatian, Czech, English, German, Hungarian, Polish, Romanian, Slovakian, Slovene
ISO8859-3	Dutch, English, German, Spanish, Turkish
ISO8859-5	English, Russian
ISO8859-6	Arabic, English
ISO8859-7	English, Greek
ISO8859-8	English, Hebrew
ISO8859-9	Danish, Dutch, English, Finnish, French, German, Italian, Norwegian, Portuguese, Spanish, Swedish, Turkish
Shift-JIS (SJIS)	English, Japanese
GB2312-80	English, Chinese
BIG5	English, Taiwanese
KSC5601	English, Korean
ISO8859-11 (TIS620)	English, Thai

Highlighted rows indicate multibyte code pages.

characters are included, and these characters are always 1 byte long.)

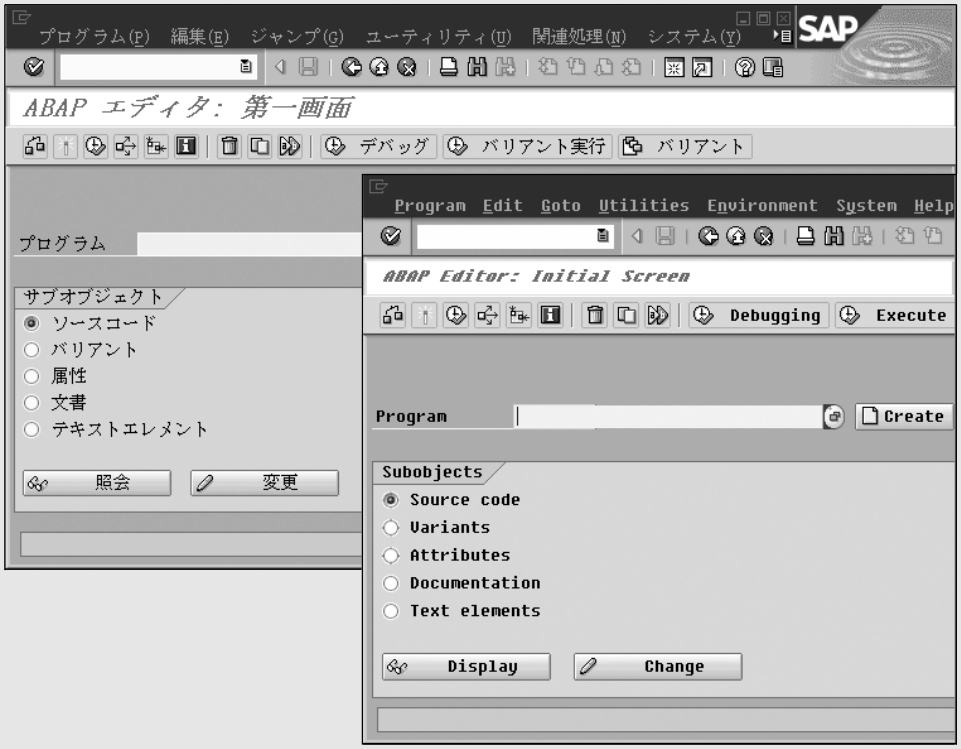
Figure 2 lists the standard code pages and languages supported by mySAP.com (EBCDIC code pages are not shown). A single code page contains characters for a specific set of languages. For example, the code page ISO8859-1 provides the characters used in Western European languages, including Danish /Å/, Finnish /ä/, French /è/, German /ü/, Icelandic /ð/, and Spanish /¿/. The code page Shift-JIS (SJIS) contains Japanese and English characters, enabling both Japanese and English users to use their respective languages. **Figure 3** shows the screen for transaction SE38 (ABAP Editor). The menus are in Japanese, and the shortcut keys (e.g., H for Help) are correctly represented because all English characters are included in the SJIS code page.

The login language also determines the locale, which specifies a user's country and language. This information is required for punctuation and for formatting items such as numbers, time and date, and currencies. The locale also determines sorting procedures. Normally, a computer sorts according to internal (binary) criteria, which often differs from the sort order a user would expect. In addition, the sort order varies from language to language. In German, /ä/ is sorted as the sequence /ae/, while in Swedish, /ä/ is sorted after /z/.[3] The sort order is important for searches, for example. A German user who enters <a-b> in a search field expects to find, and will find, all entries beginning with /ä/; if the same user had logged in as a Swedish user, none of the entries beginning with /ä/ would be found.

If you want to sort a table according to language and country-specific lexicographical conventions, the command SET LOCALE LANGUAGE is all that is required: ABAP programs are written to be

[3] In fact, the situation is even more complex. In a German telephone book /a/ = /ae/, but in a dictionary /ä/ comes directly after /a/.

Figure 3 Logon Language JA and EN: Transaction SE38



language-neutral and the locale provides all the necessary language information. The following code example shows the results of a sort with an English (`'EN'`) locale, and with a Spanish (`'SP'`) locale:

```
SET LOCALE LANGUAGE.
SORT Z00 AS TEXT.
```

Z00	'EN'	'SP'
llama	leopard	leopard
monkey	lion	lion
lion	**llama**	lynx
leopard	lynx	**llama**
lynx	monkey	monkey

The traditional Spanish sorting order treats /ll/ as a single character between /l/ and /m/, and "llama" therefore comes *after* "lynx." By setting the locale, lists are sorted as a user expects them to be. Locales are platform-dependent and must be created for each platform. Unfortunately, the sort results are also operating-system dependent, particularly when sorting characters that do not belong to the active code page.

The locale also determines the relationship between lowercase and uppercase letters. Note that converting from lowercase to uppercase actually changes a character's code point: /a/ is 0x41 and /A/ is 0x61. In English the conversion from lowercase to uppercase is straightforward, but for languages with diacritics, there is some variation. In German the capital of lowercase /ä/ is /Ä/, but in French it is /A/. In English the capital letter of /i/ does not have a dot, but in Turkish it does /İ/. The proper conversion is essential because when conducting searches or comparing key fields, the difference can be crucial.

In a table with mixed-language data, such as our next example, the conversion to uppercase must occur line by line to ensure that the correct uppercase

character is selected, or, in the case of Japanese, that no conversion occurs:

```
TRANSLATE words TO UPPER CASE.
```

words

EN	inside	→	INSIDE
TR	için	→	İÇİN
FR	häir	→	HAIR
DE	äpfel	→	ÄPFEL
JA	東	→	東

For SAP to provide language support, it is not enough to simply provide a set of locales and code pages, however. Standard code pages are inadequate when the languages that need to be supported do not belong to the same standard code pages — for example, Japanese and German, or French and Russian. The original R/3 design only permitted one code page per system, and the same code page was required for the database, all application servers, and all frontends. Thus multiple languages could only be supported if they belonged to the same code page; the size of a code page therefore placed limits on the number and combinations of languages that could be used in R/3. To improve language support, either (1) new code pages had to be introduced, or (2) more code pages had to be simultaneously available in R/3. SAP has done both, introducing new, blended code pages in Release 3.0D and support for multiple code pages in Release 3.1.

SAP Blended Code Pages

In Release 3.0D, SAP introduced multibyte blended code pages, which contain characters out of several standard code pages. Blended code pages are not standard code pages. These are special, SAP-customized code pages that were devised to support an increased number of possible language combinations within a single code page. (Remember, an R/3 system can only support one code page.) The blended code page Eurojapan, for example, contains the majority of Japanese characters (SJIS) and all characters needed for Western European languages (ISO8859-1).

Figure 4 shows a blended code page system. The system code page is Eurojapan, and the frontend code

Figure 4 Blended Code Page

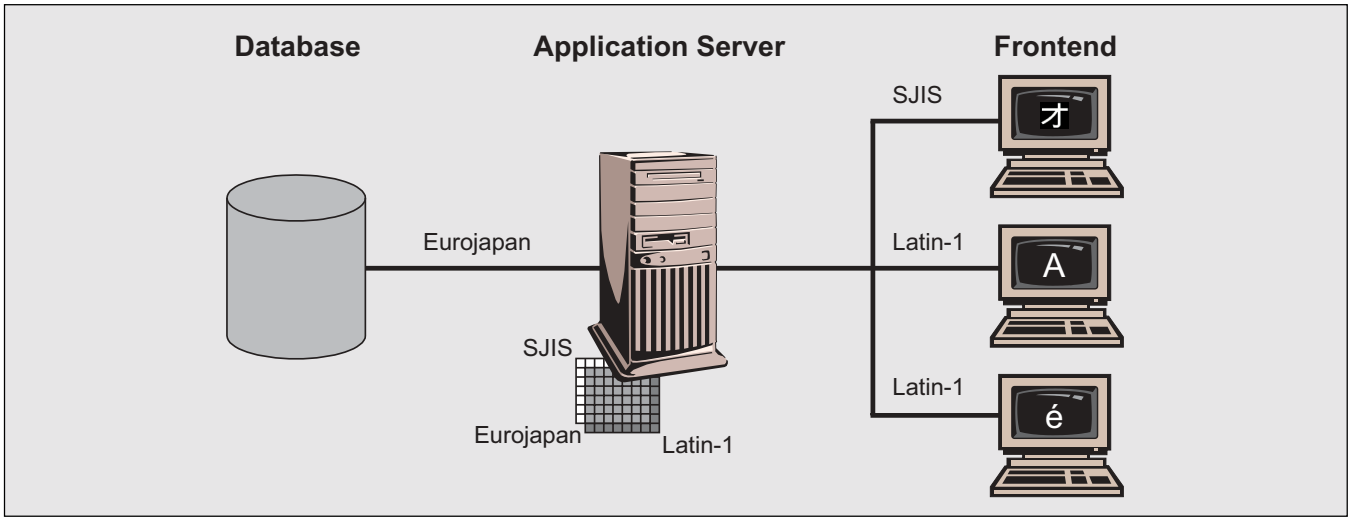


Figure 5 *SAP Blended Code Pages*

SAP Code Pages	Supported Languages	Unambiguous or Ambiguous?
Asian Unification^{C}	English, Japanese*, Chinese	Unambiguous
Asian Unification^{K}	English, Japanese*, Korean	Unambiguous
Asian Unification^{T}	English, Japanese*, Taiwanese	Unambiguous
Nagamasa	Japanese*, Thai, English	Unambiguous
Eurojapan	German, English, French, Italian, Danish, Dutch, Finnish, Norwegian, Portuguese, Spanish, Swedish, Japanese*	Unambiguous
Silk Road	Japanese*, English, Greek	Unambiguous
Trans Siberian	Japanese*, English, Russian	Unambiguous
Asian Unification	English, Japanese**, Chinese, Korean, Taiwanese	Ambiguous
Diocletian	German, English, French, Italian, Danish, Dutch, Finnish, Norwegian, Portuguese, Spanish, Swedish, Greek	Ambiguous
SAP Unification	German, English, French, Italian, Danish, Dutch, Finnish, Norwegian, Portuguese, Spanish, Swedish, Japanese*, Czech, Hungarian, Polish, Slovakian, Romanian, Slovene, Croatian, Turkish, Greek	Ambiguous

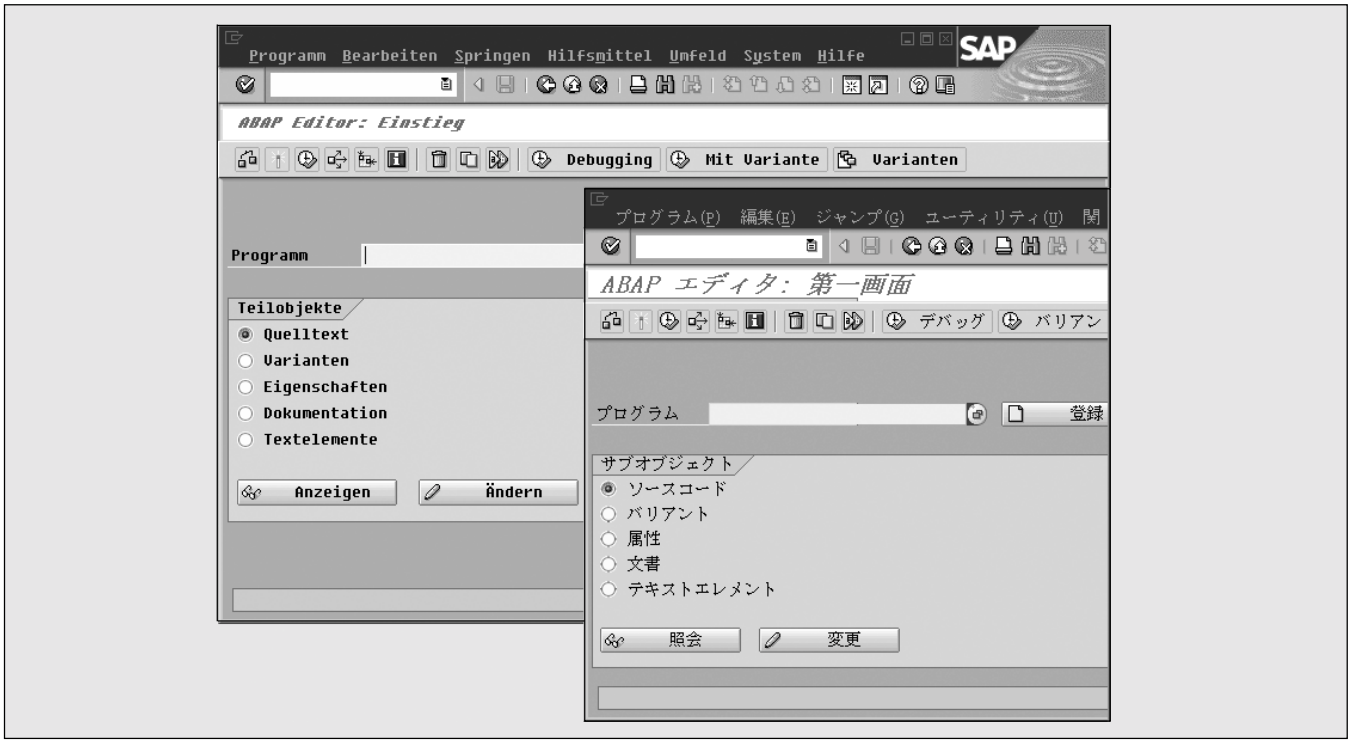
* Single-byte Katakana and some level 2 Shift-JIS Kanji (first byte >H'E0) are not supported.
** Single-byte Katakana is not supported.

page is selected according to the login language. The same Japanese front code page is used on both an SJIS system and a blended code page system. (The code pages used by the printer are ignored for the moment.)

With the Eurojapan blended code page, English, Japanese, and French users of the same SAP system can use their native languages and scripts as they access data and applications. While all users are accessing characters from the same code page, they must still specify the language they wish to use when they log on. They may find that the frontend does not represent certain characters correctly, however, because there is no blended font for the frontend. Blended code pages are for characters, not glyphs, which means they have certain "frontend" limitations, which I will discuss in just a moment.

A list of SAP blended code pages is given in **Figure 5**. The third column in Figure 5 indicates whether the blended code page is *unambiguous* (i.e., each code point refers to exactly one character) or *ambiguous* (i.e., two characters can share the *same* code point). Normally, and ideally, each code point is reserved for one character, but double occupancy is possible under specific circumstances. French uses the character /û/ and Hungarian uses the character /ű/, but no language uses both characters simultaneously. Mapping /û/ and /ű/ to the same code point allows more languages to fit in a single code page. The ambiguity is resolved by the login language, which determines whether /û/ or /ű/ is the correct character. Should a French user inadvertently look at Hungarian texts, some of the characters will be incorrect, but this does not affect data in the database. A list that contains Hungarian and French texts, such as a list of international customers, will not display or print correctly, however, since /û/ and /ű/ cannot be used together. Unambiguous blended code pages have had a low acceptance in Japan because of the limited set of characters available.

Figure 6 Eurojapan mySAP.com Logon Languages JA and DE



Unambiguous and ambiguous code pages are used in productive SAP systems, and blended code pages have been a good interim measure for SAP to provide internationalization. (Unicode, which is the topic of my next article, is the best way to achieve internationalization.) A Japanese user and a German user can log on to the same SAP system in their own respective languages (see **Figure 6**). Users have access to all languages included in the blended code page, and can access all characters in the code page if needed. A Japanese user, for example, who wants to check the stock of widgets in several warehouses, would see the following on the screen:

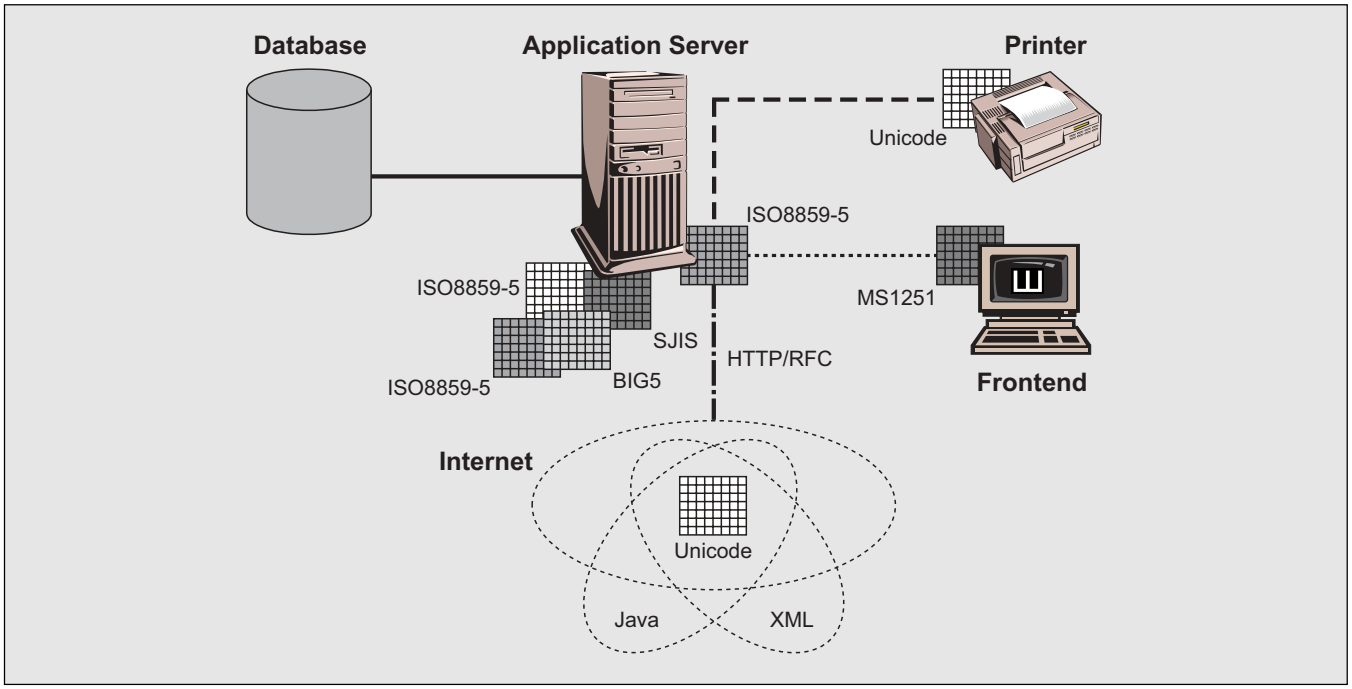
Widgets	
New York	1098
M#nchen	104
東京	95
Orl#ans	28129

Here we see the limitation inherent in blended code pages. The characters for English appear correctly, because every standard code page contains 7-bit ASCII characters, but the German /ü/ and the French /é/ are not in the frontend Japanese code page, so the glyphs are incorrect. Suppose in the process of working with the widget data, the ABAP command `TRANSLATE widgets TO UPPER` converts all of the warehouse names to uppercase — the Japanese user would then see the following on the screen:

Widgets	
NEW YORK	1098
M#NCHEN	104
東京	95
ORLEANS	28129

The # does not have any corresponding uppercase character, but the conversion to uppercase did occur. When a German user looks at the table, the Japanese characters will not appear correctly, but the user will also see that the conversion to uppercase was complete:

Figure 7 *The MDMP System*



Widgets	
NEW YORK	1098
MÜNCHEN	104
gqN#	95
ORLEANS	28129

Blended code pages have additional shortcomings. They do not contain single-byte Katakana characters, which are used by Japanese banks and public institutions. In addition, new locales must be built for each new platform. Once a blended code page has been implemented, it is impossible to add additional code pages — for example, to add BIG5 in addition to Eurojapan. Ambiguous code pages can also have an additional disadvantage if all users use English as a login language and the SAP system is to be converted to Unicode. Because some language-specific data does not contain a language key, which is a table key that specifies the language of the text data, some data may have to be checked.[4] Blended code pages provide greater support for multiple languages, but the size of the code page still restricts the number of code pages that could be blended. The next development, Multi-Display/Multi-Processing (MDMP) made multiple code pages available as of Release 3.1.

MDMP Code Pages

In an MDMP system, the code page used on the application server is selected dynamically, according to the user's login language. Because the frontend may use proprietary code pages, such as Microsoft, the frontend code page can be different than the code page on the application server, provided that the set of characters in both code pages is (nearly) identical. The code page on the application server is still converted to and from the frontend code page, depending on the direction of data transfer.

For example, as shown in **Figure 7**, a user logs on to the SAP system as a Russian user,

[4] This process will be discussed in the second installment of this two-part series, which will appear in the next edition of *SAP Professional Journal*.

and the application server selects the ISO8859-5 (Cyrillic) code page, which is converted to MS1251 for presentation on a Windows frontend. The user then enters text, which is converted from MS1251 to ISO8859-5 on the application server. All frontends (e.g., SAP GUI, Java GUI, HTML GUI) make these conversions automatically. Recall that the frontend code page can be different than the system code page as long as the two code pages are compatible, which is the case with these two code pages.

MDMP makes it possible to use multiple standard code pages, which increases the number and combinations of languages that can be used. Additional code pages can also be added to an existing system, and MDMP gives SAP customers the option of increasing the number of supported languages whenever necessary. In an MDMP system, languages with the same code page can be used in parallel and concurrently, but *more than one code page cannot be used simultaneously*. Local data, which has a language key for text fields, is processed in the appropriate language, regardless of the user's language.[5] Global data — that is, data *without* a language key — must use characters common to all code pages on the system (7-bit ASCII), or else the data may not be processed correctly and may present a problem when converting to Unicode.

> **Warning!**
>
> *Language keys should not be used for any other purposes! Even if a system only uses one language, the language keys should not be used, for example, to distinguish warehouse locations. This will lead to system errors.*

Despite the advantages of MDMP, there are some potential drawbacks. Although unlikely, it is conceivable that the key for a table could become non-unique if the table key does not have a language key and the characters in the table key are from different code pages but have the same code points.

On an MDMP system, Japanese, Chinese, or Russian users need to log on in their own language because those language characters are not available when they are logged on as English users. Fields that contain language-specific data must be translated into the user's login language. R/3 customers must also translate all language-specific data so it is visible to all users: this leads to large data redundancies and system administrative overhead to fill/translate customer text data from English into every language used in the system. Lastly, migration to Unicode requires a special project, which I will discuss in part two of this series.

Rules for Using an MDMP System

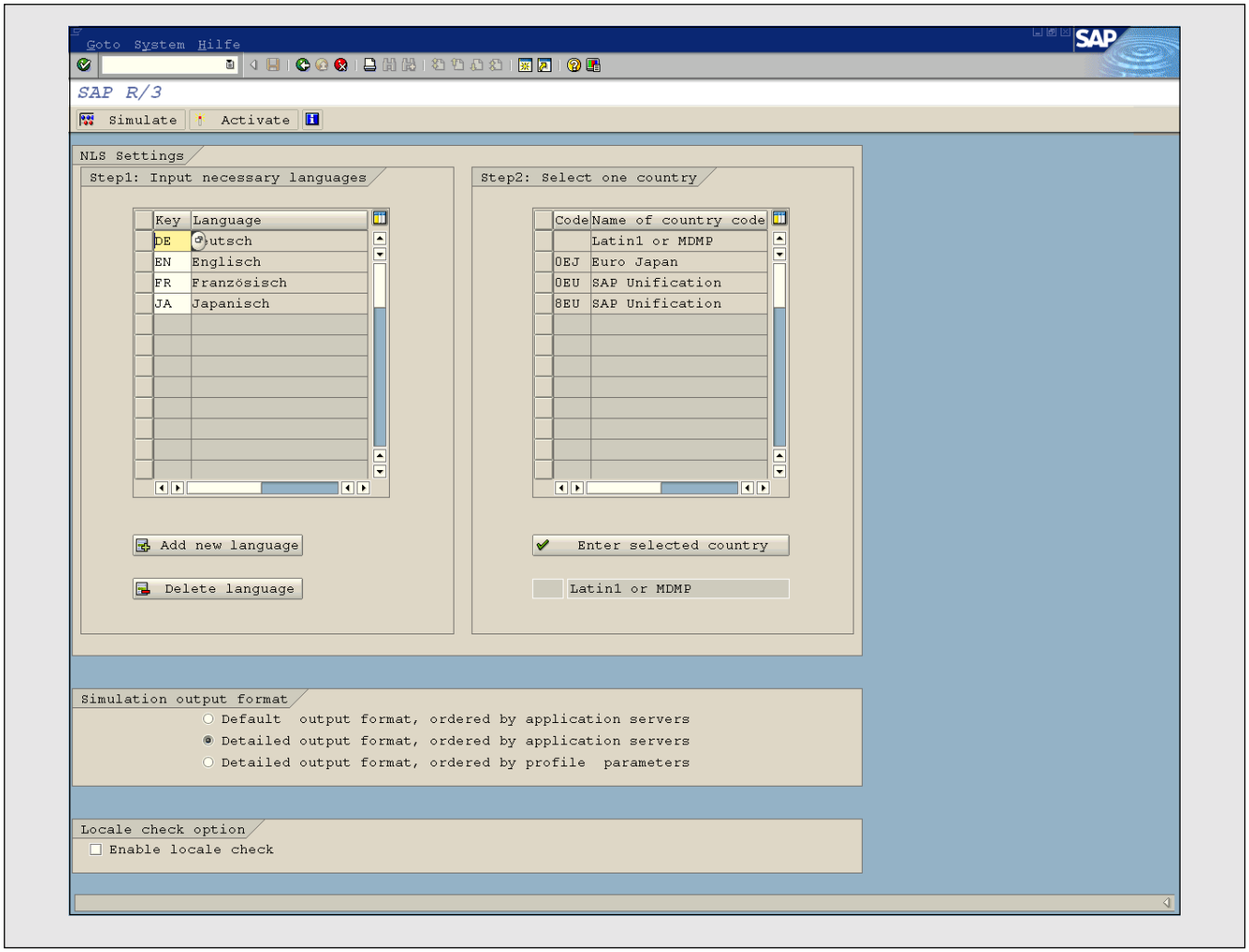
- ✓ Users may only use the characters of their login language or 7-bit ASCII.
- ✓ Global data must contain only characters occurring in all code pages, i.e., 7-bit ASCII.
- ✓ Batch processes must be assigned with the correct user ID and language.

Blended Code Pages or MDMP Code Pages — Which Solution Is Right for You?

To make it easier to determine which solution — blended code pages or MDMP code pages — is best suited for your particular needs, use the report `RSCPINST`. This report selects and installs code pages, and can help in the selection process (see OSS note 42305). Often there are several possible configurations to choose from. As a general rule, SAP recommends the configurations in this order: a single standard code page, a non-ambiguous blended code page for new installations, and an MDMP system for

[5] As of Release 3.0F, a language key was added to tables that contained language-specific data.

Figure 8 *Report RSCPINST*

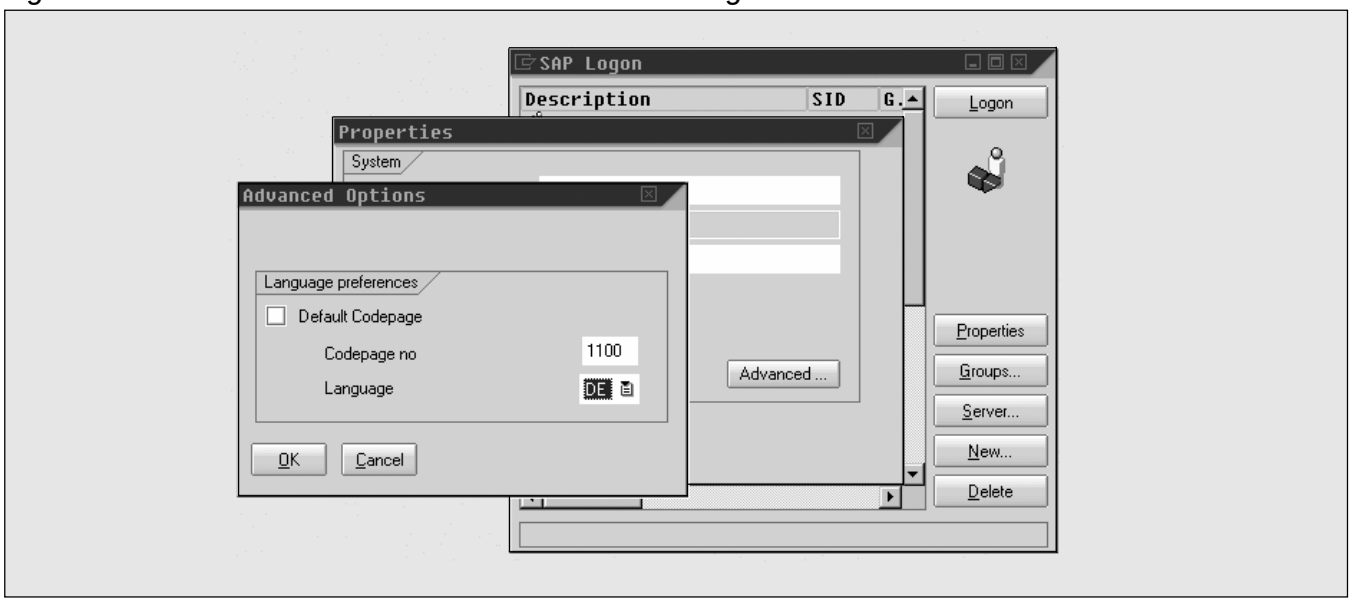


any other situations, e.g., adding a new code page to an existing single code page system. The best configuration will vary from customer to customer, and can depend on other factors, such as whether additional languages or code pages will be needed in the future (see OSS note 73606 for detailed information).

The report tool checks all the profile parameters, the settings of the tables `TCP0D` (the SAP code page catalog) and `TCPDB` (the code page used in the database), and the required locales (see **Figure 8**). At the upper left of the screen, the required languages are selected, and the right-hand side indicates which code pages are appropriate. A customer who needs German, English, French, and Japanese can set up an MDMP code page, or use the blended code page Eurojapan or SAP Unification. It is also possible to simulate the setup to see which profile parameters need to be changed.

Now that we have gone behind the scenes to take a look at what happens in the SAP system when using character code pages, SAP blended code pages, and Multi-Display/Multi-Processing code pages, let's shift our focus to the frontend. In the next two sections, we'll examine how things work on the frontend by looking at what the user sees — on the screen and on paper.

Figure 9 Set Default Code Page: Frontend



The Frontend: What the User Sees on the Screen

On the frontend, the keyboard mapping, the input method, and the screen output must match one another. Prior to 4.6C, users were required to select a frontend code page when they logged in, which created a potential for handling errors and for incorrect settings (see **Figure 9**[6]).

The potential for error derives from the fact that it was possible to select the application code page as the frontend code page. You would think that would obviate the need for a conversion process between the two tiers, but because there may not be a one-to-one correspondence between the two pages, loss of data could occur. In the ISO8859-1 code page, the code points 80-A0 do not contain printing characters, but these code points do contain characters in the Microsoft code page MS1252 (e.g., 9A = š). The character /š/ would appear correctly on the screen, but the semantics of the character are incorrect, and ABAP commands such as `TRANSLATE TO UPPER CASE` would fail.

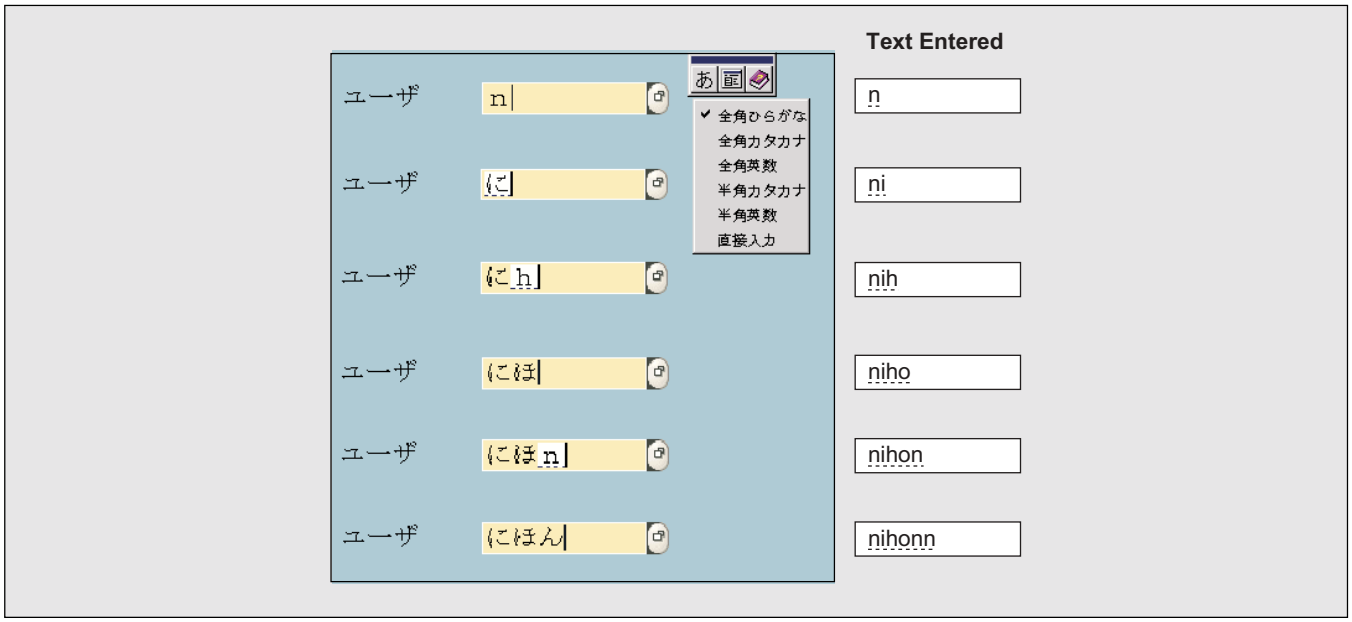
To prevent this from happening, as of Release 4.6C, the relationship between the language key and the locales is defined system-wide, which prevents incorrect settings. A user selects a login language, and then the matching frontend code page is sent to the frontend. Data is automatically converted from the application code page to the frontend code page and the user is therefore insulated from code page selection.

On the frontend, a Japanese user naturally wants to enter Japanese characters, which means an import editor is needed. It is possible to purchase a localized entry editor, but until recently the only way to use the Japanese input method under Windows was with a localized Japanese language version of the frontend OS. There are third-party programs as well, but they vary greatly and there is no standard. The Global Input Method Editor (GIME) from Microsoft enables users to enter Japanese, Chinese, Korean, and Taiwanese characters with an English OS. GIME is fully supported by SAP GUI 4.6C and the Windows GUI. For example, a Japanese user can enter Japanese with GIME by typing phonetically on a QWERTY[7] keyboard;

[6] The screen shot has been edited to remove other advanced options, which are not relevant here.

[7] The standard English language keyboard configuration (the first six letters on the upper row of the keyboard spell "QWERTY").

Figure 10 Entering Japanese Characters with GIME



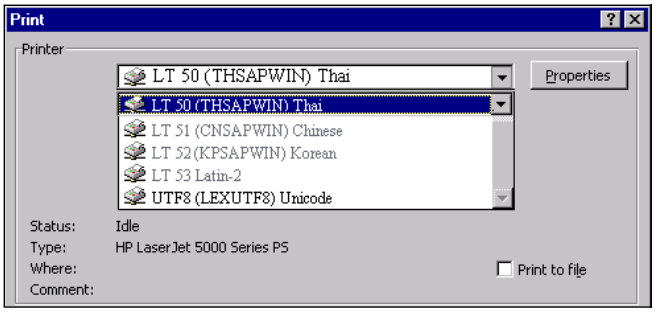
the “letters” are then converted into Japanese characters on the fly.

Figure 10 shows the step-by-step process of entering the word <nihonn> — “Japan.” On the right, the text entered is given. The [n] first appears on the screen because there is no Japanese character that corresponds to it. After typing the <i>, the sequence /ni/ is then converted into Japanese characters, and so on. (The small pulldown menu is GIME help.)

Printing: What the User Sees on Paper

Prior to Release 4.6C, each printer supported only one code page or a small number of related code pages, which restricted the number of languages that could be printed. Localized printers exist, but they were not available in the US and almost impossible to procure. Moreover, an international company that wanted to print yearly financial reports at its headquarters would have to somehow find, set up, and maintain a dedicated printer for each language, at great expense.

SAP has developed software to enable high-speed printing of all languages on a single, English printer. The exact details are described in OSS note 0083502, but in essence several “virtual” printers are set up, and they print on the same hardware using “soft fonts,” i.e., 24-bit pixel bitmaps. This is much faster than graphic printing. A Thai user selects a printer — which is, in fact, merely a device type — and can print Thai characters on the same printer that a Chinese coworker has just printed on:



For the user, there are multiple printers to choose from, but there is only one physical HP LaserJet printer. One disadvantage is that an English user could print on the ISO8859-2 (Latin-2) printer and the printout would then have incorrect characters on it.

As of Release 4.6C, SAP has offered support for printers that use a Unicode (UTF-8) interface, which at the moment are only produced by Lexmark. With Unicode, "one printer does all" and there are no additional hardware costs, and there is no need for the user to select a printer based on the language they want to print, thereby eliminating handling errors. Printing speed and print quality are also greatly improved because soft fonts are not used.

Conclusions

To improve the internationalization of the SAP system, either new code pages had to be introduced or more code pages had to be made available, and both options have been successfully implemented. With custom-built blended code pages, users have the advantages of a single code page, but the number and combination of languages is still limited by the size of a single code page. An MDMP system uses multiple, standard code pages, but users are limited to one code page at any given time. There has been some significant progress, but ideally, there would be a single, standard code page that could support all of the world's languages. Such a code page does exist, and in my next article, I will discuss Unicode-based mySAP.com.

Acknowledgements

I would like to thank Nobuyoshi Mori and Dr. Christian Hansen for their comments.

