ORACLE

Oracle® Documaker

Unicode Reference

Part number: E51709-01

March 2014

CONTENTS

Preface

Oracle Documaker is a powerful, adaptive enterprise document automation platform used worldwide to acquire, create, manage, and present structured, on-demand, and interactive customer communications. It is designed to put power in the hands of business users, giving them the flexibility to create interactive, dynamic documents on demand.

Spanning the entire business lifecycle, Oracle Documaker helps you manage customer communications enterprisewide—including document production, correspondence, and cross-selling campaigns—across all locations and lines of business. The industry-leading platform offers a cost-effective way to address the design, production, and multichannel distribution of a broad spectrum of customer-facing documents. With robust functionality and cutting-edge technical capabilities, it maximizes efficiencies, ensures compliance, and enhances customer service.

Oracle Documaker is based on open standards and integrates easily into today's service-oriented architecture environments. It integrates with any type of system across the enterprise. It can even be integrated with your self-service Web portal so stakeholders can get immediate access to up-to-date information. Oracle Documaker provides the agility and flexibility you need to roll out new products quickly and remain competitive.

Oracle offers proven tools and migration methods, along with experienced, highly trained technical personnel to ease conversions while maintaining the intelligence of your data. Leveraging Oracle Documaker as a single system can dramatically reduce costs. One insurance customer recouped the full cost of an Oracle Documaker implementation within nine months.

Business users can easily author content in Oracle Documaker Studio using Microsoft Word through a plug-in that leverages the power of Documaker Studio in the background. For even more capability, Oracle Documaker's intuitive, easy-to-use design tool, Documaker Studio, empowers business users to create powerful, persuasive content minimizing their reliance on IT, so you can produce dynamic, *intelligent* transactional documents that transmit data and content.

AUDIENCE

Unicode provides a way to enter content in the major language, by assigning a unique number to each character. This document provides an overview of how Documaker uses Unicode. It will be useful to those who want to understand how to use Unicode with your Documaker systems.

Documentation Accessibility

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Related Documents

For more information, refer to the following Oracle resources:

- Documaker Installation Guide
- Documaker Administration Guide
- Documaker Studio User Guide
- Printers Reference
- Fonts Reference
- Documaker Desktop Installation Guide
- Documaker Desktop Administration Guide
- Doc Factory Guide
- WES Guide

Conventions

The following text conventions are used in this document:

Convention	Description
bold	Indicates information you enter.
italic	Indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Indicates commands, URLs, code in examples, and text that appears on the screen.

Chapter 1

Using Unicode

This document discusses how to use Unicode with Documaker and includes information on these topics:

OVERVIEW

What is Unicode?

Basically, computers understand only numbers. Through various encoding systems (code pages), computers can associate numbers with characters for the purposes of display, printing, and processing text. Originally, encoding systems mapped values in a single byte (values between 0 and 255). Such a system (known as a Single Byte Character Set or SBCS) can only address up to 256 characters. The European Union alone requires several different encodings to cover all its languages.

Far Eastern languages (Japanese, Chinese, and so on) contain 1000s of characters – too many for a SBCS. As a result, encoding systems using a mixture of one and two byte values were developed. These encoding systems are known as Double Byte Character Sets (DBCS) but would more accurately be called Multibyte Character Sets (MBCS).

In all, there are literally hundreds of encoding systems. It is not surprising that these encoding systems conflict with one another. That is, the character that should be associated with a number (computers only understand numbers) changes depending on the encoding system.

Unicode was developed to provide a single encoding for scripts for world's major languages. Unicode provides a unique number for every character. Industry leaders such as IBM, Microsoft, HP, Oracle, SAP and many others have adopted the Unicode Standard. See the Unicode web site for more information on the Unicode Standard:

www.unicode.org

Related standards

XML (eXtensible Markup Language) is a markup language for interchange of structured data. The Unicode Standard is the reference character set for XML content.

UTF-8 (Unicode Transformation Format, 8-bit encoding form) is a format for writing Unicode data in text files (which are normally processed sequentially, one byte at a time). Unicode values (code points) are written as a sequence of one to four bytes. When writing Unicode data into Documaker text files (FAP files, and so on), the Unicode data will be written in UTF-8 format.

Implementing Unicode

Documaker provides the ability to compose, print, and present documents for Far Eastern languages like Chinese, Japanese, Korean, and Vietnamese. Accordingly, it lets you:

- Use the Unicode 3.0 character set (two-byte Unicode values)
- Compose FAP files containing Unicode text
- Run the Documaker batch process to assemble documents from FAP files and data that contain Unicode text (on a limited set of platforms)
- Print documents with Unicode (using a limited set of printers)

- Archive, retrieve, and view the documents, including thick-clients on Windows, and also PDF over the Internet

There are some limitations:

- All user interfaces, help, error messages, and documentation for tools and runtimes would still be single-byte ANSI characters, primarily in English.
- The target user is an English-speaking developer that needs to be able to create documents for languages other than English.
- Names of objects (field names, section names, and so on) will still be single-byte ANSI characters, primarily in English. This means that you should not convert Unicode text to a field in Documaker Studio.
- Limited formatting or word-wrapping or other syntax (for example - justification) related functions for Unicode text in both editing text and mapping data into multiline text fields.
- No support for vertical writing style used in Japanese newspapers, magazines.
- No support for bi-directional languages (Arabic, Hebrew, and so on) for releases prior to 12.2 and higher. Note, this feature (support for bi-directional language) is not backward compatible, so if you generate resources containing text in these languages, they cannot be edited or used by development or runtime systems in a prior Documaker version.
- No support for Unicode data entry in Documaker Desktop.
- Methods supported for input of Unicode data in Documaker Studio data entry check \ wip edit plug-in:
  - copy/paste
  - keyboard entry
- A Windows system is needed in order to develop and maintain Unicode forms.
- Production runtime platforms that support Unicode will be Windows and Unix platforms supported by Documaker
- Printer support for Unicode forms will be through GDI (Windows print drivers), PCL (via PCL6 driver), PDF (via TrueType font support). Unicode support is not available for Metacode print driver.

Documaker version 12.2 and higher includes support for Complex Text Layout (CTL) languages such as Arabic, Hebrew, Thai, Devanagari, and Tamil. A Complex Text Layout (CTL) language is any language which stores text differently from how it is displayed.

Many CTL languages such as Arabic and Hebrew, use bidirectional script. Words and sentences are written from right to left. Some text, such as numbers and Roman-based words, are written from left to right.

Some CTL languages are context dependent. In Arabic, characters can be modified depending on the preceding and following characters. Written Thai uses character clusters, a collection of syllabic elements denoting consonants, vowels, and tonal values.

CTL languages require complex transformations to properly display or print characters. The fundamental characteristics of CTL languages are:

- Bidirectional text--Arabic and Hebrew are written right to left, but numbers and Roman-based characters are written left to right.
- Contextual analysis--each Arabic character has up to four display representations. The representation glyph depends on its position in the text.
- Ligatures--combinations of two or more Arabic characters can form a different, single character.
- Diacritics--diacritical marks placed above, below, or inside vowels and consonants modify their tonal value.
- Character clusters--syllables and cells, especially in Thai, are composed of several alphabetic elements, including vowels, consonants, diacritics, and tone marks.
- Number and date formats--some Arabic countries use Hindi digits rather than Arabic digits, and the Islamic calendar rather than the Western calendar.

Note There are no rules or DAL (Document Automation Language) functions in Documaker to specifically to support additional number and date formats for Hindi digits or the Islamic calendar.

Eventually, Oracle Insurance plans to create fully internationalized versions of the products. This involves translating the user interface, online help, error messages, and documentation, for tools and runtimes.

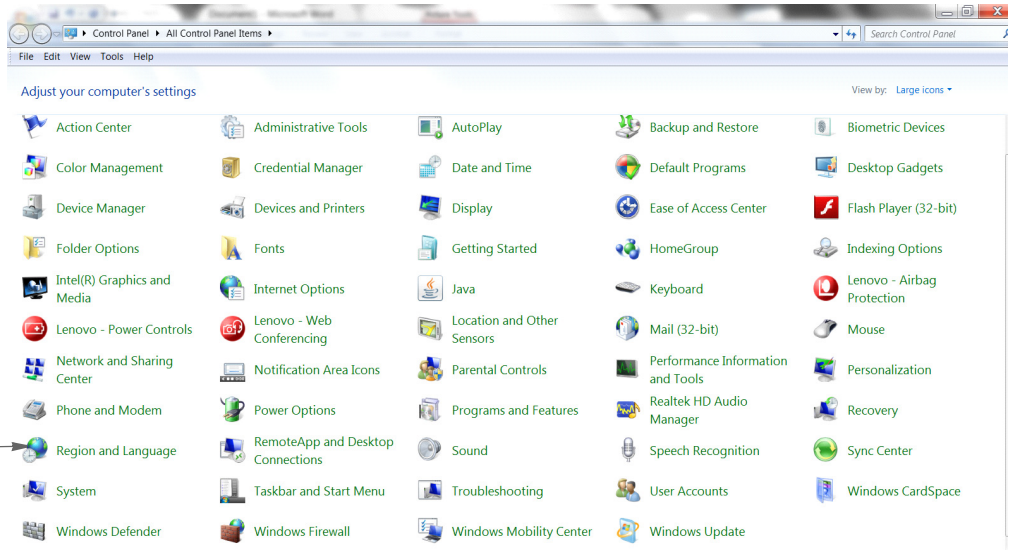
Experience shows that having the runtime version translated is probably more critical than having the development tools translated. Therefore, translation focus will be on the runtime system before the design tools.

ENABLING LANGUAGES

The Documaker Studio and Docucreate tools require Windows to develop Unicode forms. To type characters in the languages you will be using on forms, you may need to enable these languages in Windows.

To enable additional languages in Windows, open the Windows Control Panel and double-click on the Regional and Language Options icon (the name of the icon will vary somewhat depending on the version of Windows you are using).

For instance, in Windows 7, double-click here

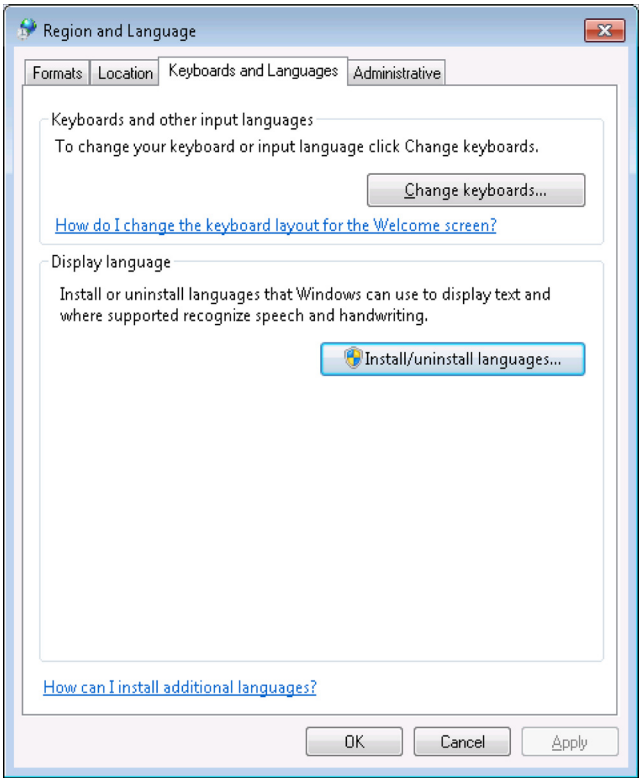


The remaining steps differ slightly, depending on which version of Windows you use.

If you use	See
Windows 7	*Choosing Languages in Windows 7* on page 14

CHOOSING LANGUAGES IN WINDOWS 7

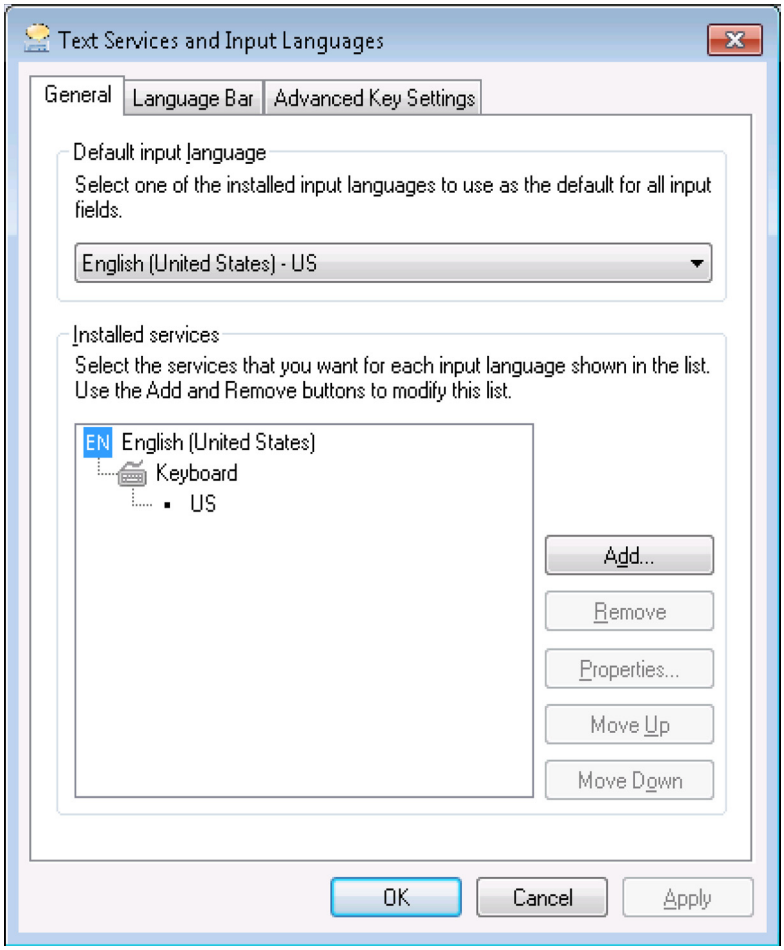
In Windows 7, the Regional and Language Options window appears as shown below. Click the Keyboards and Languages tab. Click on the Install/uninstall languages button to install additional languages.



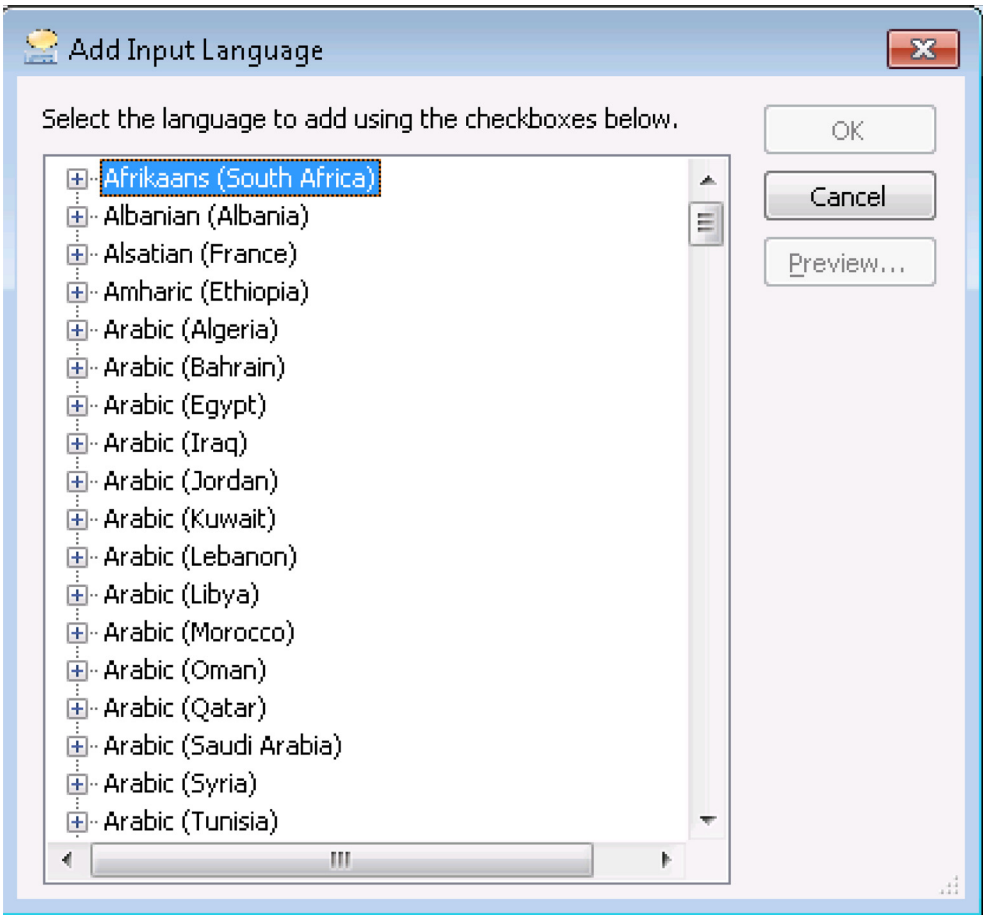
In Windows 7, you can download languages in two ways:

- Using Windows Update. If you're running an Ultimate or Enterprise edition of Windows, you can download available language packs by using Windows Update. Language packs installed using Windows Update provide a fully translated version of Windows dialog boxes, menu items, and help content.
- You can download Language Interface Packs (LIPs) from the Microsoft Download Center. To install a LIP, you will need to have the required parent language installed on your PC.

After installing additional languages, return to the Regional and Language Options window, click on the Keyboards and Languages tab and then click the Change keyboards button to view or change the languages and methods you can use to enter text.



Click Add to add additional languages you want to input. Then select the input language and the keyboard layout/IME (input method editor) you want to use.



SELECTING THE INPUT LANGUAGE

Once you have selected the input languages you want to use, you should test the new input languages in a simple environment such as Microsoft Word. On the task bar, you should see a small icon with the letters, *EN* (assuming your original language for Windows was English), or you may see an icon for a typewriter. If you click on the *EN* or typewriter icon, you will see a list of the input languages now available in Windows.

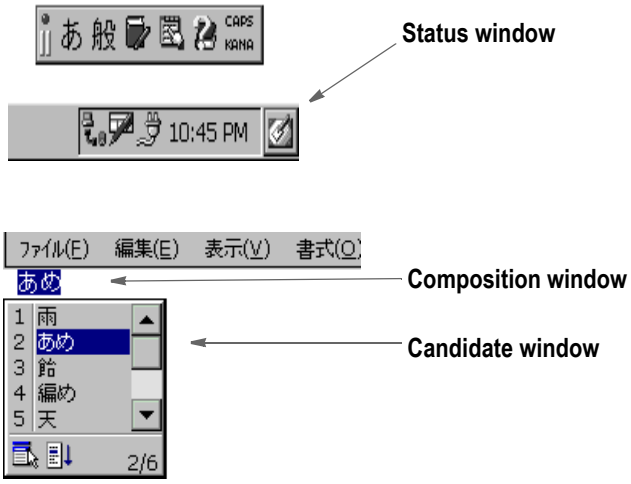
You can change input languages by clicking on the input language icon and using the mouse to select the new input language from the list provided. Alternatively, you can change the current input language by pressing and holding the Alt key and then pressing and releasing the Shift key until the desired input language is selected.

Once you have selected a new input language, you should be able to enter characters in the new input language. With some input languages, you can just use the keyboard as you normally would for English. For example, suppose you had installed and selected Russian as the current input language. If you typed the letters *ABC* on an English style keyboard, you would see these characters:

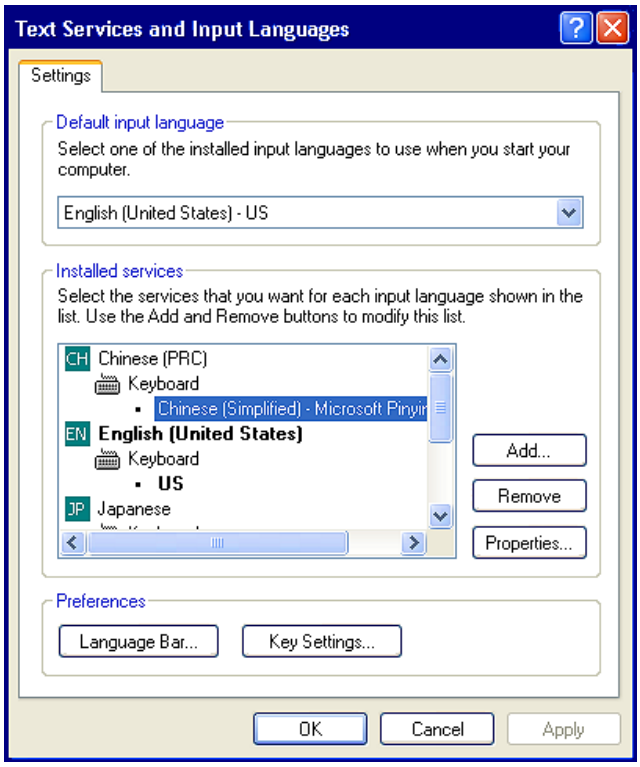
ФИС

Some input languages (like Japanese and Chinese) use an input method editor (IME) to type characters. The basic operation of an IME is as follows.

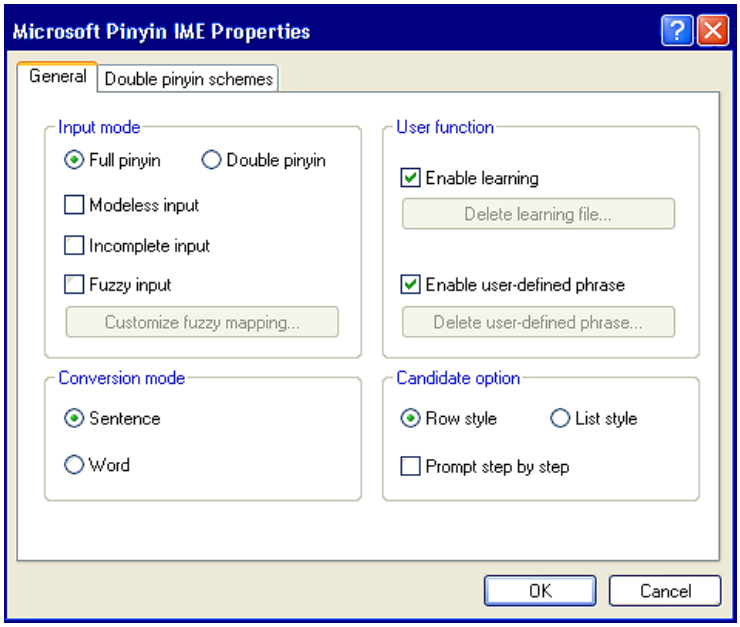
The IME user interface (UI) consists of a Status window, a default Composition window, a Candidate window, and a Guideline window. The appearance varies based on the IME you use.



As you type, characters are built into a composition string. The string appears in the default Composition window. The way you indicate the appropriate characters to use differs, depending on the IME. You can configure how the IME works by right clicking on the EN or typewriter button and selecting Settings.



Highlight the IME you want to configure and click Properties. The Properties window varies based on the IME you use.



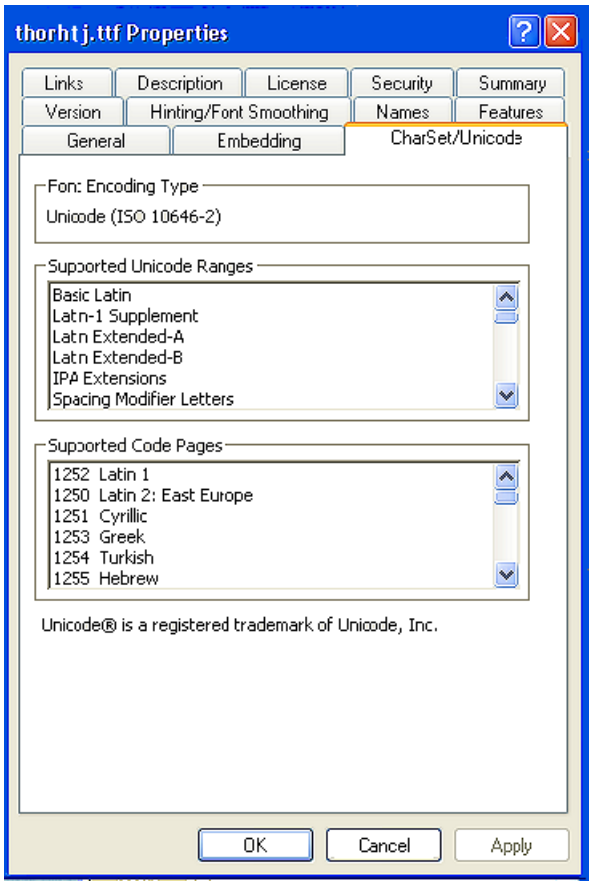
SETTING UP UNICODE FONTS

Most TrueType fonts contain Unicode encoding information. The other types of encoding formats for TrueType fonts are Symbol (used for fonts with graphic symbols), ShiftJIS (Japanese Industrial Standard), Big5 (used in Taiwan and Hong Kong and associated with traditional characters), PRC (used in Singapore and mainland China and associated with simplified characters), Wansung (Korean), and Johab (Korean). Claims of Unicode conformance in a font can be misleading when there is no common understanding about the number of languages supported.

Documaker supports TrueType fonts with Unicode encoding information. One way to determine the Unicode encoding information and languages supported for a TrueType font is to use Microsoft's Font Properties Extension utility. To find this utility, use the Search option at this web site to search for the Font Properties Extension utility:

www.microsoft.com

Once installed, you can right click on a TrueType font to get a properties window such as the following:



LICENSING FONTS

Documaker includes Unicode TrueType fonts in Albany and Andale Duospace fonts. These founds can be accessed from the Oracle Software Delivery Cloud. See Documaker Installation Guide for more details on how to locate the correct installation. To incorporate the Unicode fonts, choose the right font set and associated font cross reference file (FXR) that will support the output you need to product. Documaker offers Unicode fonts in Albany and Andale Duospace fonts for Japanese, Simplified Chinese, Traditional Chinese, and Korean languages.

Typeface	Language	File name
Albany WT J	Japanese	ALBANWTJ.TTF
Albany WT K	Korean	ALBANWTK.TTF
Albany WT S	Simplified Chinese	ALBANWTS.TTF
Albany WT T	Traditional Chinese	ALBANWTT.TTF
Andale Duospace WT J	Japanese	ADUOJ.TTF
Andale Duospace WT J Bold	Japanese	ADUOJB.TTF
Andale Duospace WT K	Korean	ADUOK.TTF
Andale Duospace WT K Bold	Korean	ADUOKB.TTF
Andale Duospace WT S	Simplified Chinese	ADUOSC.TTF
Andale Duospace WT S Bold	Simplified Chinese	ADUOSCB.TTF
Andale Duospace WT T	Traditional Chinese	ADUOTC.TTF
Andale Duospace WT T Bold	Traditional Chinese	ADUOTCB.TTF

The reason for different TrueType fonts that target the Japanese, Simplified Chinese, Traditional Chinese, and Korean languages is that the same character is represented using different shaped glyphs. For example,

Simplified Chinese	Traditional Chinese	Japanese	Korean
与	与	与	与
今	今	今	今
全	全	全	全
才	才	才	才

Typically, using the rel121j.fxr and the Japanese font set would support most languages. However, since the shapes of the characters used for certain languages are unique you may need to use one of the other provided fonts/font cross reference files. Once you have selected the desired font set, add these to the Documaker Studio workspace by following these steps:

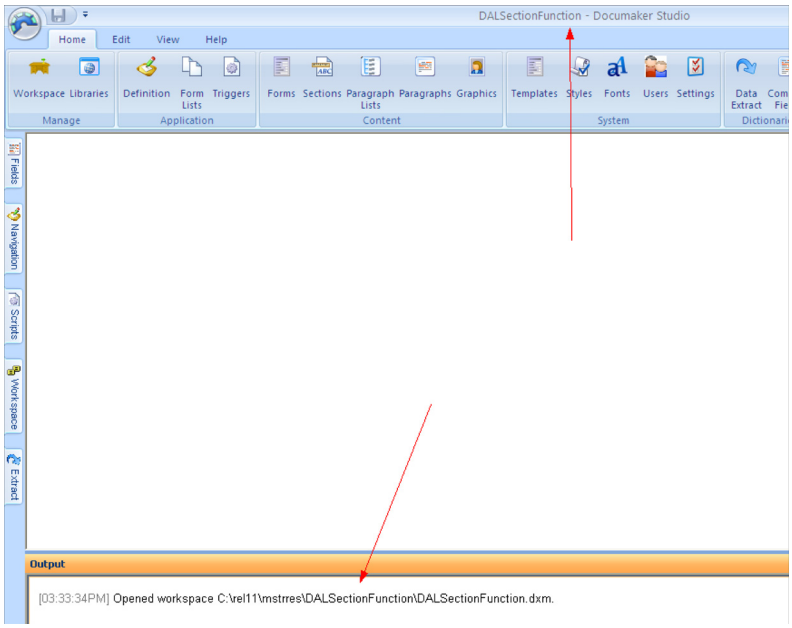
Changing the FXR in an Existing Workspace

These steps need to be performed by a user with System Administrator rights to the workspace or a user who has permission to access Business Application Definition Manager.

Prerequisites:

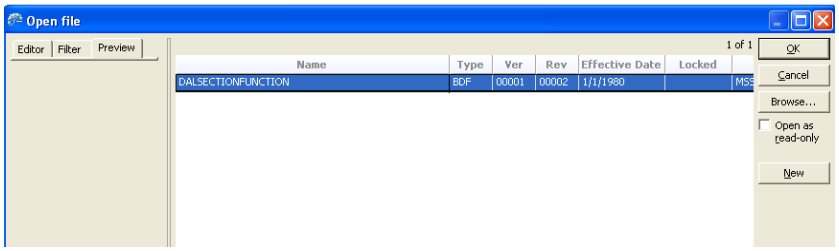
- Assumes you have installed Documaker Studio and Documaker JAPAC fonts
- Assumes you have an existing Documaker Studio workspace

1. Copy rel121j.fxr from location where you installed the JAPAC fonts to the \deflib directory of your Documaker Studio workspace.
2. Start Documaker Studio
3. Click on the button in the upper left corner to display the File menu.
4. Select Open Workspace
5. Browse to the location of your workspace
6. Double click on the XXX. DXM file in the root directory of the workspace (where XXX represents the workspace name)
7. The workspace is opened.
8. Workspace name is displayed in the title bar and in the output area

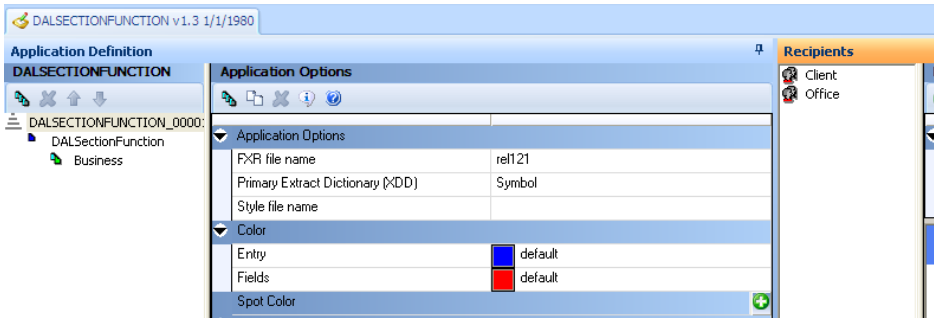


9. Check the Business Application Definition file out of the library.

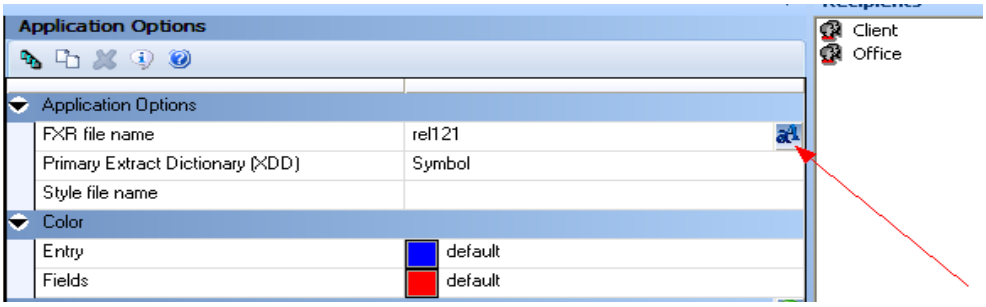
10. Double click on the Definition icon in the Application tab
11. Open file dialog displays



12. The BDF file should be selected\highlighted.
13. Click OK
14. The BDF file is checked out of the library
15. Application Options panel is as below:

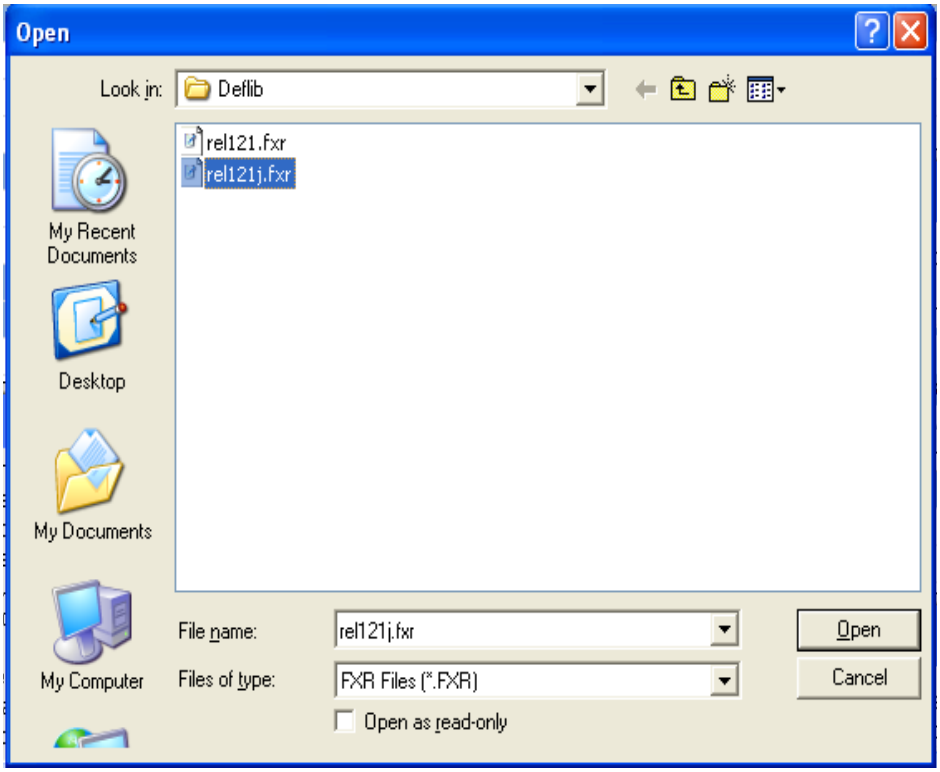


16. Click on the field to right of the label FXR file name
17. Click on the icon that displays in that field

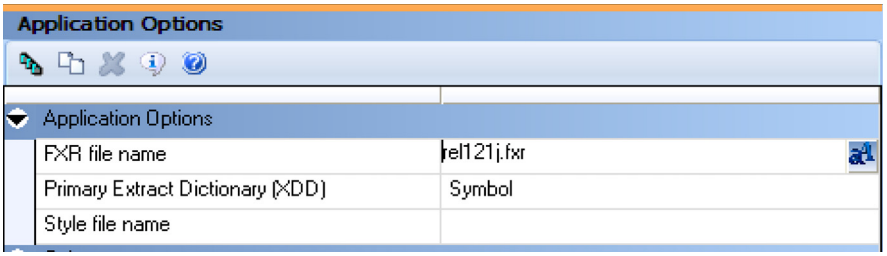


18. The Open dialog displays
19. You should be pointing to the \deflib directory for this workspace.
20. If you are not, browse to the \deflib directory for this workspace

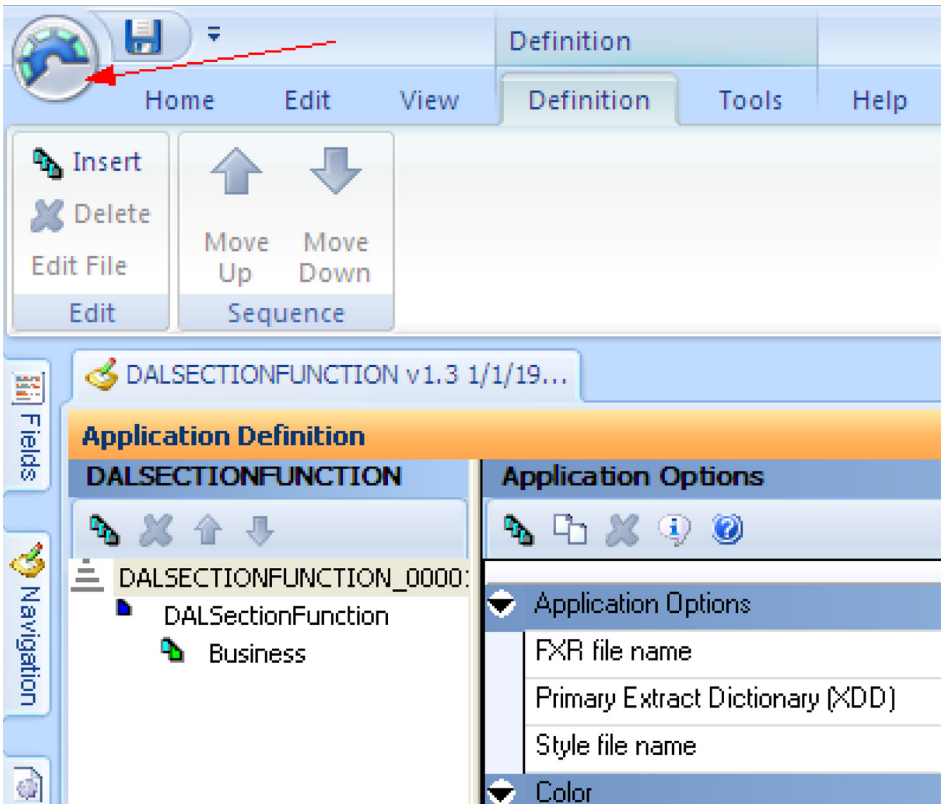
21. Select the rel121j.fxr, click Open



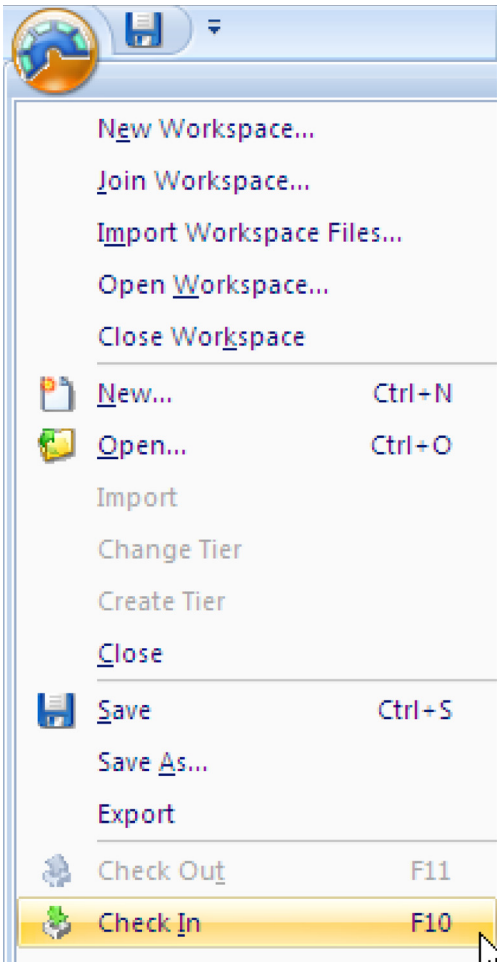
22. The FXR file name show should now show rel121j.fxr



23. Click on the Documaker button in the upper left corner to display the File menu.

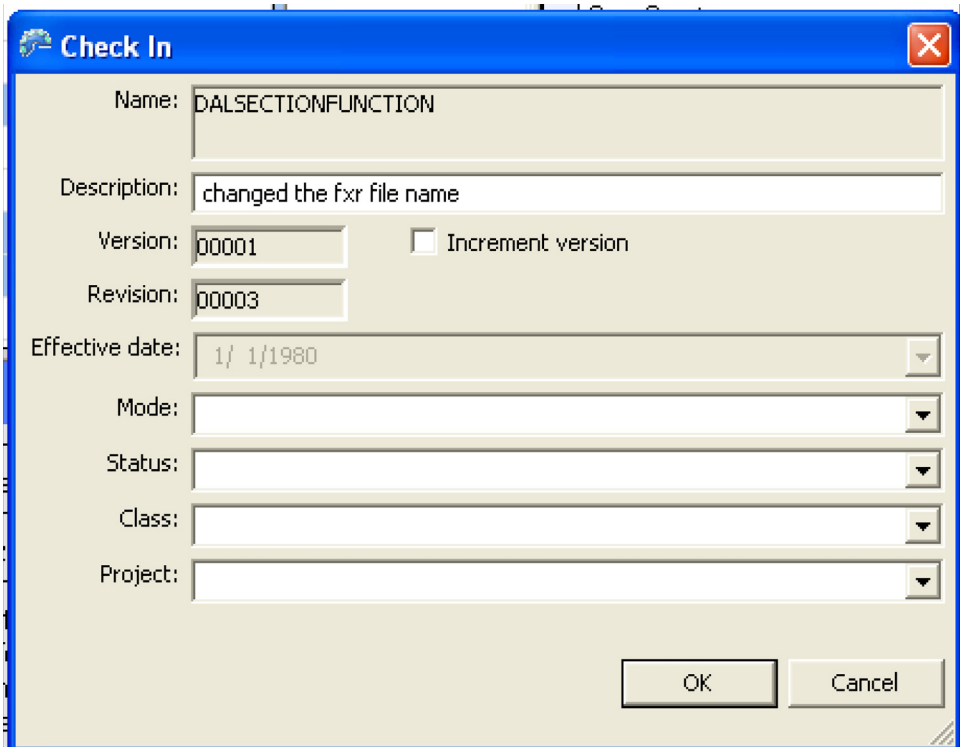


24. Select Check In



25. The Check In dialog displays

26. If desired, enter a description. For example, you could enter changed the fxr file name



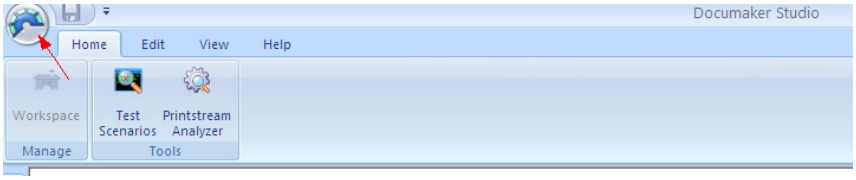
27. Click OK
28. The BDF is checked back into the library

Creating a new Workspace (without import) using Rel121j.fxr

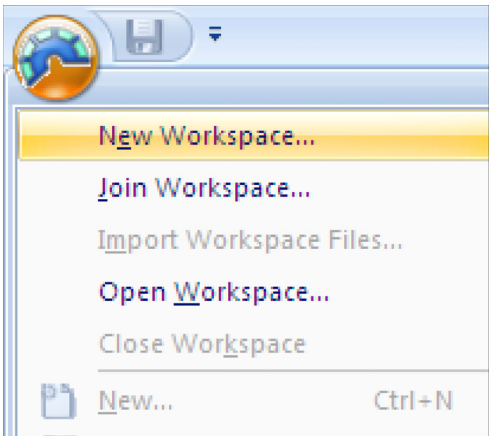
Prerequisites:

- Assumes you have installed Documaker Studio and Documaker JAPAC fonts

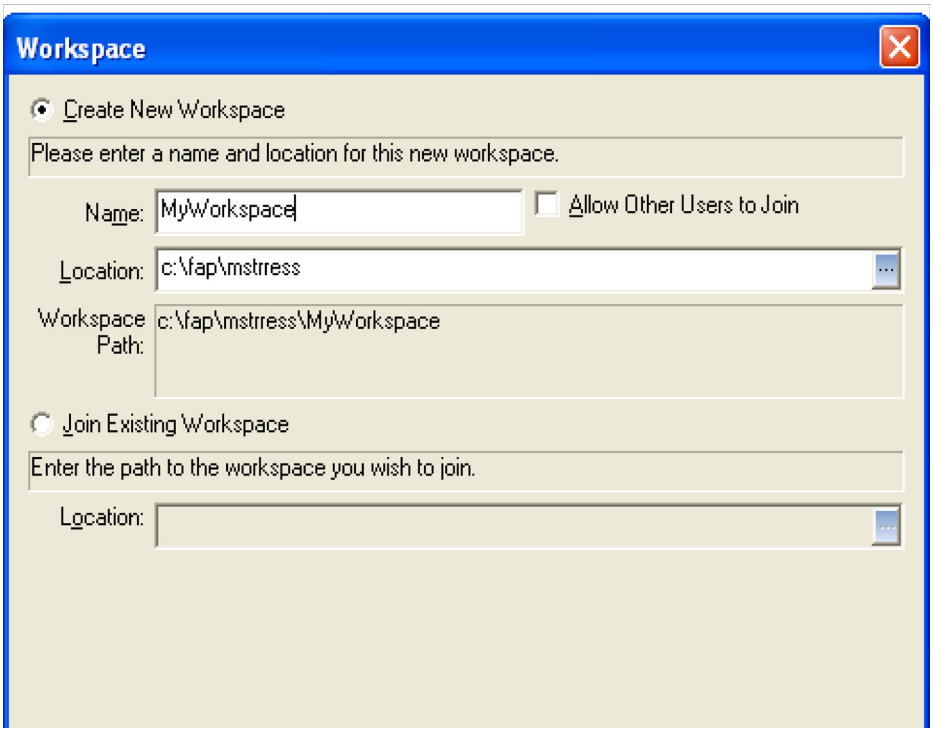
1. Start Studio
2. Click on the button in upper left corner to display the File menu



3. Select New Workspace

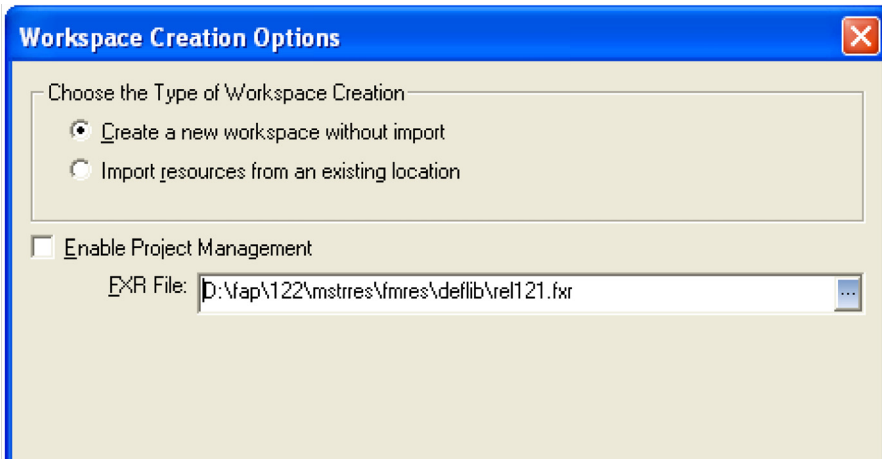


4. Enter a name for your workspace
5. Change the location as desired

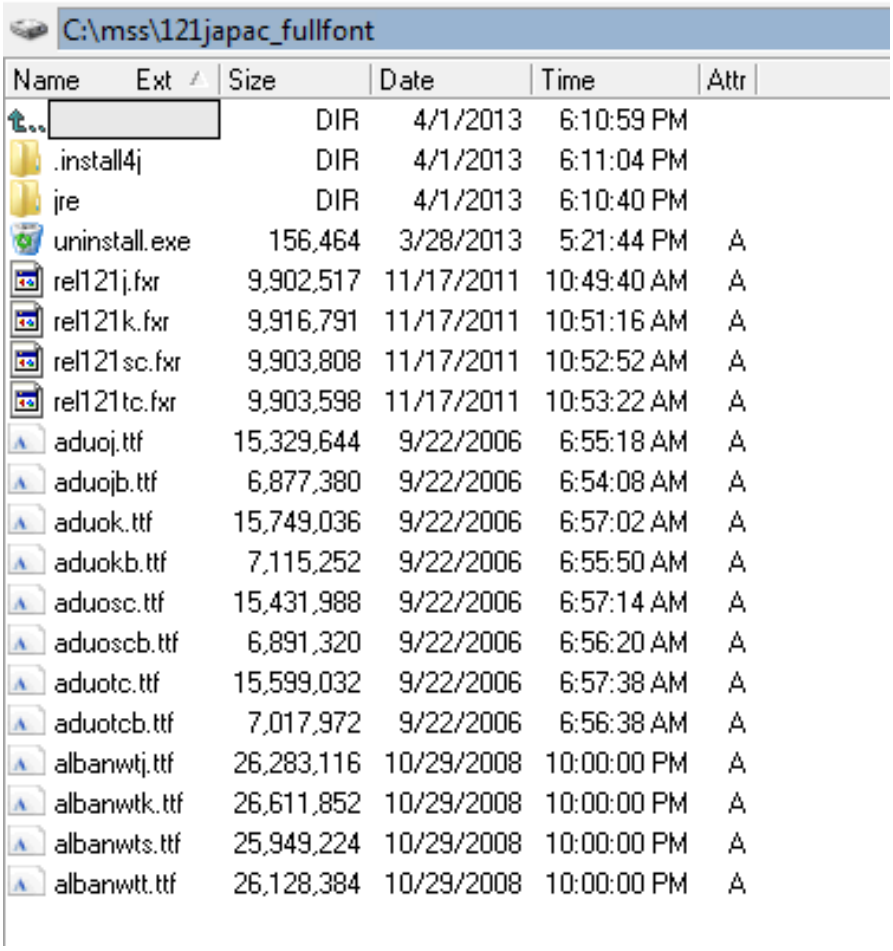


6. Click Next
7. Select Create a new workspace without import
8. FXR file – the FXR file defaults to rel121.fxr located in the FMRES\deflib directory. FMRES is automatically installed when you install Documaker Studio.

9. You should not delete FMRES. FMRES contains resources used by Documaker Studio such as FXR and font files.



10. Click on the ellipses in the FXR file option
11. Open dialog displays
12. Browse to where JAPAC fonts were installed



13. Select rel121j.fxr

Alternatively, you can license TrueType fonts from other vendors, as well as using the fonts supplied with the system. You can license fonts that support a only few languages or you can license fonts that support the entire Unicode 3.0 character set.

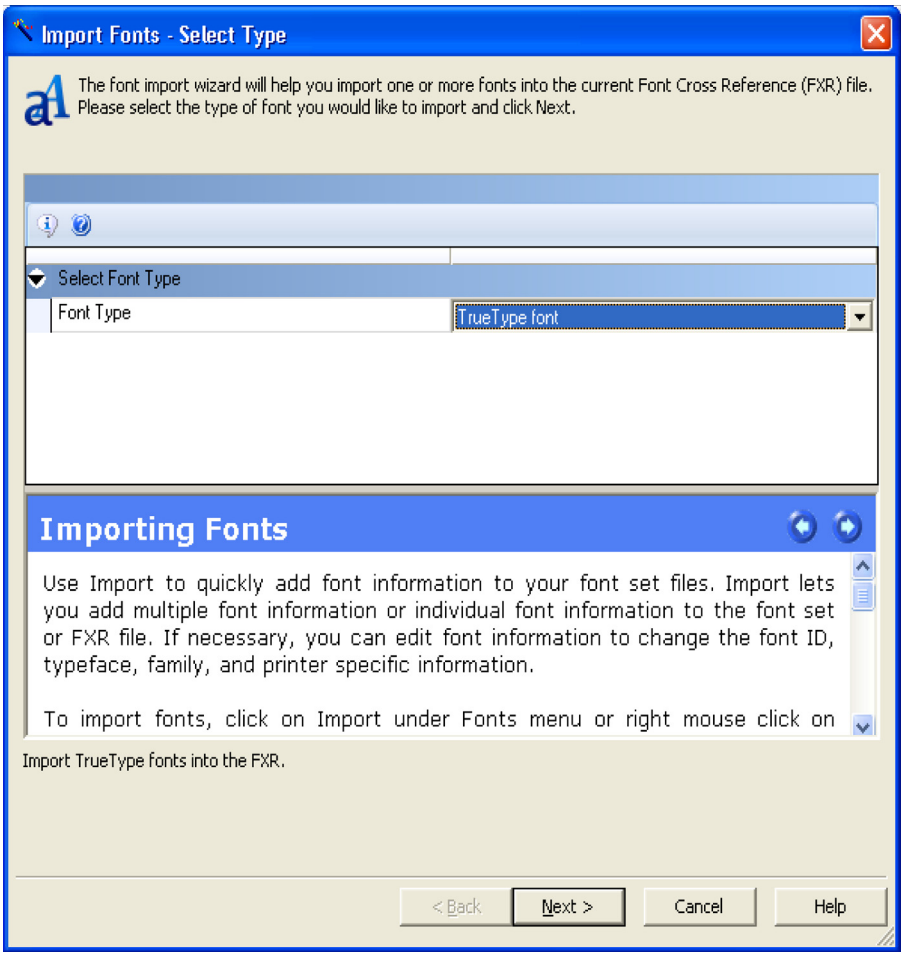
There are several legal issues to be aware of before converting fonts from other vendors. Font vendors generally copyright the fonts they create. You can legally convert fonts only if the font vendor grants permission.

The converted font is bound by the same copyright restrictions that apply to the original font. For example, if your license does not permit you to use the font on more than one computer at a time, then you are not permitted to use the converted font on more than one computer at a time. In addition, it may be a copyright violation to copy converted fonts to other platforms running on the same computer.

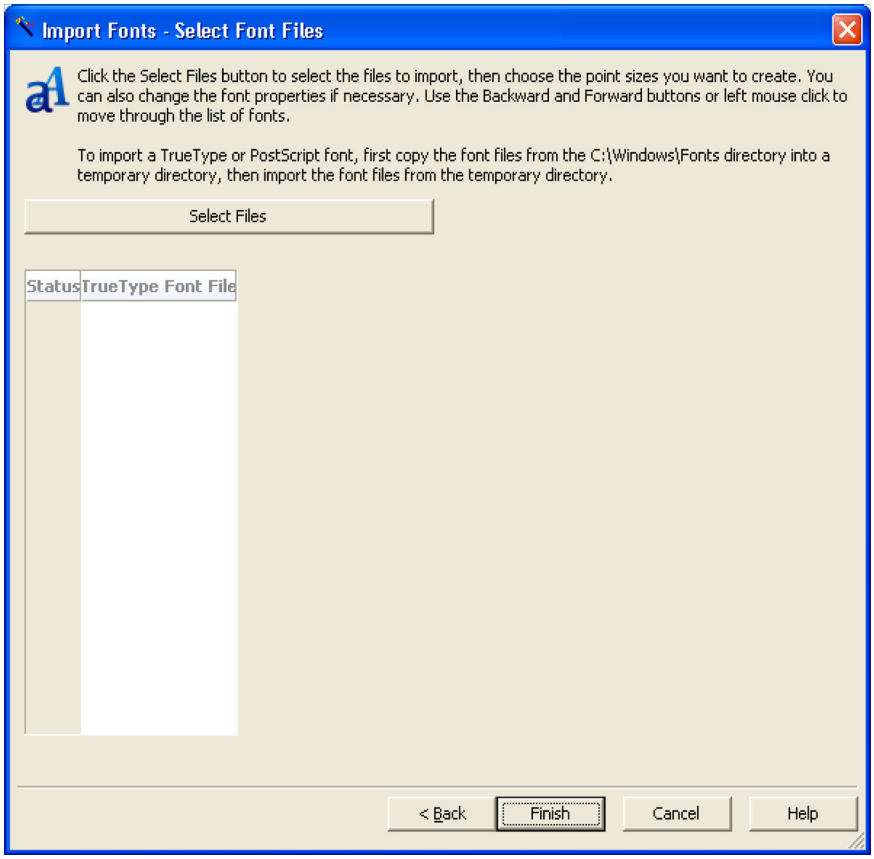
Generally, if you plan to use a TrueType font on more than one machine or plan on permanently installing the font on a printer, your license expressly must give you the right to do so.

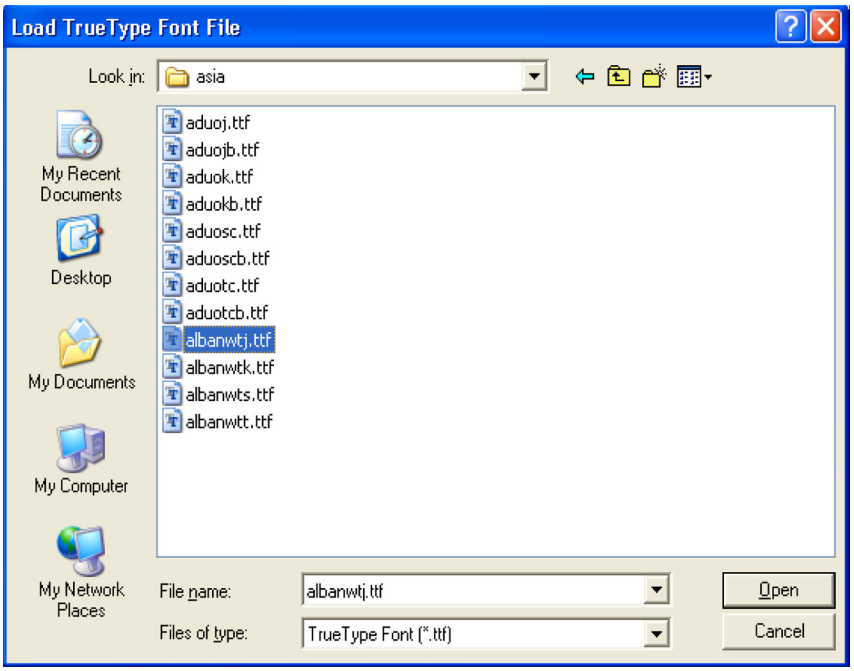
If you want to add selected Unicode fonts to your current font set, then you can add Unicode-enabled fonts to a font cross-reference file via Font Manager.

In this example, we will use the Studio Font Manager to create a new FXR file that will contain Unicode-enabled fonts. Note that you need access rights to Font Manager in order to create or modify an FXR. Right click on the Font Manager and select the New option to create a font cross-reference file. The Import Fonts wizard will begin as shown below. Select TrueType font as the Font Type to import and then click the Next button.



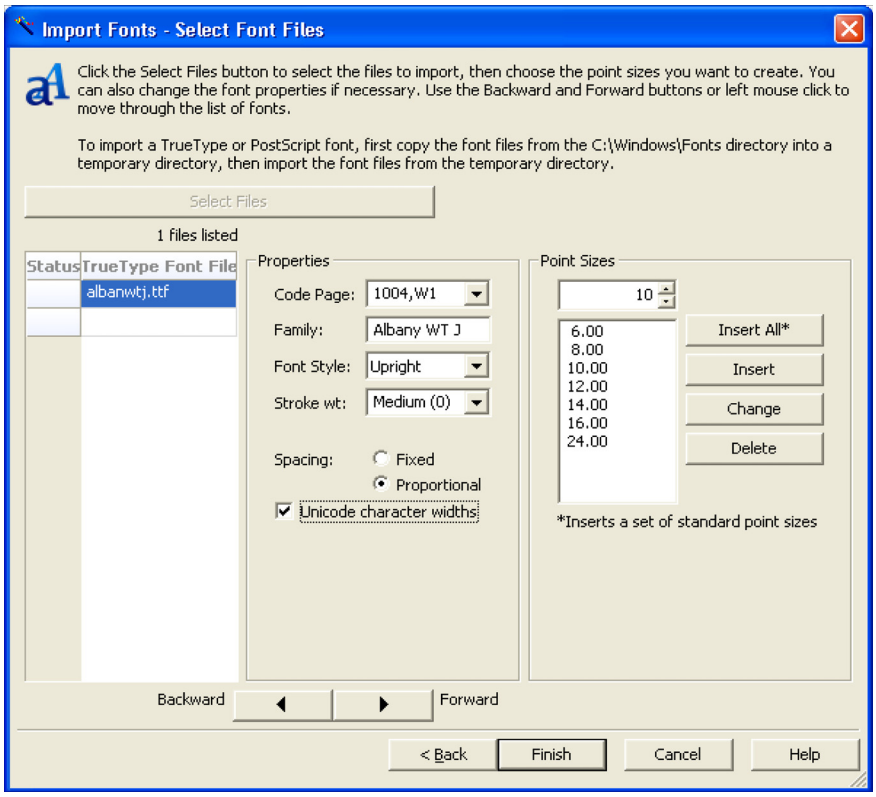
Click on the Select Files button to select the TrueType font(s) to import.



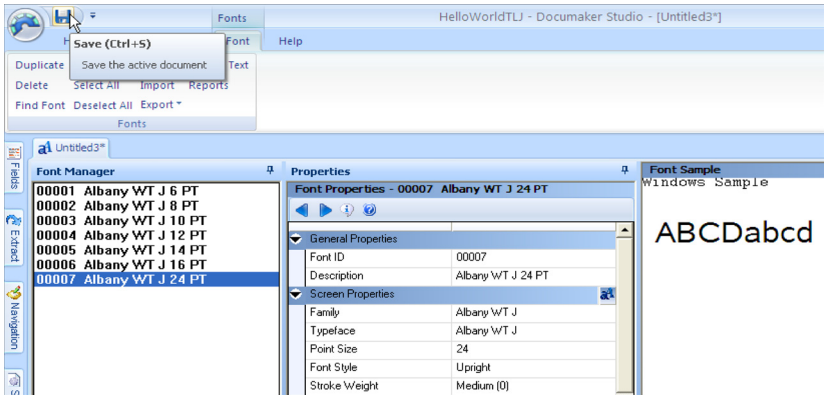


To enable Unicode support for this TrueType font, you must click the Unicode

Character Widths field on the Import window. After you have added all of the point sizes you want to use for this TrueType font, click Finish.



After the system has inserted the point sizes you specified for this TrueType font into the font cross-reference, you will see the main font list window again. Click on the disk icon or select Save from the main menu to save your font cross-reference file.



After you have saved your FXR, you'd want to define it in the Application Definition file (*.BDF) in the workspace.

Installing Fonts in Windows

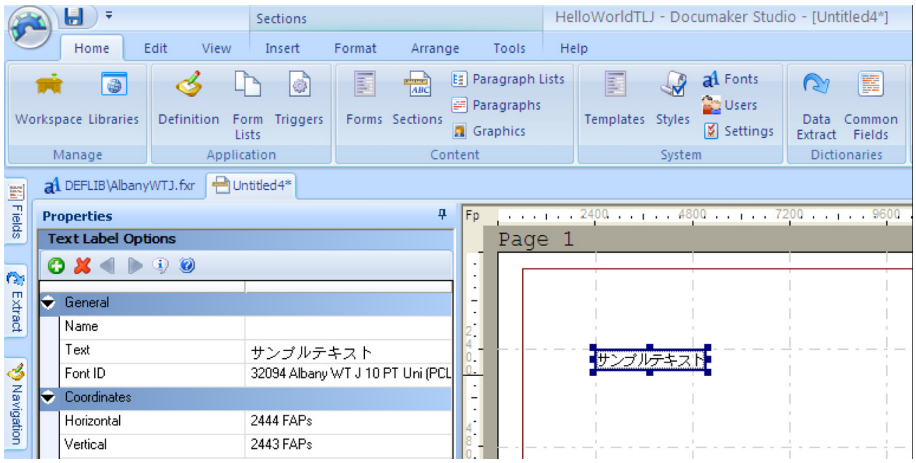
Whatever font set is selected, you must install the TrueType font into Windows before you can begin using the TrueType font to design forms. You can install fonts into Windows using the Fonts folder (located in the Control Panel). After opening the Fonts folder, select the File, Install New Font option. The Add Fonts window appears and asks for the drive and directory in which the new TrueType font files are located. When you finish selecting the fonts you want to install, click Ok to install them.

CREATING FORMS

ADDING UNICODE TEXT

After you have created a font cross-reference file that contains Unicode-enabled fonts, specify that font cross-reference in your master resource library . This should be defined in the BDF (requires access rights to BDF). Then, define it in the XRF setting in config group in INI.

Make sure the FXR has been copied to the \deflib directory in the workspace, then you are ready to begin creating forms with Unicode characters.



In Documaker Studio, insert a new text label. The Text Label Properties window lets you enter Unicode characters.

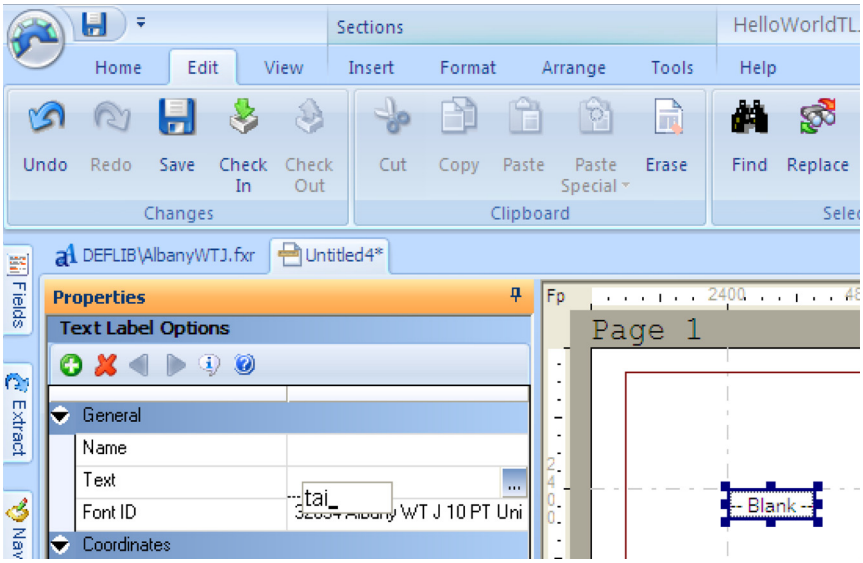
If the font you have selected for this text label is Unicode-enabled then you will see the letters *Uni* in the font description after the point size.

To enter Unicode text into the Text field, select the input method editor (IME) you want to use by clicking the EN icon on the task bar. After you have selected the IME to use, you can begin typing characters for that language. As you accept characters from the IME, they will appear in the Text Label Properties window.

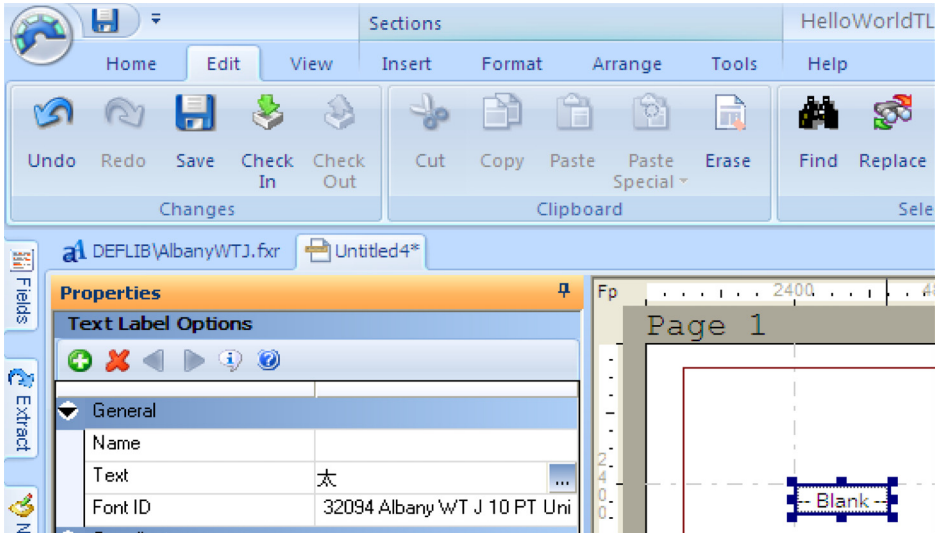
The mechanics of each input method editor (IME) are different so you will need to learn the specifics of the IME you will use. Some input method editors let you type the letters that represent the pronunciation of the word you want to enter. You can also paste the desired text into the text label.

For example, the word *sun* is pronounced as *tai yang* in Chinese and is displayed as shown here:

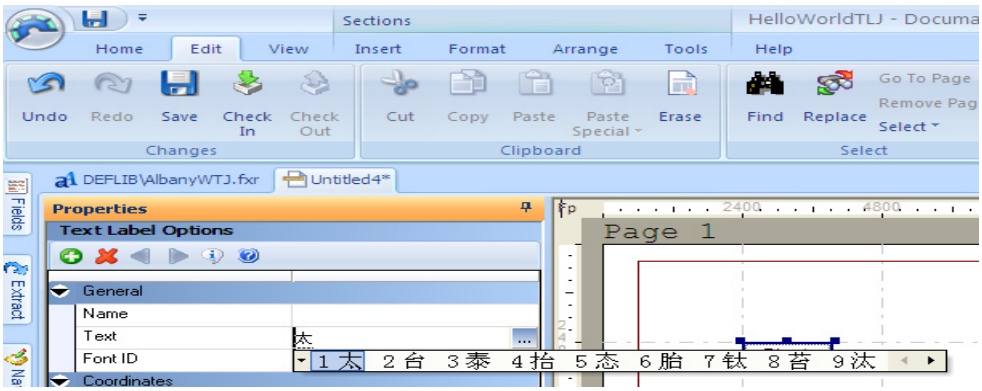
After enabling a Chinese IME, you would begin by typing the letters *tai*. The composition window might appear as follows:



When you press the spacebar, the letters in composition window disappear and the first character appears in the Text field.

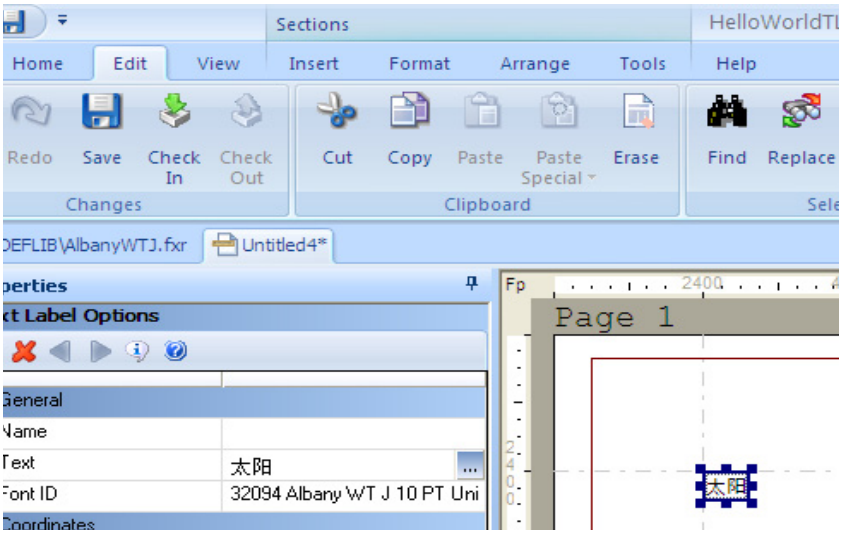


The character displayed is just the first choice of a list of candidate characters that sound similar to *tai*. Some IMEs show you this list of candidate characters immediately and some require you to mouse click on the initial character.

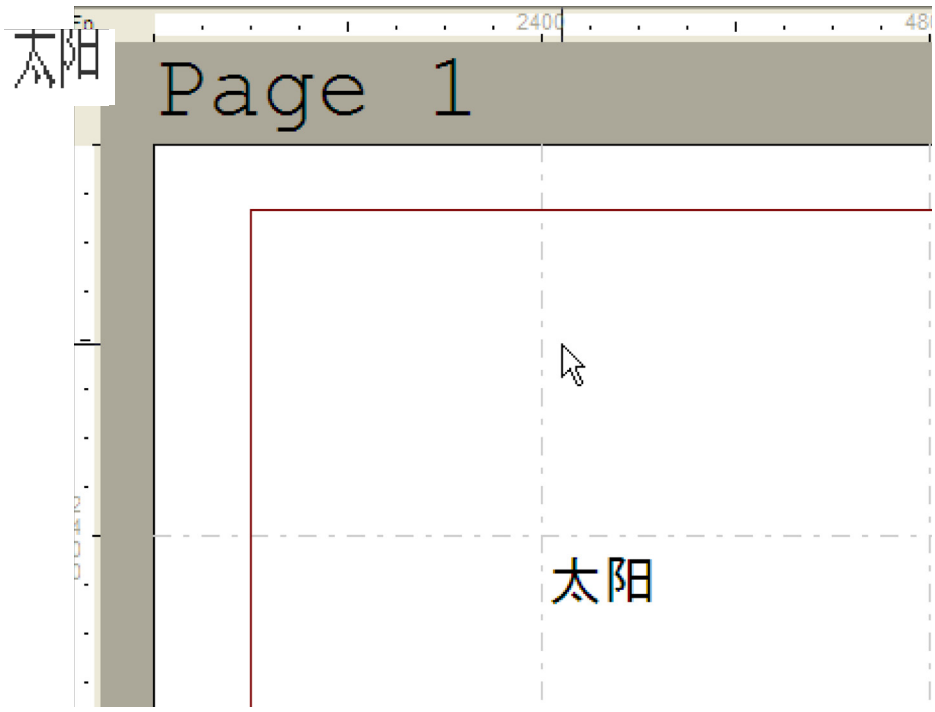


Once you have selected the character you want to use, press the Spacebar to finalize the selection. The display of the character and cursor will change slightly to indicate that the IME has completed the character. At this point, you would begin typing *yang* and selecting as the final character

.



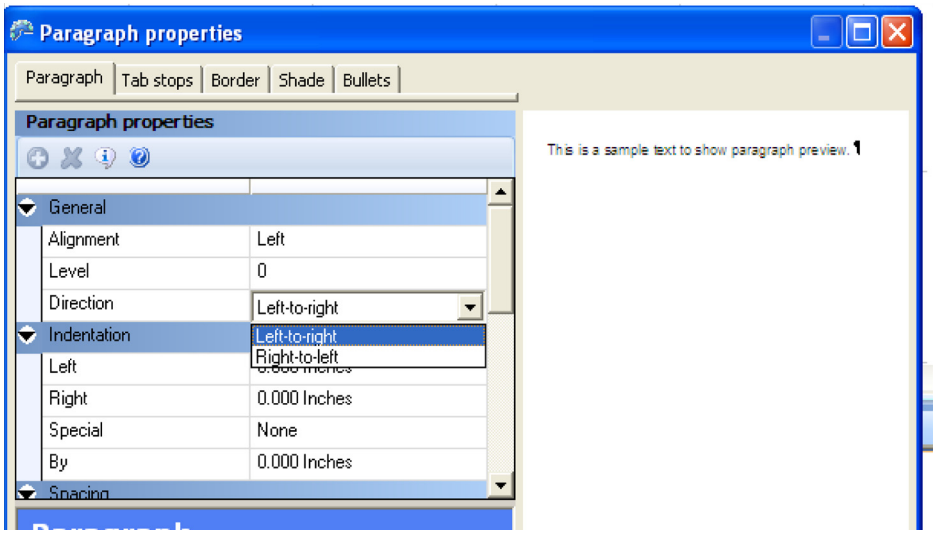
After you finish typing the characters for the text label, click Ok and the characters appear on the form.



Similarly, you can create text areas with characters for the IME you have selected. As you accept characters from the IME, they appear in the text area.

When creating variable fields, you must select a font that is Unicode-enabled. When you have finished composing your form, it might look something like this:

Text Area : After inserting a text area, select Format, Paragraph to access paragraph properties.



There are 2 direction options –Left to right and Right to Left.

Use Right to Left for languages such as Arabic.

Documaker uses resources you create using Studio to process information and forms to generate documents. This processing includes merging external data onto forms, processing data according to rules you set up, creating print-ready files, archiving data and forms, and, if applicable, sending incomplete forms to Work In Progress (WIP) for completion by a user.

USING VARIABLE FIELDS

To enable a variable field for Unicode data, you must use a font ID that is Unicode-enabled. In this example, a field called INSUREDNAME has been created and it uses a font ID of 24010. You can tell that this font ID is Unicode-enabled because of the characters, *Uni*, displayed after the point size in the font description.

MAPPING DATA

Encoding standards

Here is a description of the encoding standards supported by the Documaker XML parser.

Use	When the XML document contains
US-ASCII	Only ASCII characters (code points 32-126). ASCII characters are the type-able characters found on a U.S. English keyboard.
ISO-8859-1	A subset of the characters in the Windows code page 1252 (ASCII characters at code points 32-126, plus additional characters at code points 160-255).
UTF-8	Any character supported by Unicode. Unicode 3.0 supports over 49,000 characters.

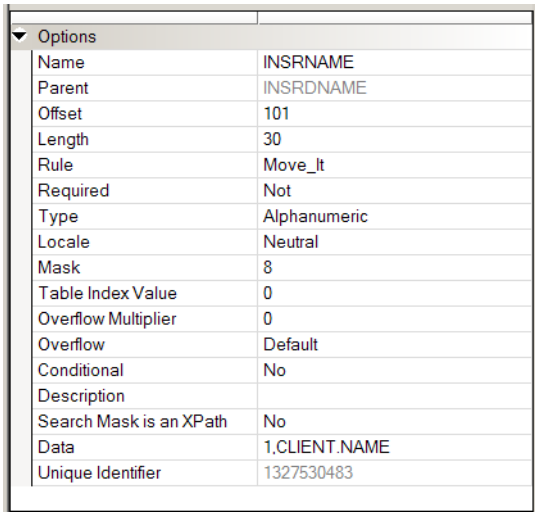
Note UTF-8 (Unicode Transformation Format, 8-bit encoding form) is a format for writing Unicode data in text files (which are normally processed sequentially, one byte at a time). UTF-8 data can be written into an extract file and subsequently used by Documaker. If no encoding is specified, then an encoding of ISO-8859-1 is assumed.

Once you select a font ID that is Unicode-enabled for a variable field, you can use the AnyToAny rule to have the system format the data when it maps it to a field. This rule uses the XDD (Extract Data Dictionary) format as the source format and whatever you specify as the target field format for the output.

Alternatively, you can use the Move_It rule to map the data into this field. The Move_It rule can move UTF-8 formatted data into a Unicode-enabled variable field.

Note When defining the length of text that you want to pull from the extract file, set the value to 3 times the number of characters to be displayed when the text will be either Japanese or Simplified Chinese. If the text will be a CTL language - Hindi, Urdu, Telugu for example, set the length to 5 times the number of characters to be displayed.

Each field using the Move_It rule to map UTF-8 formatting data into a Unicode enabled variable field must have a mask of "8" to indicate that the extract data for this field is stored in the UTF-8 format. Here is how you specify the 8 mask (UTF-8) for a variable field in Studio:



Options	
Name	INSRNAME
Parent	INSRDNAME
Offset	101
Length	30
Rule	Move_It
Required	Not
Type	Alphanumeric
Locale	Neutral
Mask	8
Table Index Value	0
Overflow Multiplier	0
Overflow	Default
Conditional	No
Description	
Search Mask is an XPath	No
Data	1,CLIENT.NAME
Unique Identifier	1327530483

Note For more information about these rules, see the Rules Reference.

In the preceding dialog, the INSRNAME record has been assigned the Move_It rule. It has a mask of 8 indicating that the extract data is in UTF-8 format.

Now look at the extract file data to make sure this is correct. Note in the window above, the record for INSRNAME is identified by the text, *CLIENT.NAME* starting in column 1. The actual data to be used for INSRNAME is stored at Offset 101 for a length of 30. Here is an example of this record in an extract file:

CLIENT.NAME :å–¬æ²»å¸ƒæ®Š

The colon (:) indicates column 100, the UTF-8 data follows the colon at column 101.

The UTF-8 data (å–¬æ²»å¸ƒæ®Š) is not very readable in a regular text editor.

One way to view this data is through the Internet Explorer. By loading the extract file in the browser, the example record in the extract file would appear as follows: CLIENT.NAME :å–¬æ²»å¸ƒæ®Š

In other words, the UTF-8 data (å–¬æ²»å¸ƒæ®Š) can be interpreted as the following characters : å–¬æ²»å¸ƒæ®Š

In many cases, the browser will automatically detect the UTF-8 data and display the UTF-8 data appropriately. If it does not, choose View, Encoding and select the Unicode (UTF-8) option.

Keep in mind, it is expected that the extract file that will contain the Unicode data is provided in UTF-8 encoding. While Documaker supports the following encodings: UTF-8, ISO-8859-1, and US-ASCII; only those XML files encoded in UTF-8 can be used for Unicode data processing. Here is an example of an XML header that specifies UTF-8 encoding:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

When Documaker gets data from the extract file using either the ANY2ANY or the MOVE_IT rule, it retrieves the number of characters as indicated in the source field's length property. If the data to be mapped is Japanese or Simplified Chinese, set the length to a value three times the number of characters you want to retrieve from the extract file. If the data to be mapped is a Complex Text Layout (CTL) language such as Hindi, Hebrew and Telugu, set the length to a value five times the number of characters you want to retrieve from the extract file.

If your library resources are maintained with Image Editor instead of Documaker Studio, be sure to set the LoadCordFAP INI option to Yes in your runtime environment when using Unicode variable fields.

Viewing UTF-8 Data

One way to view this data is through Internet Explorer. By loading the extract file in the browser, the example record in the extract file would appear as follows:

CLIENTNAME:

In other words, the UTF-8 data (å–¬æ²»å¸ƒæ®Š) can be interpreted as the following characters:

In many cases, the browser will automatically detect the UTF-8 data and display it appropriately. If it does not, choose View, Encoding and select the Unicode (UTF-8) option.

PRINTING UNICODE DATA

To print Unicode data, you must have printer fonts and print drivers that support Unicode. Documaker has seven print drivers that support Unicode: GDI, PCL6, RTF, HTML, Bitmap, MPM and PDF (via TrueType fonts).

Note For information on using these print drivers, see the Printers Reference.

GDI print driver

The GDI print driver uses a Windows print driver for printing. There is nothing special you need to do to print Unicode using the GDI driver. Since our PostScript driver does not support Unicode yet and our PCL5 driver will not support Unicode, you can use the GDI driver to print to these types of printers.

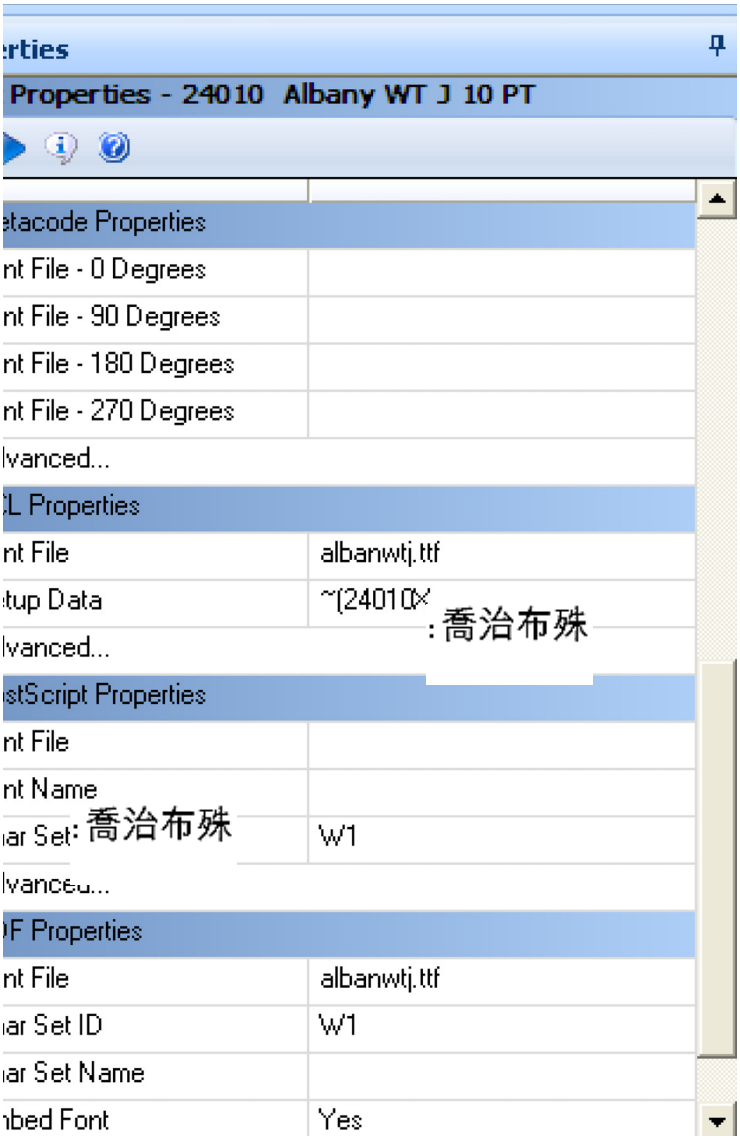
PCL6 print driver

Documaker includes both the original PCL printer driver, PCLLIB, which supports PCL 5, 5c, and 5e and a PCL print driver, PXLLIB, which supports PCL 6.

PCL 6 is a newer version of the PCL language, not a superset of PCL 5. It is a completely different page description language (PDL). PCL 6 is a stack-based protocol (similar to PostScript) composed of attributes and operators that let you define paths, clip paths, pens, brushes, fonts, raster patterns, and so on. PCL 6 also supports 16-bit character codes, which makes it a better choice for supporting Unicode than PCL 5.

Oracle Insurance's PCL6 print driver will support Unicode data, the PCL5 driver will not.

Both the PCL5 and PCL6 print drivers use the PCL printer section of the FXR file. The PCL6 print driver supports PCL5 bitmap fonts and TrueType fonts. You can use PCL5 bitmap fonts for printing ASCII data (SBCS). However, to printing Unicode data, you must use TrueType fonts. To specify a TrueType font in the PCL printer section of the FXR, enter the name of the TrueType font in the Font File field as shown below:



The TrueType font must reside in the FontLib directory as specified in your master resource library (MRL).

You must define the necessary printer options for the GenPrint and other Documaker programs to produce PCL 6 output. These options specify PCL 6 output and are located in a PrtType:xxx control group, such as PrtType:PXL. Many of the PCL 5 and PCL 6 printer INI settings are the same.

The following INI options are different for PCL 6:

```
Module          = PXLW32
PrintFunc       = PXLPrint
DownloadFonts   = Yes (internal font selection is not supported)
```

These INI options are not supported for PCL 6:

- SendOverlays
- OverlayPath

- OverlayExt
- FitToWidth
- AdjLeftMargin
- StapleBin
- PJLCommentScript
- PJLCommentOn
- OutputBin
- HighlightColor
- Tray# (where # is 1,2,3, and so on)

Keep in mind...

- Because the PCL 6 driver supports PCL bitmap fonts, you can use existing MRLs. Just remember to change your INI options as discussed above.
- The PCL 5 driver does not support TrueType fonts.
- When printing with a TrueType font using the PCL 6 driver, only the characters used on the form are downloaded into the print stream. This reduces the size of print stream files, particularly if the TrueType font includes support for Asian languages.
- In comparison to the PCL 5 printer driver, the PCL 6 driver has these limitations:
  - No overlay support
  - No support for a separate downloadable font file which contains multiple PCL fonts
  - No internal printer font support
  - Less paper tray support, no INI options to specify which PCL commands to use
  - No INI options to specify PCL commands to output bin or staple bin
  - No highlight color support
  - No comment script support

PDF print driver

The PDF print driver supports Unicode data. Adobe Acrobat supports Unicode in version 4.0 and higher. We recommend you use version 5.0 or higher.

To create a PDF file containing Unicode data, you must specify a TrueType font to be embedded (downloaded) into the PDF file. You cannot use the internal Acrobat fonts or embed PostScript fonts when printing Unicode data; you must embed a TrueType font. The process for embedding a TrueType font into a PDF file is the same as it was before.

Note You may not be able to view archived data containing Complex Text Layout languages, if the archive is retrieved using Documaker releases prior to version 12.2 and higher.

The Unicode data in the NA file will be written out as UTF-8 data. To look at the UTF-8 data as Unicode characters, see *Viewing UTF-8 Data* on page 30.

For Documaker Studio, you could run Test Manager.

WRITING CUSTOM RULES WITH UTF-8 ENCODED XML

Memory representation

Use this information if you have to write Documaker or IDS code and you need to know how data is presented in memory on the XML tree.

- The data on XML tree is always stored in UTF-8 format. The Documaker rules or any other code has to do a conversion into the appropriate format, either Unicode or the ANSI character set, based on the destination format you need.
- The code that uses Oracle libraries to deal with the Unicode characters should be passing in the flag to create XML text (DXMFLAG_WIDECHAR). If this flag is passed in, the XML library converts the Unicode string into a UTF-8 encoded string.
- The code that needs to create XML text from already UTF-8 encoded string should use the DXMFLAG_UTF8 flag to create text. When this flag is used, no conversion occurs.
- The code that has to create text from an ANSI string should not set any flags. The CreateText API takes care of converting strings into a UTF-8 representation in memory.
- When getting text from XML library, the calling application should do its own conversion from UTF-8 into the appropriate format, either ANSI or Unicode.

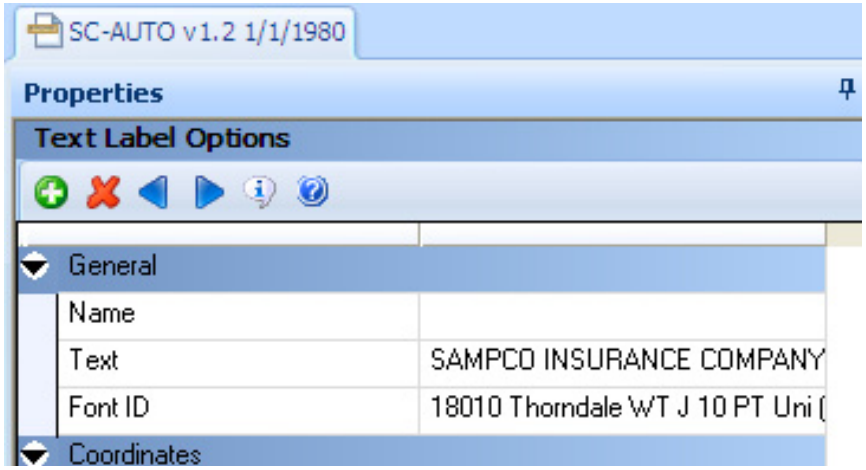
  Oracle may add an API or a parameter to the existing API to help with this conversion, but for now you must handle the conversion. There are a couple of APIs in UTLLIB, UTLUtf8ToAnsi and Utf8ToUtf16, that convert a UTF-8 string into ANSI characters and Unicode characters.

Frequently Asked Questions

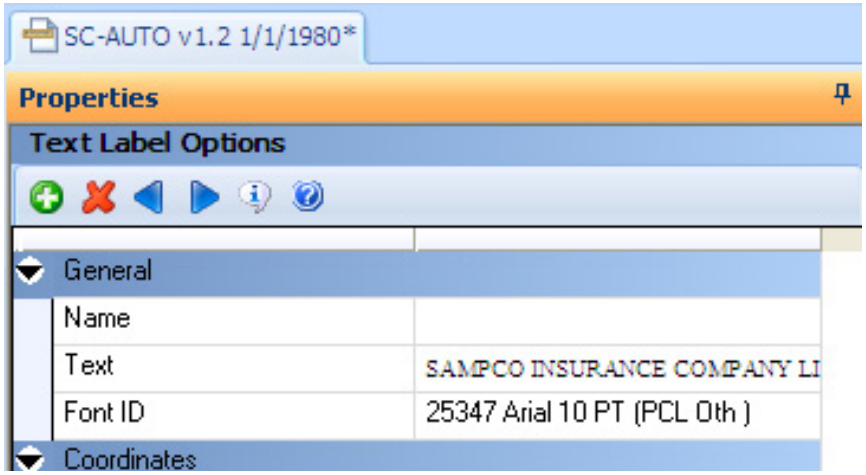
Why does text appear jumbled or change to question marks after you change the font ID?

When you create a text label with a Unicode-enabled font ID, the text label contains two-byte Unicode characters, even if the text is English. Font IDs that are not Unicode-enabled do not contain character width information for Unicode characters. Therefore, if you use the Edit, Font Change option in the Image Editor to change the font ID to a font ID that is not Unicode-enabled, the Unicode characters will have no width and will appear jumbled on top of one another.

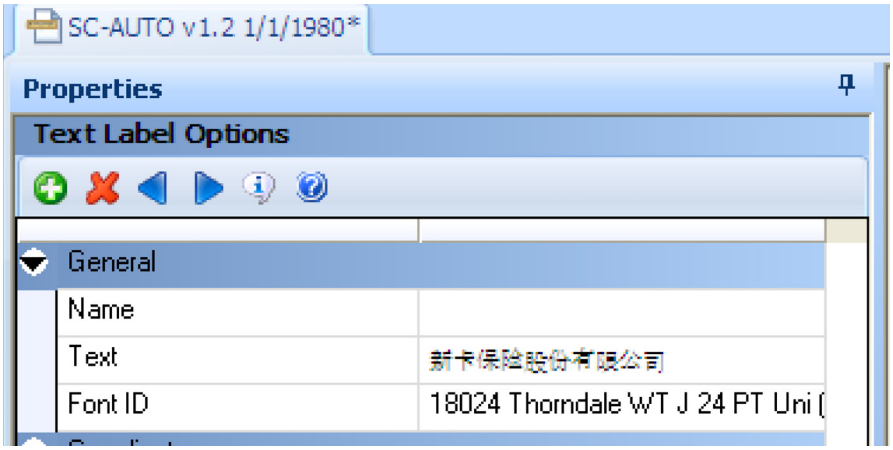
If you change the font ID from within the Text Label window to a font ID that is not Unicode-enabled, the system tries to convert the text from Unicode to normal ANSI characters. For example, the following text label starts out as Unicode text because it uses a Unicode enabled font ID:



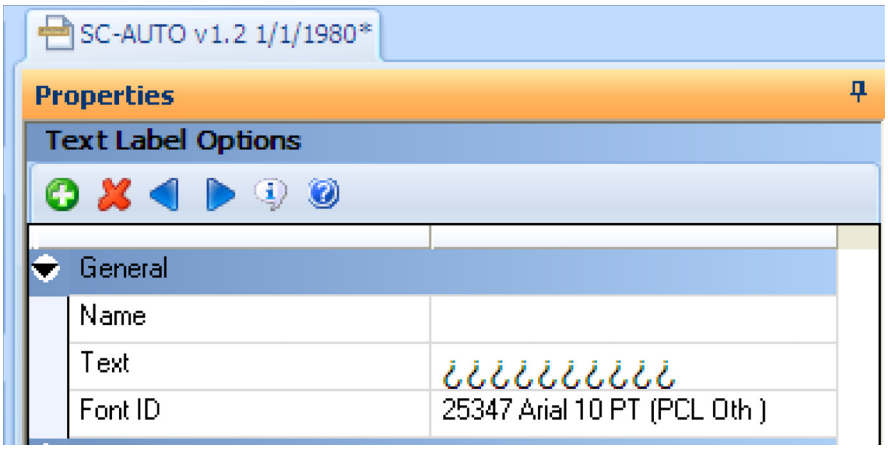
Now if you change the font ID to a normal (non-Unicode) font ID, the text is changed to ANSI characters (notice that the characters *Uni* do not appear in the font description:



However, if you tried this with a text label containing Chinese Unicode text, the text would lost when switching to a non-Unicode font ID:



Watch what happens when you change the font ID to a normal font ID:



At this point, the Chinese text is lost unless you close the window. The system tried to convert the Unicode Chinese text to ANSI characters but of course, it could not. At this point, the text label actually consists of ANSI question marks as:

¿¿¿¿¿¿¿¿¿¿¿

Changing the font ID back to a Unicode-enabled font ID just results in Unicode question marks.