UNIFACE Dev Conf

Uniface Unicode and UTF-8

Arjen van Vliet

Solution Consultant

UNIFACE Dev Conf

Developing International Applications

Topics

- Types of International Applications
- Character Sets
- Uniface Support for Unicode
- Installable Character Sets
- Developing a Language-Specific Application
- Language and Locale
- Time Zones
- Support for Unicode in database connectors
- Migrating to Unicode

Types of International Applications

- Multilingual applications
  - A multilingual Uniface application that runs in a Unicode environment
  - Can display many different languages and accept inputs from different languages
- Specific language applications
  - National language application that can be built directly in the language’s platform
  - Example: Build a Simplified Chinese application in a Simplified Chinese Microsoft Windows environment, Arabic in Arabic Windows etc.
- Generic language applications
  - An application can be deployed in multiple language environments.
  - Using Uniface’s language variation technique.
  - Deployed in a specific language environment, its user interface appears in that language. Example: Deployed in Japanese Windows, appears in Japanese

Character Sets

- Uniface supports multiple character sets for data entry and display in various languages.

  Two categories:

  1. Installable character sets
     - During Uniface installation, you select a character set in the Character set selection
  2. Unicode
     - Uniface's internal character set (since version 9)

Uniface Support for Unicode [1]

- Uniface supports Unicode in:
  - User Interface
  - Database storage and retrieval
  - Sorting (based on current ‘locale’)
  - Case conversion
  - Data exchange
- On GUI platforms, Unicode characters :
  - appear in the user interface
  - can be stored in string fields
- On any (other) platform, Unicode characters:
  - can be stored in and retrieved from any Unicode database
  - can be exchanged with other Unicode based components, like Java and Unicode files

Uniface Support for Unicode [2]

- Uniface uses Unicode UTF-8 encoding as its internal character set
  - Uniface does **not** provide the glyphs to display the characters
  - Uniface does **not** supply the tools to enter the characters.

…These are handled by Microsoft Windows.

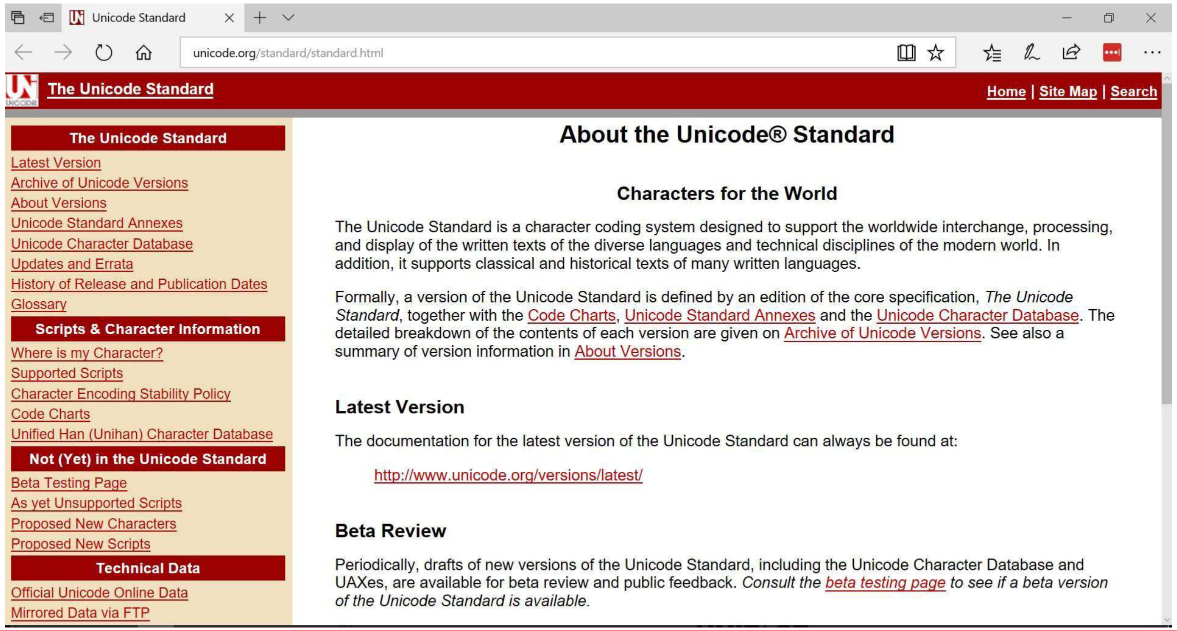
- For data entry and display:
  - Microsoft Windows must be configured to display and input the desired Unicode characters
  - Uniface must be configured to use a Unicode Font for displaying Unicode characters

Uniface Support for Unicode [4]

- To develop and deploy a Unicode application in Uniface, you need to understand the following:
  - The correct data type and packing code for storing Unicode characters
  - Where Unicode characters are allowed and where not
  - Proc functions that can handle Unicode characters

Unicode

- Unicode is an industry standard designed to allow text and symbols from all languages to be consistently represented and manipulated by computers.
- The Unicode Standard: http://unicode.org/standard/standard.html



The Unicode Standard

Latest Version
Archive of Unicode Versions
About Versions
Unicode Standard Annexes
Unicode Character Database
Updates and Errata
History of Release and Publication Dates
Glossary

Scripts & Character Information

Where is my Character?
Supported Scripts
Character Encoding Stability Policy
Code Charts
Unified Han (Unihan) Character Database

Not (Yet) in the Unicode Standard

Beta Testing Page
As yet Unsupported Scripts
Proposed New Characters
Proposed New Scripts

Technical Data

Official Unicode Online Data
Mirrored Data via FTP

About the Unicode® Standard

Characters for the World

The Unicode Standard is a character coding system designed to support the worldwide interchange, processing, and display of the written texts of the diverse languages and technical disciplines of the modern world. In addition, it supports classical and historical texts of many written languages.

Formally, a version of the Unicode Standard is defined by an edition of the core specification, *The Unicode Standard*, together with the Code Charts, Unicode Standard Annexes and the Unicode Character Database. The detailed breakdown of the contents of each version are given on Archive of Unicode Versions. See also a summary of version information in About Versions.

Latest Version

The documentation for the latest version of the Unicode Standard can always be found at:

http://www.unicode.org/versions/latest/

Beta Review

Periodically, drafts of new versions of the Unicode Standard, including the Unicode Character Database and UAXes, are available for beta review and public feedback. *Consult the beta testing page to see if a beta version of the Unicode Standard is available.*

Unicode terminology

Unicode concept	Description
Basic Multilingual Plane (BMP)	The set of characters whose code points are in the range 0000 to FFFF. The BMP contains characters for almost all modern languages, and a large number of special characters.
Supplementary Multilingual Plane (SMP)	Characters whose code points are greater than FFFF. The SMP is mainly used for historic scripts and for musical and mathematical symbols.
Primary Private Use Area	Code points in the range E000 to F8FF. This area is reserved for Unicode users to define their own characters. In Uniface, the assignment setting $GAIJI is used to designate this area for Japanese Gaiji characters or for private use.
Supplementary Private Use Area-A	Code points in the range F0000 to FFFFD
Supplementary Private Use Area-B	Code points in the range 100000 to 10FFFD This area is reserved for Uniface users
UTF-8	Unicode Transformation Format, 8-bit. A variable-length Unicode character encoding format, that can represent every character in the Unicode character set

Packing Codes for Unicode

- W Packing Code
  - [illegible]
  - [illegible]
  - [illegible]
- U* Packing Code
  - [illegible]
  - [illegible]
  - [illegible]
- C Packing Code [illegible]
  - [illegible]

Configuring for Unicode [1]

- To use Unicode in Uniface development and deployment
  - Unicode character fonts and input methods must be installed
  - Uniface must be correctly configured
- In a Unicode-based GUI platform such as Microsoft Windows
  - not all Unicode characters are enabled; the total number of characters is too great
  - the Unicode characters that you want must be installed if they are not available
- There are many Unicode fonts available from which you can choose to install.
  - Uniface does not know which font you want
  - so you must specify the desired font in Uniface’s .ini file.

Configuring for Unicode [2]

- To configure for Unicode In Microsoft Windows
  1. Copy the Unicode font file (for example, ARIALUNI.TTF) into the Windows Fonts directory
  2. Set the Unicode font to the items displayed on your desktop via Windows' Control Panel
  3. Enable a set of characters and its input method, via Regional Options from the Control Panel (IME)
  4. Use a Unicode-supported database and database connector
  5. Install Uniface. During installation, there is a Character set selection screen
  6. Modify the .ini file:
     ```
     [SCREEN]
     EditFont=Arial Unicode MS,8,regular
     ```
- Procedure varies with the versions of Windows, check Windows help

Unicode Compatibility Issues

- Earlier versions of Uniface (before v9) used ‘meta character set’ internally
- Uniface 9 (and 10) use Unicode internally
- For compatibility reasons, the meta character set is still usable
- Assignment settings are available for compatibility
  - Missing Meta Characters in Unicode
  - Incorrectly Returned Mappings

Assignment Settings for Unicode Compatibility

Section/Settings	Description
`[META_CHARSET]`	Specify your own mappings between the meta character set and Unicode.
`$EXTENDED_SYNTAX`	For the extended characters (~& or ~@) in the field syntax definition. • If database only supports Unicode BMP, set `$EXTENDED_SYNTAX=BMP`. • If application contains meta characters that are missing in Unicode, set `$EXTENDED_SYNTAX=v8`.
`$FULL_SYNTAX`	For the `FUL` shorthand code for field syntax. • If database only supports Unicode BMP, set `$FULL_SYNTAX=BMP`. • If application contains meta characters that are missing in Unicode, set `$FULL_SYNTAX=v8`.

Unicode in Uniface Applications

- In Uniface applications, Unicode characters can be used in:
  - String data fields, variables, and parameters
  - String literals in the user interface, such as labels, menu items, and form titles
  - String parameters passed between the application and other components; for example, file I/O, or calling a WebService
- Uniface provides:
  - The W packing code for string fields to contain Unicode characters
  - The proc function `$string` to generate Unicode characters from their codes

Unicode in Uniface Development Environment

- The first 128 Unicode characters (the ASCII characters) can be used everywhere in Uniface as long as the name conventions are satisfied
  - Example: Control characters cannot be part of the object's names

- The remaining Unicode characters:
  - are allowed in widgets, Unifields, initial values, and menu items
  - are not allowed as object names; for example, form, template, entity names
  - are allowed when passing string parameters to a non-Uniface component, if the component is Unicode based such as Java.

- Non-ASCII characters in an assignment file, need to be saved in Unicode (UTF-16 or UTF-8) encoding.

Packing Codes for Unicode

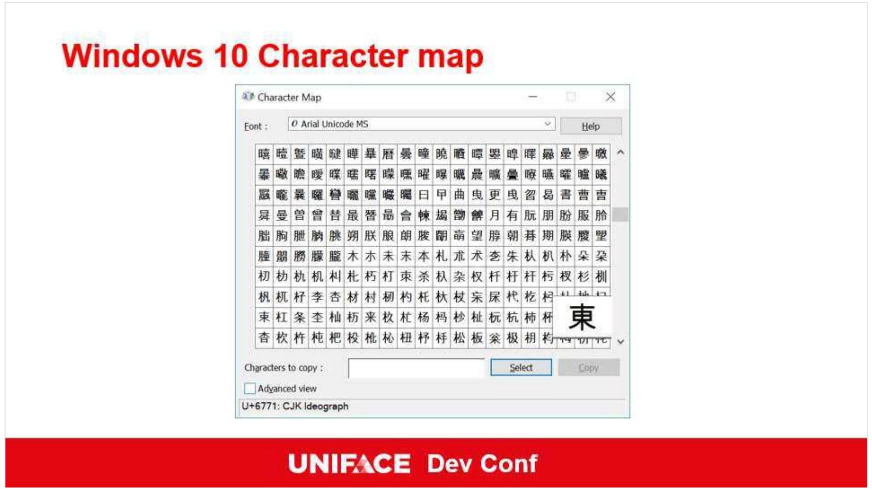
- W Packing Code
  - By default, the W packing code is for Unicode characters
  - A string field that uses the W packing code can contain any Unicode character
  - Some databases support only Unicode BMP (Basic Multilingual Plane), or do not support Unicode at all.
    Use ASN setting: **`$WIDE_CHAR_BEHAVIOR {=} Unicode | BMP | Charset`**
- U* Packing Code
  - Unicode data can be stored in fields with a U* packing code
  - …by setting the $META_IN_TRX assignment setting to 0
  - Data is stored in XML format, therefore, Unicode characters are possible
- C Packing Code (**`$DEF_CHARSET`**)
  - If the specified character set is Unicode, the C String fields are also used for Unicode

Entering Unicode [1]

1. Keyboard
   - Western characters can be entered using a standard Western keyboard
2. IME (Input Method Editor)
   - An (external) program for entering complex characters and symbols using a standard Western keyboard
   - Example:
     - Use a Traditional Chinese IME for entering Traditional Chinese characters
     - Use a Japanese IME for Japanese characters
     - Etcetera
   - Initialization settings to configure default behavior for using the IME
     - AutoImeOpen, AutoImeClose, AutoImeGold

Entering Unicode [2]

3. Copy and Paste
   - Example: Copy and paste from Microsoft Windows' Character Map:
   - If Windows is configured for a subset of Unicode characters, a Unicode font that covers all the characters in the subset is installed.
   - Suppose the font is Arial Unicode MS and the desired character is ‘east’ in Traditional Chinese:



Windows 10 Character map

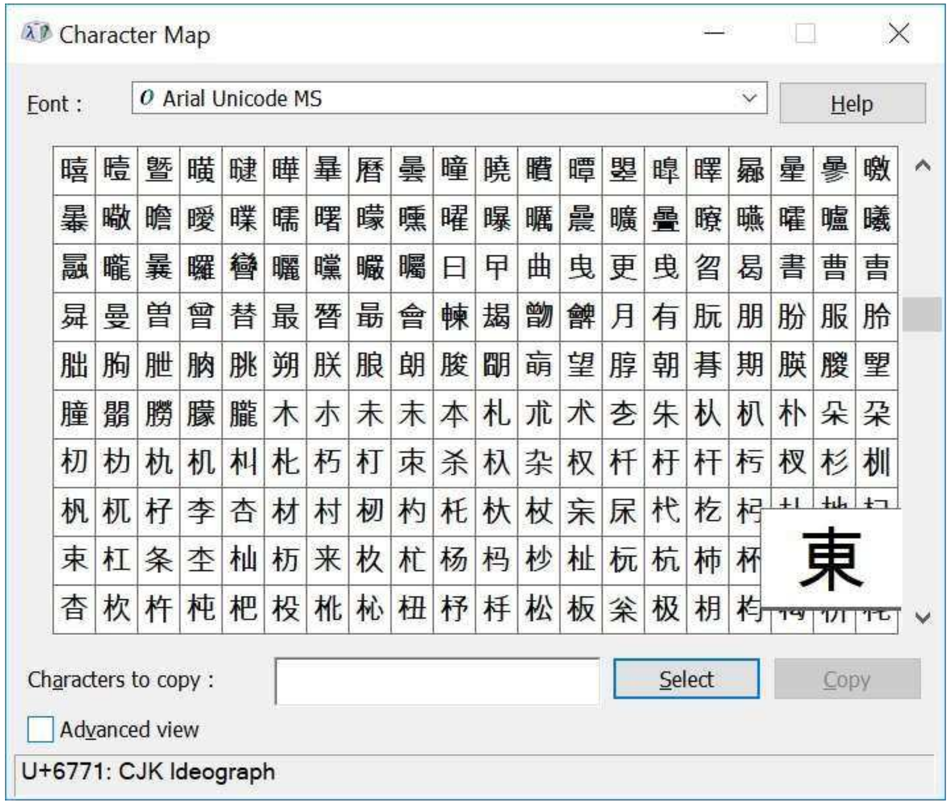
Character Map

Font : Arial Unicode MS

Characters to copy :

U+6771: CJK Ideograph

Windows 10 Character map



Character Map

Font : Arial Unicode MS

Help

暿	曀	曁	曂	曃	曄	曅	曆	曇	曈	曉	曊	曋	曌	曍	曎	曏	曐	曑	曒
曓	曔	曕	曖	曗	曘	曙	曚	曛	曜	曝	曞	曟	曠	曡	曢	曣	曤	曥	曦
曧	曨	曩	曪	曫	曬	曭	曮	曯	曰	曱	曲	曳	更	曵	曶	曷	書	曹	曺
曻	曼	曽	曾	替	最	朁	朂	會	朄	朅	朆	朇	月	有	朊	朋	朌	服	朎
朏	朐	朑	朒	朓	朔	朕	朖	朗	朘	朙	朚	望	朜	朝	朞	期	朠	朡	朢
朣	朤	朥	朦	朧	木	朩	未	末	本	札	朮	术	朰	朱	朲	朳	朴	朵	朶
朷	朸	朹	机	朻	朼	朽	朾	朿	杀	杁	杂	权	杄	杅	杆	杇	杈	杉	杊
杋	杌	杍	李	杏	材	村	杒	杓	杔	杕	杖	杗	杘	杙	杚	杛	[illegible]	[illegible]	[illegible]
束	杠	条	杢	杣	杤	来	杦	杧	杨	杩	杪	杫	杬	杭	杮	杯	[illegible]	[illegible]	[illegible]
杳	杴	杵	杶	杷	杸	杹	杺	杻	杼	杽	松	板	枀	极	枂	枃	[illegible]	[illegible]	[illegible]

東

Characters to copy :

Select

Copy

Advanced view

U+6771: CJK Ideograph

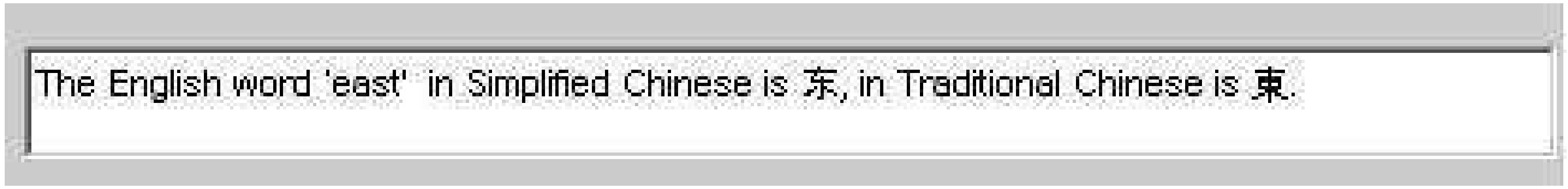
UNIFACE Dev Conf

Entering Unicode [3]

4. Use the **`$string`** Proc Function

- Unicode characters can be generated from their codes using the Proc function **`$string`**
- Example: Chinese character for ‘east’.
    - In traditional Chinese, its code is x6771
    - In Simplified Chinese, its code is x4E1C
    - The data type of field EAST is String with the W packing code. The following Proc assigns the characters to the field EAST

```
EAST = $string("The English word 'east' in Simplified Chinese is &#x4E1C;, %\
in Traditional Chinese is &#x6771;.")
```


The English word 'east' in Simplified Chinese is 东, in Traditional Chinese is 東.

- By using **`$string`**, you can generate the entire set of Unicode characters

Unicode-Related Settings and Proc Instructions

Setting	Description
`$GAIJ`	If the Unicode private area ranged from E000 to F8FF is needed for your application, set $GAIJ to 0
`$META_IN_TRX`	To let U*-String fields hold Unicode characters, set **`$META_IN_TRX=0`**
`$WIDE_CHAR_BEHAVIOR`	If the database only supports Unicode BMP, set **`$WIDE_CHAR_BEHAVIOR=BMP`**
`[META_CHARSET]`	Make your own mappings between Unicode and the meta character set

Proc instruction	Description
`$string`	Generate Unicode characters from their codes
`fileload, lfileload`	Specify the Unicode format in the *CharacterSetName* parameter to load a Unicode file into Uniface
`filedump, lfiledump`	Specify the Unicode format in the *CharacterSetName* parameter to write texts into a Unicode file

Displaying Unicode Characters [1]

- Unicode characters can be displayed only on GUI platforms. The GUI deployment environment needs to have the correct Unicode font to display the Unicode characters.
- For a Windows desktop application, the Unicode font specified in the [screen] section of the .ini file must be installed in the Microsoft Windows.
- If your application contains Unicode characters in the form titles, menu items and tool tips, the correct Unicode font needs to be set for the corresponding items in the Display Properties of the Windows.
- If the writing and reading direction of the language is from right to left such as Arabic and Hebrew, special assignment settings are needed.

Displaying Unicode Characters [2]

- For a web application, the browser should be able to display Unicode. If the characters do not appear correctly, you can check whether or not the encoding of your browser is Unicode.
  - Microsoft Internet Explorer: Menu bar -> View -> Encoding, and select Unicode
- The browser's encoding should be switched automatically to Unicode through the web configuration using the **`DEFAULTENCODING`** setting.
  - In the **`UnifaceInstallationDirectory\uniface\webapps\uniface\WEB-INF`**, modify the file web.xml by inserting the following lines into the WRD servlet's specification:

```
<init-param>
          <param-name>DEFAULTENCODING</param-name>
          <param-value>windows-1251</param-value>
</init-param>
```

Printing Unicode

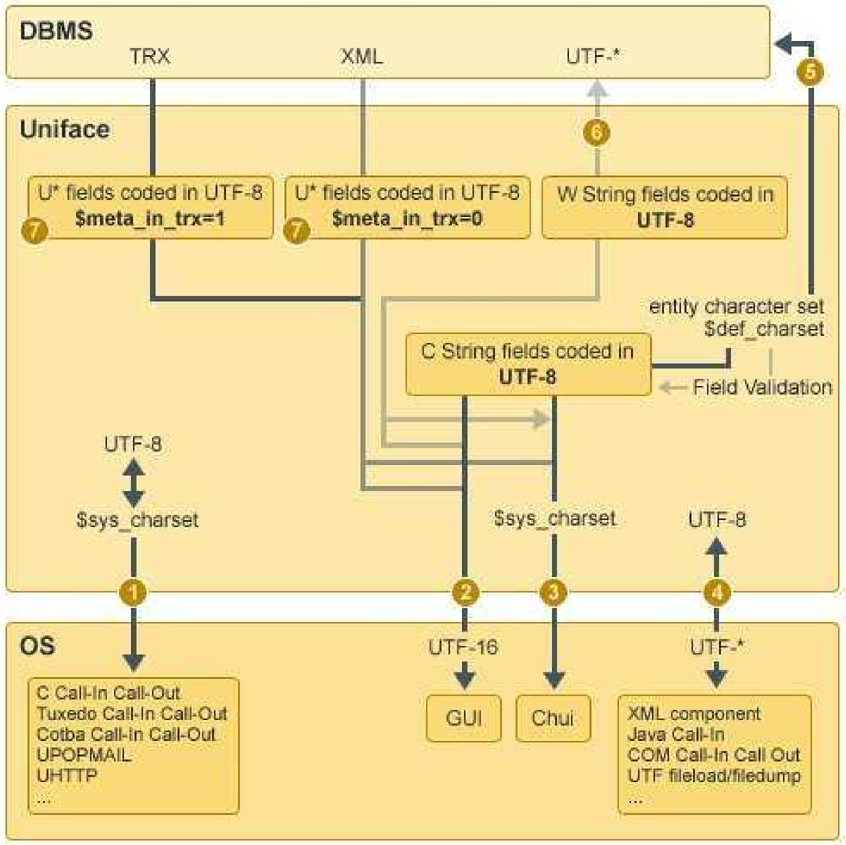
- If a Uniface report or form contains non-ASCII Unicode characters, use enhanced printing to print it.

- Example:
  A Simplified Chinese Uniface version is in a Simplified Chinese Microsoft Windows.
  To print reports and forms that contain complex Unicode characters, define the **Print Job Model** using `P_MSWINX_GB` as the **Device Type**.

Uniface Support for Unicode and Other Char Sets [1]

- A Uniface installation simultaneously supports at least two character sets:
  1. Unicode
  2. The installed character set
- Characters are encoded in Unicode when displayed in a GUI platform or when passed to a Unicode-based component, such as Web services.
- Characters are encoded in the installed character set when displayed in a character-mode user interface (CHUI), or when passed to a non-Unicode-based component, such as C.
- A Uniface installation can simultaneously support many character sets. In addition to Unicode and the installed character set, you can assign **`$DEF_CHARSET`** a character set different from **`$SYS_CHARSET`**
  Normally, you do not do this unless there is a good reason

Uniface Support for Unicode and Other Char Sets [2]



Restrictions on Use of Unicode and Installable Character Sets

Although Uniface supports Unicode and the installable character sets, there are restrictions in using them:

- ASCII characters are allowed anywhere.
- The following tables show where the full Unicode character set (as opposed to the BMP), and the installable character set can be used.

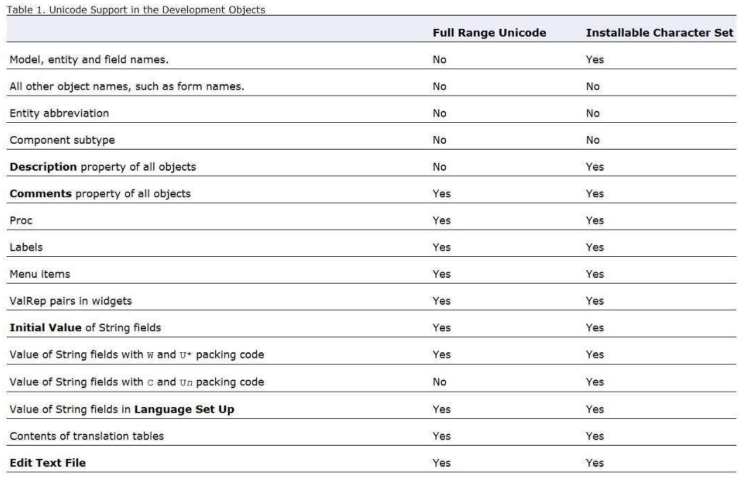


Table 1. Unicode Support in the Development Objects

	Full Range Unicode	Installable Character Set
Model, entity and field names.	No	Yes
All other object names, such as form names.	No	No
Entity abbreviation	No	No
Component subtype	No	No
Description property of all objects	No	Yes
Comments property of all objects	Yes	Yes
Proc	Yes	Yes
Labels	Yes	Yes
Menu items	Yes	Yes
ValRep pairs in widgets	Yes	Yes
Initial Value of String fields	Yes	Yes
Value of String fields with `W` and `U*` packing code	Yes	Yes
Value of String fields with `C` and `Un` packing code	No	Yes
Value of String fields in **Language Set Up**	Yes	Yes
Contents of translation tables	Yes	Yes
Edit Text File	Yes	Yes

Installable Character Sets [1]

- Character sets that conform to one of several characters encoding systems can be installed into Uniface applications.
- Before Unicode was invented, there were many character-encoding systems to provide different character set standards. For example:
  - ISO (International Standard Organization) character sets
  - Code Pages (Microsoft character sets)
  - National standards, such as GB for Simplified Chinese character sets, Shift-JIS for Japanese character sets
- These standards specify both the available characters and the way they are encoded, so the terms ***character set*** and ***character encoding*** are often considered synonymous.
  It is possible to install one of these character sets into a language-specific Uniface application. They are therefore called installable character sets.

Installable Character Sets [2]

- When installing Uniface, you can choose to install a language-appropriate character set such as:
  - Western European
  - Cyrillic
  - Traditional Chinese
  - Etc.
- The actual character set installed depends on the platform, operating system, and language.
- Installable character sets can be either single-byte or double-byte. Single-byte character sets can be left-to-right, right-to-left, or bidirectional

Types of Character Sets [1]

- The installable character sets can be either single-byte or double-byte. Single-byte character sets can be divided into left-to-right and right-to-left.
- Single-Byte and Double-Byte Character Sets
  - Single-byte character set: Each character is represented by 1 byte in the computer.
  - Double-byte character set: Each character is represented by 2 bytes in the computer.
  - A single-byte can have (2^8 = 256) different codes.
  - Two bytes can have (2^{16} = 65536) different codes.
  - A single-byte character set is enough to represent characters in most languages
  - Each Chinese word has a unique Chinese character to represent it. To support Chinese, at least 7000 characters must be supported, so a double-byte character set is needed.

Types of Character Sets [2]

- Left-to-right and Right-to-left Character Sets
  - Left-to-right (LTR) and right-to-left (RTL) indicate the reading and writing directions of a language.
  - If a language is LTR, the type of the character set is also LTR.
  - However, a character set used for RTL languages usually has both RTL and LTR capability. This is called a bidirectional character set.
  - A dual direction character set is necessary because the character set of a language contains not only those characters used in the language, it also contains characters of other commonly used languages, especially ISO Latin-1 characters.
  - Hebrew and Arabic are bidirectional character sets.

Uniface Mappings to Installable Character Sets [1]

- During Uniface installation, it is possible to choose a character set that is appropriate for one or more languages.
  - Western European is appropriate for English, German, French, and so on
  - Cyrillic is appropriate for Slavic languages such as Russian
- The character set name is generic, for example Western European, Cyrillic, or Simplified Chinese, but the actual character set installed is specific to the platform.
- Different platforms use different character set standards. Uniface uses one name to represent the different character sets that represent the same characters. For example, `CP1251` and `ISO 8859-5` are character sets in different platforms to represent Cyrillic characters. Uniface uses ‘Cyrillic’ to represent all of them. The generic character set is also referred to as a meta variant.

Uniface Mappings to Installable Character Sets [2]

- Each meta variant is mapped to several installable character sets, as listed in the following table:

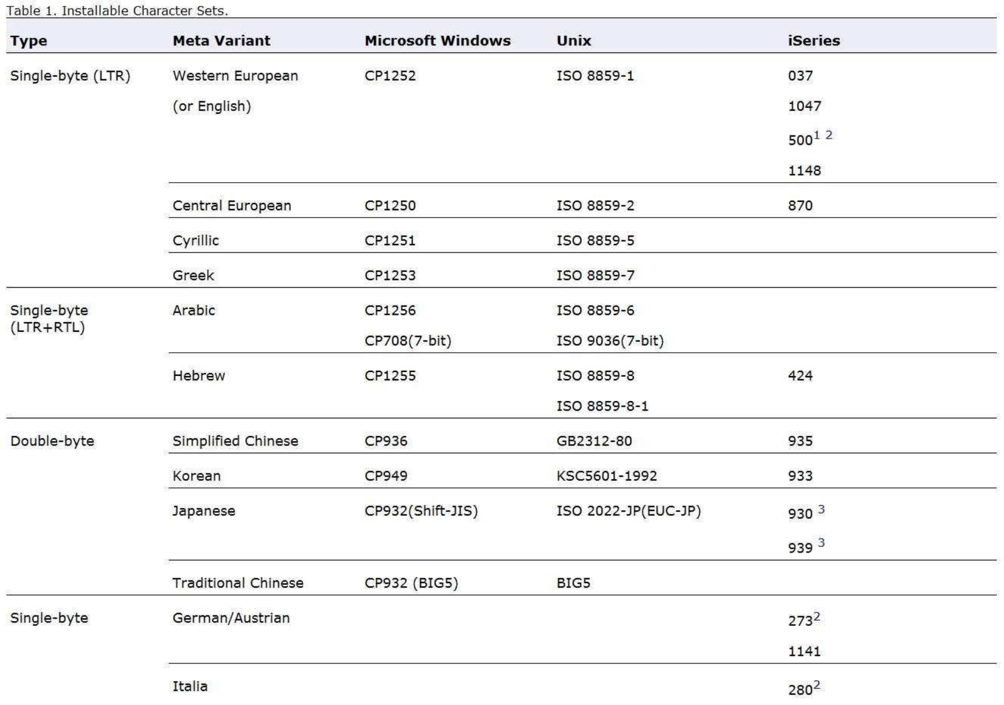


Table 1. Installable Character Sets.

Type	Meta Variant	Microsoft Windows	Unix	iSeries
Single-byte (LTR)	Western European	CP1252	ISO 8859-1	037
	(or English)			1047
				500[1] [2]
				1148
	Central European	CP1250	ISO 8859-2	870
	Cyrillic	CP1251	ISO 8859-5	
	Greek	CP1253	ISO 8859-7	
Single-byte (LTR+RTL)	Arabic	CP1256	ISO 8859-6	
		CP708(7-bit)	ISO 9036(7-bit)	
	Hebrew	CP1255	ISO 8859-8	424
			ISO 8859-8-1	
Double-byte	Simplified Chinese	CP936	GB2312-80	935
	Korean	CP949	KSC5601-1992	933
	Japanese	CP932(Shift-JIS)	ISO 2022-JP(EUC-JP)	930 [3]
				939 [3]
	Traditional Chinese	CP932 (BIG5)	BIG5	
Single-byte	German/Austrian			273[2]
				1141
	Italia			280[2]

Installing and Configuring Character Sets [1]

- Uniface needs to be installed with the correct character set for a specific language and platform.
- Settings in the `.ini` and `.asn` determine which character sets are used.
  - Some settings are common for all types of character sets.
  - Others are for specific types of character sets.
- Platform Requirements (Microsoft Windows)
  - Microsoft Windows—a single-byte character set can be installed either on the localized operating system (for example, Central European Uniface on a Czech Windows system, Arabic on a Arabic Windows) *or* on an English Windows that has a multi-language setup (called an *enabled version*)
    For double-byte character sets, a localized Windows is required. For example, install Japanese Uniface in a Japanese Windows system, Simplified Chinese in a Simplified Chinese system, and Korean in a Korean system.

Installing and Configuring Character Sets [2]

  - EBCDIC-based platforms—the installed character set should match the language of the job.
  - Unix platforms—the installed character set should match the language of the operating system.

- To install and configure a character set:
  - Install Uniface, as appropriate for your platform. During installation:
    - When prompted for the directory in which to install Uniface, avoid using non-ASCII characters in directory names.
    - In the Character set selection screen, select the generic character set you want to install.
  - In the assignment and initialization files, set the configuration settings as appropriate. On Microsoft Windows, the Uniface installation program generates settings for the selected character set. On other platforms, you need to ensure that configuration settings are appropriately set.

Settings for All Character Sets [1]

- These .ini and .asn settings are used for all character sets

Settings in .ini file	
`NLS`	Specify the character set that is supported by the current Uniface installation
`Profile`	Specify the display profile characters
`[printer]`	Some GUI settings are dependent on the character set. For example, a Simplified Chinese Uniface version, has setting **`HyperLabel=Arial,Simplified Chinese,8,underline`**.

Translation Tables in Assignment File	
`$SYS_CHARSET`	Specify a character set used to communicate with the operating system
`$DEF_CHARSET`	Specify a character set used to communicate with the database
`$KEYBOARD`	Specify a translation table used for entering characters
`$DISPLAY`	Specify a translation table used for displaying characters

Settings for All Character Sets [2]

- **`$LANGUAGE`**
  Uniface installation program initializes the **`$LANGUAGE`** with the language abbreviation of the installed character set in the **`usys.asn`** file.
- **`[META_LANGUAGE]`**
  In earlier Uniface versions, its internal character set was the meta character set. From version 9, Unicode becomes its internal character set. For the compatibility reason, the meta character set is still used. These result in some problems. This assignment section is used to define your own character mapping to solve the problems.
- **`DEFAULTENCODING`**
  For a web application, the browser should be able to display the specific language. Make sure that the browser's Encoding view is switched to the correct language. To do so, set the **`DEFAULTENCODING`** parameter.

Settings for Specific Character Sets

- You can specify the layout orientation of the application.
  This is important for the bidirectional character sets such as Arabic and Hebrew.
- To use Japanese Gaiji characters, use $GAIJI setting
- To select the sorting order for a double-byte character set, use $NLS_SORT_ORDER.

Assignment Settings for RTL Character Sets

Settings for RTL	Description
`$RTL_APPLICATION`	Specify the layout orientation of forms and menus during deployment.
`$RTL_FIELDS`	Specify the layout orientation of fields during deployment.
`$NO_UNIFIELDS`	Replace unifield by an edit box because $rtl_field=on cannot be applied to a unifield.
`$RTL_PRINTING`	If the printer cannot handle right-to-left printing, use $rtl_printing.

Settings for Double-Byte Character Sets

Settings for Double-Byte	Type	Description
`$GAIJI`	Assignment	Set $GAIJI to True to enable Gaiji characters. The Uniface 8 setting $DOUBLE_WIDTH = 8 is equivalent to the Uniface 9 setting $GAIJI = True.
`$NLS_SORT_ORDER`	Assignment	Determine how the Proc statement sort and sort/list order data during sorting.
`AutoImeOpen`	Initialization	If AUTOIMEOPEN is on, the IME is switched to the language-specific input mode, for example Hiragana, when the focus moves to a string field. To select an Input Method mode for an individual string field, use field syntax YIME or NIME. For more information, see Field Syntax Shorthand.
`AutoImeGold`	Initialization	If AUTOIMEGOLD is on, the IME is switched to direct input mode when the GOLD key is used. Once the complete GOLD key sequence has been entered, the IME is returned to its previous state.

Configuration Settings for Installable Character Sets on Windows

- Following slides contain the contents of the usys.ini and the usys.asn files generated by the Uniface installation program for each installable character set in the Windows platform.

Central European Character Set

- .ini file:

```
 [UNIFACE_DLLS ]
NLS          = CEN

 [SCREEN]
Label        = Arial,Central European,8,regular
HyperLabel   = Arial,Central Western,8,underline
FormText     = Courier New,Central European,9,regular
Font0        = Courier New,Central European,9,regular
EditFont     = Arial,Central European,8,regular
ListFont     = Arial,Central European,8,regular
GFP          = Arial,Central European,8,regular
ButtonFont   = Arial,Central European,8,regular
Combo        = Arial,Central European,8,regular
Debug        = Arial,Central European,8,regular
Buttons      = Arial,Central European,8,regular
Messagefont = Arial,Central European,8,regular
```

- .asn file:

```
 [SETTINGS ]
$LANGUAGE     = CEN
$DISPLAY      = mswin3_1250
$KEYBOARD     = mswinx_1250
$DEF_CHARSET = cp1250
$SYS_CHARSET = cp1250
```

Cyrillic Character Set

- .ini file:

```
 [UNIFACE_DLLS ]
NLS            = CYR

 [SCREEN]
Label          = Arial,Cyrillic,8,regular
HyperLabel     = Arial,Cyrillic,8,underline
FormText       = Courier New,Cyrillic,9,regular
Font0          = Courier New,Cyrillic,9,regular
EditFont       = Arial,Cyrillic,8,regular
ListFont       = Arial,Cyrillic,8,regular
GFP            = Arial,Cyrillic,8,regular
ButtonFont     = Arial,Cyrillic,8,regular
Combo          = Arial,Cyrillic,8,regular
Debug          = Arial,Cyrillic,8,regular
Buttons        = Arial,Cyrillic,8,regular
Messagefont = Arial,Cyrillic,8,regular
```

- .asn file:

```
 [SETTINGS ]
$LANGUAGE     = CYR
$DISPLAY      = mswin3_1251
$KEYBOARD     = mswinx_1251
$DEF_CHARSET = cp1251
$SYS_CHARSET = cp1251
```

Arabic Character Set

- .ini file:

```
[UNIFACE_DLLS ]
NLS           = ARB

[SCREEN]
Label         = Arial,Arabic,8,regular
HyperLabel    = Arial,Arabic,8,underline
FormText      = Courier New (Arabic),9,regular
Font0         = Courier New (Arabic),9,regular
EditFont      = Arial,Arabic,8,regular
ListFont      = Arial,Arabic,8,regular
GFP           = Arial,Arabic,8,regular
ButtonFont    = Arial,Arabic,8,regular
Combo         = Arial,Arabic,8,regular
Debug         = Arial,Arabic,8,regular
Buttons       = Arial,Arabic,8,regular
Messagefont = Arial,Arabic,8,regular
```

- .asn file:

```
 [SETTINGS ]
$LANGUAGE     = ARB
$RTL_APPLICATION
$NO_UNIFIELDS
$DISPLAY      = mswin3_1256
$KEYBOARD     = mswinx_1256
$DEF_CHARSET = cp1256
$SYS_CHARSET = cp1256
```

Greek Character Set

- .ini file:

```
 [UNIFACE_DLLS ]
NLS            = ELL

 [SCREEN]
Label          = Arial,Greek,8,regular
HyperLabel     = Arial,Greek,8,underline
FormText       = Courier New,Greek,9,regular
Font0          = Courier New,Greek,9,regular
EditFont       = Arial,Greek,8,regular
ListFont       = Arial,Greek,8,regular
GFP            = Arial,Greek,8,regular
ButtonFont     = Arial,Greek,8,regular
Combo          = Arial,Greek,8,regular
Debug          = Arial,Greek,8,regular
Buttons        = Arial,Greek,8,regular
Messagefont    = Arial,Greek,8,regular
```

- .asn file:

```
 [SETTINGS ]
$LANGUAGE    = ELL
$DISPLAY     = mswin3_1253
$KEYBOARD    = mswinx_1253
$DEF_CHARSET = cp1253
$SYS_CHARSET = cp1253
```

Hebrew Character Set

- .ini file:

```
 [UNIFACE_DLLS]
NLS            = HEB

 [screen]
Label          = Arial,Hebrew,8,regular
HyperLabel     = Arial,Hebrew,8,underline
Font0          = Courier New,Hebrew,9,regular
EditFont       = Arial,Hebrew,8,regular
ListFont       = Arial,Hebrew,8,regular
GFP            = Arial,Hebrew,8,regular
ButtonFont     = Arial,Hebrew,8,regular
Combo          = Arial,Hebrew,8,regular
Debug          = Arial,Hebrew,8,regular
Buttons        = Arial,Hebrew,8,regular
Messagefont    = Arial,Hebrew,8,regular
```

- .asn file:

```
 [SETTINGS ]
$LANGUAGE     = HEB
$RTL_APPLICATION
$NO_UNIFIELDS
$DISPLAY      = mswin3_1255
$KEYBOARD     = mswinx_1255
$DEF_CHARSET = cp1255
$SYS_CHARSET = cp1255
```

Simplified Chinese Character Set

- .ini file:

```
 [UNIFACE_DLLS]
NLS            = CHZ

 [SCREEN]
Label          = Arial,Western,11,regular
HyperLabel     = Arial,Simplified Chinese,8,underline
FormText       = Courier New,Western,12,regular
Font0          = Courier New,Western,12,regular
EditFont       = Arial,Western,11,regular
ListFont       = Arial,Simplified Chinese,11,regular
GFP            = Arial,Western,11,regular
ButtonFont     = Arial,Western,11,regular
Combo          = Arial,Western,11,regular
Debug          = Arial,Western,11,regular
Buttons        = Arial,Western,11,regular
Messagefont    = Arial,Western,11,regular
```

- .asn file:

```
 [SETTINGS ]
$LANGUAGE     = CHZ
$DISPLAY      = mswin3_gb
$KEYBOARD     = mswinx_gb
$DEF_CHARSET  = gb
$SYS_CHARSET  = gb
```

Korean Chinese Character Set

- .ini file:

```
 [UNIFACE_DLLS]
NLS            = KOR

 [SCREEN]
Label          = Arial,Korean,10,regular
HyperLabel     = Arial,Korean,8,underline
FormText       = Courier New,Korean,11,regular
Font0          = Courier New,Korean,11,regular
EditFont       = Arial,Korean,10,regular
ListFont       = Arial,Korean,10,regular
GFP            = Arial,Korean,10,regular
ButtonFont     = Arial,Korean,10,regular
Combo          = Arial,Korean,10,regular
Debug          = Arial,Korean,10,regular
Buttons        = Arial,Korean,10,regular
Messagefont    = Arial,Korean,10,regular
```

- .asn file:

```
 [SETTINGS ]
$LANGUAGE     = KOR
$DISPLAY      = mswin3_ksc
$KEYBOARD     = mswinx_ksc
$DEF_CHARSET  = ksc
$SYS_CHARSET  = ksc
```

Japanese Character Set

- .ini file:

```
[UNIFACE_DLLS]
NLS           = JPN

[SCREEN]
Label         = Arial,Japanese,8,regular
HyperLabel    = Arial,Japanese,8,underline
FormText      = FontName,Japanese,11,regular
Font0         = FontName,Japanese,11,regular
EditFont      = Arial,Japanese,10,regular
ListFont      = Arial,Japanese,10,regular
GFP           = Arial,Japanese,10,regular
ButtonFont    = Arial,Japanese,10,regular
Combo         = Arial,Japanese,10,regular
Debug         = Arial,Japanese,10,regular
Buttons       = Arial,Japanese,10,regular
Messagefont = Arial,Japanese,8,regular
```

- .asn file:

```
 [SETTINGS ]
$LANGUAGE     = JPN
$DISPLAY      = mswin3_SJIS
$KEYBOARD     = mswinx_SJIS
$DEF_CHARSET = SJIS
$SYS_CHARSET = SJIS
$GAIJI        = false
```

Traditional Chinese Character Set

- .ini file:

```
[UNIFACE_DLLS]
NLS            = CHT

[SCREEN]
Label          = Arial,Western,11,regular
HyperLabel     = Arial,Traditional Chinese,8,underline
FormText       = Courier New,Western,12,regular
Font0          = Courier New,Western,12,regular
EditFont       = Arial,Western,11,regular
ListFont       = Arial,Simplified Chinese,11,regular
GFP            = Arial,Western,11,regular
ButtonFont     = Arial,Western,11,regular
Combo          = Arial,Western,11,regular
Debug          = Arial,Western,11,regular
Buttons        = Arial,Western,11,regular
Messagefont    = Arial,Western,11,regular
```

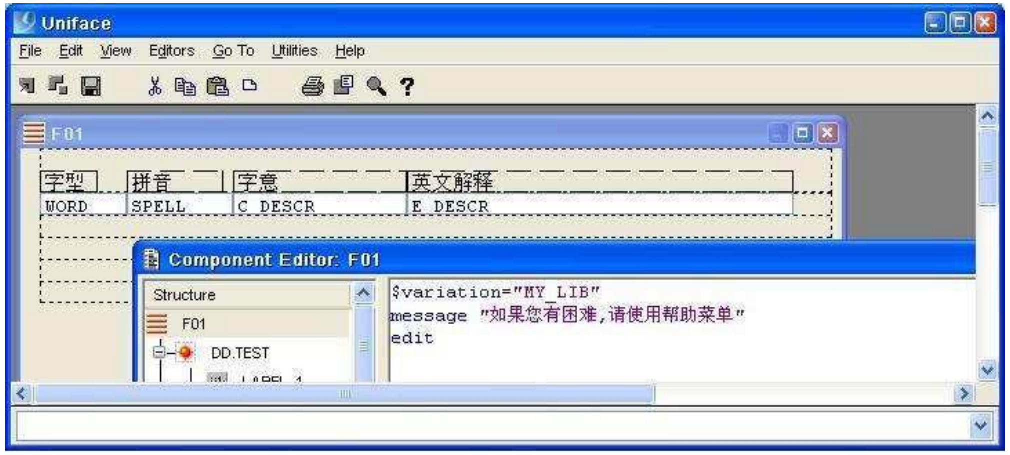
- .asn file:

```
 [SETTINGS ]
$LANGUAGE     = CHT
$DISPLAY      = mswin3_big5
$KEYBOARD     = mswinx_big5
$DEF_CHARSET  = big5
$SYS_CHARSET  = big5
```

Developing a Language-Specific Application

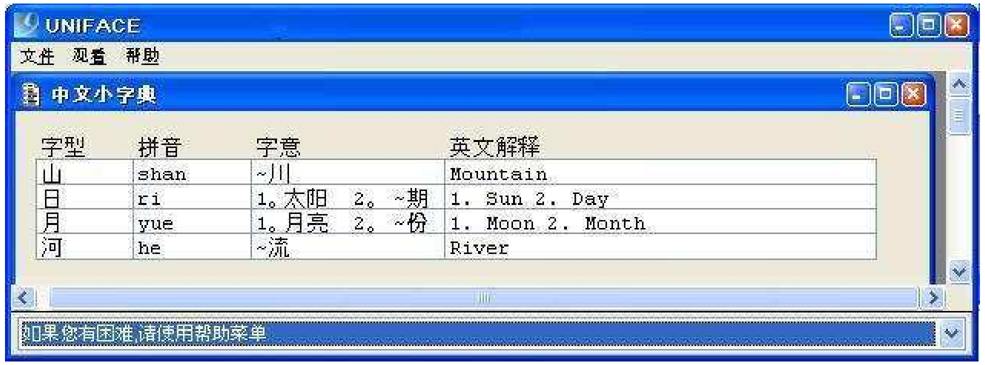
- In a language specific Uniface, you can develop an application entirely in that language.
- The Uniface (9) Development Environment has an English user interface, even if you installed Uniface with a non-English character set. However, you can enter the non-English characters in labels, menu items, and so on to develop an application completely in that language.
- In a Uniface version installed with a specific character set, there are some restrictions on where the characters can be used.
  - In general, text that is visible to the user of your application can be in the specific language, for example, labels, menu items, ValRep lists of widgets, and so on.
  - Most object names can only use ASCII characters, for example, form names, template names, menu names, and so on.

A Simplified Chinese Uniface application



字型	拼音	字意	英文解释
WORD	SPELL	C_DESCR	E_DESCR

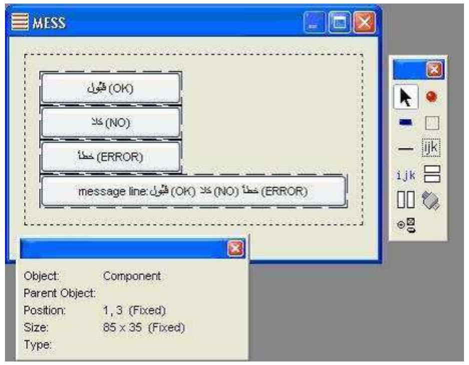
```
$variation="MY_LIB"
message "如果您有困难,请使用帮助菜单"
edit
```



字型	拼音	字意	英文解释
山	shan	~川	Mountain
日	ri	1。太阳 2。 ~期	1. Sun 2. Day
月	yue	1。月亮 2。 ~份	1. Moon 2. Month
河	he	~流	River

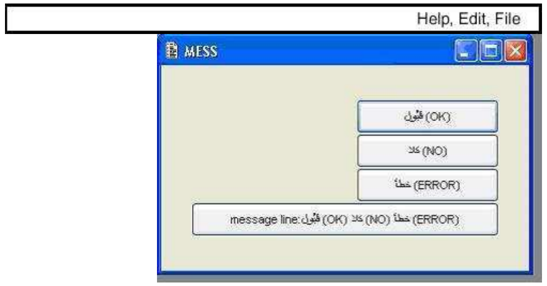
如果您有困难,请使用帮助菜单

Bidirectional Character Sets



message line: قبول (OK) كلا (NO) خطأ (ERROR)

Object: Component
Parent Object:
Position: 1 , 3 (Fixed)
Size: 85 x 35 (Fixed)
Type:



message line: قبول (OK) كلا (NO) خطأ (ERROR)

Language and Locale [1]

- Different languages and regions use different conventions when sorting and displaying data, for example:
  - English amount: £ 123,456.89
  - Europian amount: € 123.456,89
- The term locale is commonly used to designate the combination of language and country or region.
  The locale determines the default behaviour for:
  - Sorting data
  - Formatting numbers
  - Formatting date and time
  - Formatting currency
  - Switching between uppercase and lowercase
  - Case-sensitive and case-insensitive pattern matching in strings
  - Setting the time zone
- Control and configure the locale and associated behavior using:
  - Proc functions
  - Assignment settings
  - Display formats

Language and Locale [2]

- The language of an application is associated not only with a character set, but with regional conventions for sorting, for expressing numbers, currency, date and time, and so on.
- The combination of language and region is known as locale. For example, the locale can affect the way data is displayed:

Locale Code	Locale	Currency	Date	Time
en_US	English (United States)	$12,345.67	Wednesday, December 2, 2009	4:08:42 PM
en_GB	English (United Kingdom)	£12,345.67	Wednesday, 2 December 2009	16:08:42
fr_FR	French (France)	12 345,67 €	mercredi 2 décembre 2009	16:08:42
fr_CA	French (Canada)	12 345,67 $	mercredi 2 décembre 2009	16:08:42
nl_NL	Dutch (Netherlands)	€ 12.345,67	woensdag 2 december 2009	16:08:42
ja_JP	Japanese (Japan)	￥12,346	2009年12月2日水曜日	16:08:42
bg_BU	Bulgarian (Bulgaria)	12 345,67 лв.	02 декември 2009, сряда	16:08:42

Language and Locale []

- Setting the Locale
  Uniface enables you to specify a locale using the $NLS_LOCALE or the **`$nlslocale`** Proc function. The value can be: **`classic`**, **`system`** or ***`Locale`***

- Fine-Tuning Locale-Based Processing
  There may be circumstances when you want the locale to be ignored.
  - The Uniface mechanism for controlling locale-based processing is based on the NLS locale. If it is set to a locale (or to system), and no other settings are applied, NLS processing rules are applied in all circumstances.
  - To switch off locale-based processing, you can use NLS assignment settings and Proc functions that control specific areas of locale-related functionality.

Time Zones [1]

- When handling date and time data, Uniface assumes the date and time are given in the local time zone as defined on the system where processing occurs.
- You can override this assumption by explicitly defining external and/or internal time zones using Proc and assignment settings.

Proc Function	Assignment Setting	Description
`$nlstimezone`	`$NLS_TIME_ZONE`	Specify the external time zone to use.
`$nlsinternaltime`	`$NLS_INTERNAL_TIME`	Specify the internal (server) time zone to use.

- The Proc functions can also be used to check the current values.

Time Zones [2]

- Setting any of these settings and Proc functions influences the date and time values that are:
  - Returned by the Proc functions **`$clock`**, **`$date`**, and **`$datim`**
  - Displayed in fields with data types date, time, or combined date and time
  - Stored and retrieved in the database
  - Exchanged when using XML, call-in, or call-out

- External Time Zone
  If the external time zone is set to a specific time zone, Uniface uses this time zone to interpret and display date and time data.
  Thus if the local time is 11:13:48 AM on 3 December 2009 in Amsterdam, the value **`$datim`** returns is adjusted depending on the how the external time zone is set:

Time Zones [3]

Time Zone	$datim	Date and Time
Europe/Amsterdam	20091203111134800	3 December 2009, 11:13:48 AM
America/Detroit	20091203051134800	3 December 2009, 5:13:48 AM
Pacific/Pago-Pago	20091202231134800	3 December 2009, 23:13:48 PM

- Internal Time Zone
  - For applications that store date or time data, it makes sense to use UTC+00:00 as the internal time zone, so that all data conforms to a standard time.
  - For example, if you place an order at 9:00 AM in Detroit, the date is corrected to the UTC time of UTC-05:00. Corrections for Daylight Savings Time are also applied.
  - When retrieving data from the database, the UTC time is corrected to display the time according to the external time zone.

Support for Unicode in database connectors

Unicode Support in Databases and Connectors

DBMS Functionality	Use Full Range Unicode	Use Installable Char Set
sql Proc instruction	Yes for DB2, Oracle, Solid, MySQL, SQLite, Sybase, and TXT [Note 1] No for other connectors	Yes
SQL Workbench	Yes for DB2, Oracle, Solid, MySQL, SQLite, Sybase, and TXT [Note 1] No for other connectors	Yes
Error messages	Yes for DB2 [Note 1], Oracle, Solid, SQLite, and MySQL No for other connectors	Yes
Database table names	Yes for DB2 and MySQL [Note 1] No for other connectors	Yes
Database names, user names / passwords system user names / passwords	Yes for Oracle and MySQL No for other connectors	Yes
Parameters of stored procedures	No, except in DB2 and Sybase	No, except in DB2 and Sybase
DBMS service stored procedures	Yes for Sybase and Oracle [Note 1] No for other connectors	No

Note 1: Only in a Unicode Database and $DEF_CHARSET = UTF8

Unicode in SLE (SQLite) connector

- The Uniface SLE connector supports the full Unicode range

Enable Unicode by:

- Using wide-character packing codes (such as W20)
  W packing codes are stored as UTF-8 and support the full Unicode range
- The assignment setting **`$DEF_CHARSET=UTF8`**
  Ensures that the Uniface C packing code is mapped to W packing code
  This also enables Unicode characters to be used in the SQL Workbench and the **`sql`** Proc statement

Unicode in ORA (Oracle) connector

- The Uniface ORA connector supports the full Unicode range
- The sql Proc statement and SQL Workbench can handle Unicode data from Oracle
- Unicode is also supported in user names and passwords, service stored procedures, and in Oracle error messages
- **Note:** To be able to work with Unicode in SQL Workbench, Oracle Service Stored procedures, and **`where`** clauses, **`$def_charset`** must be set to **`UTF8`**

Unicode in PGS (PostgreSQL) connector

- The Uniface PGS connector supports the full Unicode range

Enable Unicode by:

- The assignment setting **`$DEF_CHARSET=UTF8`**
  Ensures that the Uniface C packing code is mapped to W packing code.
- **Important:** For PostgreSQL, ensure that the **`$DEF_CHARSET=UTF8`** is set for *all applications* that access the data via the PGS connector
  - otherwise possible corrupted data
- By default, Unicode support for W packing codes is limited to characters in the Basic Multilingual Plane (BMP).
  For full Unicode range (SMP), Only for PGS and SOL, use:

```
[DRIVER_SETTINGS]
USYS$PGS_PARAMS = smp : on | off
```

Unicode in MQL (MySQL) connector [1]

- The Uniface MQL connector can support Unicode. By default, Unicode support is disabled
- MySQL versions 5.0 and 5.1 provide Unicode support through the `utf8` and the `ucs2` character sets, which support the Unicode Basic Multilingual Plane (BMP)
- MySQL 5.5 provides a full range of Unicode character sets—`utf32, utf16, ucs2, utf8mb4, and utf8`

Unicode in MQL (MySQL) connector [2]

Enable Unicode by:

- Setting the connector option **`unicode=on`**—enables fields with wide-character packing codes (such as W10, SW*) to be mapped to the **`utf8`** character set
- Setting **`$DEF_CHARSET=UTF8`** sets the default character set for communicating with the MySQL Server to **`utf8`**
  Note: Set $DEF_CHARSET=UTF8 only if the default character set of the MySQL database is utf8

Migrating to Unicode

Migrating to Unicode

- Why migrate to Unicode?
- Planning your migration
- Understanding the current use of character encodings
- Checking the foundations
- Deciding on character encoding use for internal use
- Deciding on character encoding use for external interfaces
- Creating a road map
- Designing for Unicode
- Migrating Data
- Testing with Unicode

Further reading

- Uniface Library - Developing International Applications
- World Wide Web Consortium - Migrating to Unicode

Thank You & Questions

arjen.van.vliet@uniface.com

UNIFACE
uniface.com