Windows NLS Considerations
version 2.1

Radoslav Rusinov
Radoslav.Rusinov.remove._.spam@gmail.com
Contents

1. Introduction .................................................................................................................................................. 3
   1.1. Windows and Code Pages .................................................................................................................. 3
   1.2. CharacterSet ...................................................................................................................................... 3
   1.3. Encoding Scheme ............................................................................................................................. 3
   1.4. Fonts ................................................................................................................................................... 4
   1.5. So Why Are There Different Charactersets? ...................................................................................... 4
   1.6. What are the Difference Between 7 bit, 8 bit and Unicode Charactersets? ........................................ 4

2. NLS_LANG .................................................................................................................................................. 4
   2.1. Setting the Character Set in NLS_LANG ............................................................................................. 4
   2.2. Where is the Character Conversion Done? ....................................................................................... 7
   2.3. The correct NLS_LANG for my Windows ANSI Code Page ............................................................... 7
       2.3.1. Determine your Windows ANSI code page .............................................................................. 7
       2.3.2. Find the correspondent Oracle client character set ............................................................... 8
   2.4. The correct NLS_LANG for my DOS / Command Prompt OEM Code Page ..................................... 9
   2.5. List of common NLS_LANG's used in the Windows Registry ............................................................ 10
   2.6. List of common NLS_LANG's used in the Command Prompt (DOS box) ........................................ 11
   2.7. How Windows uses Fonts to display the different charactersets ...................................................... 11
   2.8. What about Multiple Homes on Windows? ...................................................................................... 12

3. Advanced NLS Research ............................................................................................................................ 12
   3.1. What is Locale Builder? ....................................................................................................................... 12
   3.2. How to see what's really stored in the database? ............................................................................... 13
   3.3. How to define NLS_LANG with OEM (enterprise manager) version 2 and 9? ................................. 14

4. Appendixes .............................................................................................................................................. 15
   Appendix A: List of all supported Charactersets for European Languages in Oracle9i ........................... 15
   European Language Character Sets ...................................................................................................... 15
   Table A-5 European Language Character Sets ....................................................................................... 15
   Appendix B: References ............................................................................................................................ 20

Note: this document uses many quotes from several important NLS Metalink notes and is not fully authored by its creator. The role of the author is to make more valuable and full all-in-one source, related to NLS problems and issues than existing sources and articles about the discussed topic.
1. Introduction

This document is designed to help to all Oracle Developers and Administrators in solving of NLS-specific issues under Windows Operating System. This document could be used from every specialist who is trying to solve some problem related to language settings between client and Oracle database, displaying special symbols inside any 3rd party application and corruption of special symbols while they are inserted into the database.

1.1. Windows and Code Pages

On Windows systems, the encoding scheme (=Characterset) is specified by a Code Page. Code Pages are defined to support specific languages or groups of languages, which share common writing systems. From Oracle point of view the terms Code Page and Characterset mean the same.

Note that in non Chinese-Japanese-Korean environments, the Windows GUI and DOS command prompt do not use the same code page (!). As a result windows uses 2 different charactersets for the ANSI (sqlplusw.exe) and to OEM (dos box - sqlplus.exe) environments.

So, you can keep in mind that every Windows-based application (the GUI applications) is using ANSI characterset, and DOS-based applications (including command prompt) are using OEM (or lets say DOS-based characterset).

Example:

to illustrate the issue:
open notepad, type some "üéèç□", save that file as c:\test.txt
(make sure that the encoding save as setting of notepad is set of ANSI)
open a dos box
Start - Run - cmd - ok
and type

C:\Documents and Settings\user>edit c:\test.txt

you see "³ÚÞþÓ" in edit

Here you see a file in "ANSI" encoding (notepad) displayed by the OEM (dos box) environment of edit. You see clearly that Windows do not use the same characterset for the ANSI and CMD / DOS box environments.

1.2. CharacterSet

A character set is only a collection of characters without any glyphs associated with them. Characters are represented by so-called character codes. Character codes are generated and stored when a user inputs data into for example a document. Single-Byte character sets (SBCS) provide 256 character codes. Most SBCS-es are in use for American and European countries. For some countries 8 bits are not sufficient to contain all characters used, as in for example the Far East. In these countries it can occur that about 12,000 characters may be addressed at any one time. For these countries Multi-Byte character sets (MBCS) are used. When dealing with MBCS you encounter both fixed-width and variable-width encodings. Fixed-width encodings use a fixed number of <n> bytes for each stored character, where n >=2. Variable-width encodings store characters in 1 or more bytes, where some character can be stored in fewer bytes than another character of the same character set. For example, Unicode is a variable multi-byte encoding that encompasses many characters used in general text interchange throughout the world.

1.3. Encoding Scheme
Besides the problem of visualizing the characters, there also needs to be a standard of how to interpret the data stored in documents, databases and such.

A character set defines characters to be placed in a particular order and with a particular character code; for example, the capital letter A in the U.S. ASCII character set has character code 65 and the capital letter S in the same character set has code 83). The result of assigning a character code to each individual character of a character set is called an encoding scheme.

1.4. Fonts

A font is a collection of glyphs (from "hieroglyphs") that share common appearances (typeface, character size). A font is used by the operating system to convert a numeric value into a graphical representation on screen.

A font does not necessarily contain a graphical representation for all numeric values defined in the code page you are using. That's why you get sometimes, black squares on the screen if you change fonts and the new that font has no representation for a certain symbol.

The Windows "Character Set Map" utility can be used to see which glyphs are parts of a certain font.

A font also implements a particular code page or set of code pages. For example, the Arial font implements the code pages 1252, 1250, 1251, 1253, 1254, 1257.

1.5. So Why Are There Different Charactersets?

Two main reasons:

* Historically vendors have defined different 'sets' for their hardware and software, mainly because there were no official standards.

* New character sets have been defined to support new languages. With an 8-bit characterset, you are limited in the number of symbols you can support so there are different sets for different written languages.

1.6. What are the Difference Between 7 bit, 8 bit and Unicode Charactersets?

- A 7-bit characterset only knows 128 symbols ($2^7$)
- An 8-bit characterset knows 256 symbols ($2^8$)
- Unicode (UTF8) is a multibyte characterset.

The latest version of the Unicode standard (3.1) defines 94,140 encoded characters. Unicode has the capability to define over a million characters. Oracle has several revisions of the Unicode standard implemented during the years:
- AL24UTFFSS (Unicode version 1.1) Introduced in V7 and now obsolete.
- UTF8 (Unicode version 2.0) introduced in V8.
- AL32UTF8 (Unicode version 3.1) introduced in V9.

2. NLS_LANG

2.1. Setting the Character Set in NLS_LANG

When dealing with an Oracle environment, you normally want to set up support for a particular character set. You do this by setting the NLS_LANG variable to contain a reference to a particular character set to be used by either the Oracle client or by creating a database using a particular character set.
Windows NLS Considerations

The character set chosen merely defines the encoding scheme to be used by the Oracle product. For example, when using the WE8ISO8859P1 Oracle character set, you indicate that the character codes will be interpreted using the "ISO/IEC 8859-1:1998 8-bit single-byte coded graphic character sets -- Part 1: Latin alphabet No. 1" encoding scheme.

Especially at the Oracle client side, the encoding scheme used by Oracle must match the encoding scheme used by the I/O system itself. This means that the font definition used by the OS and the character set used by the Oracle client must use the same character codes for the same characters.

For example, the Windows fonts files for Western Europe implement at least the code page 1252 (called "Latin 1"). The latest font files contain the Euro glyph with character code 128. Usually, when you want to use the Euro glyph in an Oracle environment, you want to use the WE8ISO8859P15 character set. However, in this character set the Euro glyph has character code 164. This means the encoding schemes of the Windows code page 1252 and Oracle character set WE8ISO8859P15 do not match.

You can best check the definitions of the encoding schemes to clarify what is meant above. As can be seen on the Microsoft site, the layout for code page 1252 (which Microsoft Corp. has published on the [http://www.microsoft.com/typography/unicode/1252.gif](http://www.microsoft.com/typography/unicode/1252.gif) web site) indicates that the Euro symbol can be found at character code 0x80 (= 128 decimal).

However, the WE8ISO8859P15 character set (which implements the ISO Latin 9 standard) uses character code 0xA4 (= 164 decimal) for the same character, as shown at [http://czyborra.com/charsets/iso8859.html#Future](http://czyborra.com/charsets/iso8859.html#Future).

So for example when on Windows you use the Oracle client character set WE8ISO8859P15 and type <Alt>+0128 to indicate an Euro glyph, this means that you will actually send the character code 128 to the database and not character code 164.

When the encoding schemes of the font (read: code page) and Oracle character set do not match, an incorrect character code will be sent by the client to store in the server. This will lead to logical data corruptions.

In this particular example you should use the Oracle character set WE8MSWIN1252, which uses exactly the same encoding scheme as the Windows code page does.

Please **note** that this has nothing to do with the database character set in use. When you want to store e.g. the Euro symbol in the database, you can still set the database character set to WE8ISO8859P15, whereas the client uses WE8MSWIN1252. The Oracle software will correctly translate any symbols to the new character code values of the database character set when storing data into the database or retrieving data from the database.

To specify the locale behavior of your client Oracle software, you have to set your NLS_LANG parameter. It sets the language, territory and also the character set of your client. For a short overview, it uses the following format:

\[
\text{NLS\_LANG} = \text{LANGUAGE\_TERRITORY\_CHARACTERSET}
\]

where:

- **LANGUAGE** specifies:
  - language used for Oracle messages,
  - day names and month names

- **TERRITORY** specifies:
  - monetary and numeric formats,
  - territory and conventions for calculating week and day numbers

- **CHARACTERSET**:
  - controls the character set used by the client application
    - * or it matches your Windows code page
    - * or it set to UTF8 for an Unicode application

* **NLS\_LANG** is used to let Oracle know what characterset your client's OS is **USING** so that Oracle can do (if needed) conversion from the client's characterset to the database characterset.

**4 important remarks:**
Windows NLS Considerations

* The characterset defined with the NLS_LANG parameter does **NOT CHANGE** your client's characterset, it is used to let Oracle know what characterset you are **USING** on the client side, and so Oracle can do the proper conversion. You cannot just set **NLS_LANG** to the characterset you **WANT**. If you need Hebrew support (for example) on an Cyrillic Windows then that Windows need to be changed to have an 1255 ACP, just setting the **NLS_LANG** to Hebrew will **NOT** allow you to retrieve/store Hebrew.

* Another myth is that if you don't set the **NLS_LANG** on the client it uses the **NLS_LANG** of the server. This is also **NOT** true! The characterset part of the **NLS_LANG** parameter is never inherited from the server.

* Note that LANGUAGE and TERRITORY have nothing to do with the ability to *store* characters in a database. A **NLS_LANG** set to JAPANESE_JAPAN.WE8MSWIN1252 will not allow you to store Japanese, as WE8MSWIN1252 doesn't know Japanese characters.

* Setting the **NLS_LANG** to the characterset of the database **MAY** be correct but **IS NOT ALWAYS** correct. Please DO **NOT** assume that **NLS_LANG** needs to be **ALWAYS** the same as the database characterset. **THIS IS NOT TRUE**. You must make sure that the font (read: code page used by the font) always matches the encoding scheme of the Oracle character set. If not, you will end up with logically corrupt data. If the **NLS_LANG** is the same as the database characterset then Oracle (for performance reasons) will do **NO** validation of the characterset. See "A detailed example of a *wrong* NLS setup to understand what's going on"

A detailed example of a *wrong* NLS setup to understand what's going on:

You have created a database on your Unix box with the US7ASCII characterset. Your Windows clients works with the MSWIN1252 characterset (regional settings -> Western Europe /ACP 1252) and you, as DBA, use the Unix shell (ROMAN8) to work on the database. You set **NLS_LANG** to AMERICAN_AMERICA.US7ASCII on the clients and the server.

****** Note: this is an **INCORRECT** setup to explain characterset conversion, don't use it in your environment! ******

a very important point (as mentioned before):
When the client **NLS_LANG** characterset is set to the same value as the database characterset, Oracle assumes that the data being sent or received are of the same (correct) encoding, so no conversions or checks are performed. The data is just stored as delivered by the client, bit by bit...
No way that Oracle can find out if you "lie" by using an incorrect setup ☺.

Let's do something now:
You insert an 'e'(LATIN SMALL LETTER E WITH ACUTE) into a table NLS_TEST containing one column "TEST" of the type 'char'.

As long as you insert into and select from the column on Windows NT clients with the MSWIN1252 characterset everything runs smoothly. No conversion is done and 8 bits are inserted and read back, even if the characterset of the database is defined as 7 bits. This happens because a byte is 8 bit and Oracle is ALWAYS using 8 bits even with a 7-bit characterset. In a correct setup the Most Significant Bit is just not used and only 7 bits are taken into account.

For one reason or another you need to insert from the Unix server. When you select from tables where data is inserted by the Windows clients you get a '? '(LATIN CAPITAL LETTER O WITH TILDE) for the 'e' instead of the 'e'.

If you insert 'e' on the Unix server and you select the row inserted on the Unix at the Windows client you get an '? '(LATIN CAPITAL LETTER A WITH RING ABOVE) back.

The thing is that you have **INCORRECT** data in the database. You store the numeric value for 'e' of the WIN1252 characterset in the database but you tell Oracle this is US7ASCII data, so Oracle is **NOT** converting anything and just stores the numeric value (again: Oracle thinks that the client is giving US7ASCII codes because the **NLS_LANG** is set to US7ASCII, and the database characterset is also US7ASCII -> no conversion done).

When you select the same column back on the Unix server, oracle is again thinking that the value is correct (Oracle is thinking that the terminal understands US7ASCII) and passes the value to the Unix terminal without any conversion.
Now the problem is that in the WIN1252 characterset the 'e' has the hexadecimal value 'E9' and in the Roman8 characterset the hexadecimal value for 'e' is 'C5'.

Oracle just passes the value stored in the database ('E9') to the Unix terminal, and the Unix terminal thinks this is the letter '?' because in its (Roman8) characterset the hexadecimal value 'E9' is representing the letter '?'. So instead of the 'e' you get '?' on the Unix terminal screen.

The inverse (the insert on the Unix and the select on the Windows client) is just the same story, but you get other results.

The solution is creating database with a characterset that contains 'e' (WE8MSWIN1252, WE8ISO8859P1, UTF-8, etc.) and setting the NLS_LANG on client to WE8MSWIN1252 and on the server to WE8ROMAN8. If you then insert an 'e' on both sides, you will get an 'e' back regardless of where you select them. Oracle knows then that a hexadecimal value of 'C5' inserted by the Unix and a 'E9' from a MSWIN1252 client are both 'e' and inserts 'e' into the database (the code in the database depends on the character set you have chosen).

The same problem appears if you add some Windows clients who are using another character set and have an incorrect NLS_LANG set. You don't have to switch between Unix, mainframe and Windows clients to run into this kind of problem.

To see the list with all supported character sets in Oracle9i Database (European Languages only), go to Appendix A: List of all supported CharacterSets for European Languages in Oracle9i

2.2. Where is the Character Conversion Done?

Normally the conversion is done at client side for performance reasons. This is true from Version 8.0.4 onwards.

If the database is using a character set not known by the client then the conversion is done at server side. This is true from Version 8.1.6 onwards.

2.3. The correct NLS_LANG for my Windows ANSI Code Page

2.3.1. Determine your Windows ANSI code page

The ACP (Ansi Code Page) is defined by the "default locale" setting of windows, so if you have a UK Windows 2000 client and you want to input Cyrillic (Russian) you need to change the ACP (by changing the "default locale") in order to be able to input Russian. See Note 199926.1 How to change the ANSI Code Page (ACP) on Windows.

You'll find its value in the registry:

Start -> Run...
Type "regedit", and click "ok"
Browse the following registry entry:
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Nls\CodePage\ There you have (all the way below) an entry with as name ACP
The value of ACP is your current GUI Codepage, see the table in point 3.2 for the mapping to the oracle name.

If you need to change the "ACP" please see:
Note 199926.1 How to change the ANSI Code Page (ACP) on Windows
Do NOT simply change it in the registry.

Additionally, the following URL provides a list of the default code pages for all Windows versions:
OEM = the command line codepage, ANSI = the GUI codepage

- This Microsoft web site:
http://www.microsoft.com/globaldev/reference/oslocversion.mspx
Provides a list of the default code pages for all Windows versions.

See the table with code pages for Windows 2000:

<table>
<thead>
<tr>
<th>Language Pro</th>
<th>LCID</th>
<th>Code Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>hex</td>
<td>dec</td>
</tr>
<tr>
<td>Arabic</td>
<td>0x0401</td>
<td>1025</td>
</tr>
<tr>
<td>Brazilian</td>
<td>0x0416</td>
<td>1046</td>
</tr>
<tr>
<td>Chinese (Simplified)</td>
<td>0x0804</td>
<td>2052</td>
</tr>
<tr>
<td>Chinese (Traditional)</td>
<td>0x0404</td>
<td>1028</td>
</tr>
<tr>
<td>Chinese (Hong Kong)</td>
<td>0x0c04</td>
<td>3076</td>
</tr>
<tr>
<td>Czech</td>
<td>0x0405</td>
<td>1029</td>
</tr>
<tr>
<td>Danish</td>
<td>0x0406</td>
<td>1030</td>
</tr>
<tr>
<td>Dutch</td>
<td>0x0413</td>
<td>1043</td>
</tr>
<tr>
<td>English</td>
<td>0x0409</td>
<td>1033</td>
</tr>
<tr>
<td>Finnish</td>
<td>0x040B</td>
<td>1035</td>
</tr>
<tr>
<td>French</td>
<td>0x040C</td>
<td>1036</td>
</tr>
<tr>
<td>German</td>
<td>0x0407</td>
<td>1031</td>
</tr>
<tr>
<td>Greek</td>
<td>0x0408</td>
<td>1032</td>
</tr>
<tr>
<td>Hebrew</td>
<td>0x040D</td>
<td>1037</td>
</tr>
<tr>
<td>Hungarian</td>
<td>0x040E</td>
<td>1038</td>
</tr>
<tr>
<td>Italian</td>
<td>0x0410</td>
<td>1040</td>
</tr>
<tr>
<td>Japanese</td>
<td>0x0411</td>
<td>1041</td>
</tr>
<tr>
<td>Korean</td>
<td>0x0412</td>
<td>1042</td>
</tr>
<tr>
<td>Norwegian</td>
<td>0x0414</td>
<td>1044</td>
</tr>
<tr>
<td>Polish</td>
<td>0x0415</td>
<td>1045</td>
</tr>
<tr>
<td>Portuguese</td>
<td>0x0816</td>
<td>2070</td>
</tr>
<tr>
<td>Russian</td>
<td>0x0419</td>
<td>1049</td>
</tr>
<tr>
<td>Spanish</td>
<td>0x0c0A</td>
<td>3082</td>
</tr>
<tr>
<td>Swedish</td>
<td>0x041D</td>
<td>1053</td>
</tr>
<tr>
<td>Turkish</td>
<td>0x041F</td>
<td>1055</td>
</tr>
</tbody>
</table>

2.3.2. Find the correspondent Oracle client character set

Find the Oracle client character set in the table below based on the ACP you found in point 2.3.1. **Note** that there is only ONE CORRECT value for a given ACP.
ANSI CodePage (ACP) | Oracle Client character set (3rd part of NLS_LANG)
--- | ---
1250 | EE8MSWIN1250
1251 | CL8MSWIN1251
1252 | WE8MSWIN1252
1253 | EL8MSWIN1253
1254 | TR8MSWIN1254
1255 | IW8MSWIN1255
1256 | AR8MSWIN1256
1257 | BLT8MSWIN1257
1258 | VN8MSWIN1258
874 | TH8TISASCII
932 | JA16SJIS
936 | ZHS16GBK
949 | KO16MSWIN949
950 | ZHT16MSWIN950 - except for Hong Kong (see below)

This is the character set used by the GUI sqlplus (sqlplusW.exe) that you start through the windows start menu. Please *DO* make the difference between the **GUI SQL*Plus** and the "**DOS mode**" SQL*Plus.

You can use UTF8 as Oracle client character set (=NLS_LANG) on Windows NT, 2000 and XP but you will be limited to use only client programs that explicitly support this configuration.

This is because the user interface of Win32 is not UTF8, therefore the client *program* have to perform explicit conversions between UTF8 (used on Oracle side) and UTF16 (used on Win32 side).

From the other side, programs relying on ANSI Win32 API, like SQL*Plus, older Oracle Forms, etc. cannot work with an NLS_LANG set to UTF8.

### 2.4. The correct NLS_LANG for my DOS / Command Prompt OEM Code Page

MS-DOS mode uses, with a few exceptions like CJK, a different code page (called OEM code page) than Windows GUI (ANSI code page). Meaning that before using an Oracle command line tool such as SQL*Plus (sqlplus.exe) in a command prompt then you need to MANUALLY SET the NLS_LANG parameter as an environment variable with the set DOS command BEFORE using the tool.

You need to set it in order to overwrite the NLS_LANG registry key already matching the ANSI code page. The new "MS-DOS dedicated" NLS_LANG needs to match the MS-DOS OEM code page that could be retrieved by typing `chcp` in a Command Prompt:

```
C:\> chcp
Active code page: 437
C:\> set NLS_LANG=AMERICAN_AMERICA.US8PC437
```

If the NLS_LANG parameter for the MS-DOS mode session is not set appropriately, error messages and data can be corrupted due to incorrect character set conversion.

Use the following list to find the Oracle character set that fits to your MS-DOS code page in use on your locale system:

<table>
<thead>
<tr>
<th>MS-DOS code page</th>
<th>Oracle Client character set (3rd part of NLS_LANG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>437</td>
<td>US8PC437</td>
</tr>
<tr>
<td>737</td>
<td>EL8PC737</td>
</tr>
<tr>
<td>850</td>
<td>WE8PC850</td>
</tr>
<tr>
<td>852</td>
<td>EE8PC852</td>
</tr>
<tr>
<td>857</td>
<td>TR8PC857</td>
</tr>
<tr>
<td>858</td>
<td>WE8PC858</td>
</tr>
</tbody>
</table>
For tools like SQL*Loader you need to set the NLS_LANG to the character set of the FILE you loading.

### 2.5. List of common NLS_LANG's used in the Windows Registry

**Note:** this is the correct setting for the GUI SQL*Plus version, (sqlplusW.exe).
If you are testing with "special" characters please **DO** use the GUI and not the "DOS Box" sqlplus.exe!

<table>
<thead>
<tr>
<th>Operating System Locale</th>
<th>NLS_LANG Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic (U.A.E.)</td>
<td>ARABICUNITED ARAB EMIRATES.AR8MSWIN1256</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>BULGARIAN_BULGARIA.CL8MSWIN1251</td>
</tr>
<tr>
<td>Catalan</td>
<td>CATALAN_CATALONIA.WE8MSWIN1252</td>
</tr>
<tr>
<td>Chinese (PRC)</td>
<td>SIMPLIFIED CHINESE_CHINA.ZHS16GBK</td>
</tr>
<tr>
<td>Chinese (Taiwan)</td>
<td>TRADITIONAL CHINESE_TAIWAN.ZHT16MSWIN950</td>
</tr>
<tr>
<td>Chinese (Hong Kong HKCS)</td>
<td>TRADITIONAL CHINESE_HONG KONG.ZHT16HKSCS</td>
</tr>
<tr>
<td>Chinese (Hong Kong HKCS2001)</td>
<td>TRADITIONAL CHINESE_HONG KONG.ZHT16HKSCS2001 (new in 10gR1)</td>
</tr>
<tr>
<td>Croatian</td>
<td>CROATIAN_CROATIA.EE8MSWIN1250</td>
</tr>
<tr>
<td>Czech</td>
<td>CZECH_CZECH REPUBLIC.EE8MSWIN1250</td>
</tr>
<tr>
<td>Danish</td>
<td>DANISH_DENMARK.WE8MSWIN1252</td>
</tr>
<tr>
<td>Dutch (Netherlands)</td>
<td>DUTCH_THE NETHERLANDS.WE8MSWIN1252</td>
</tr>
<tr>
<td>Dutch (Belgium)</td>
<td>DUTCH_BELGIUM.WE8MSWIN1252</td>
</tr>
<tr>
<td>English (United Kingdom)</td>
<td>ENGLISH_UNITED KINGDOM.WE8MSWIN1252</td>
</tr>
<tr>
<td>English (United States)</td>
<td>AMERICAN_AMERICA.WE8MSWIN1252</td>
</tr>
<tr>
<td>Estonian</td>
<td>ESTONIAN_ESTONIA.BLT8MSWIN1257</td>
</tr>
<tr>
<td>Finnish</td>
<td>FINNISH_FINLAND.WE8MSWIN1252</td>
</tr>
<tr>
<td>French (Canada)</td>
<td>CANADIAN FRENCH_CANADA.WE8MSWIN1252</td>
</tr>
<tr>
<td>French (France)</td>
<td>FRENCH_FRANCE.WE8MSWIN1252</td>
</tr>
<tr>
<td>German (Germany)</td>
<td>GERMAN_GERMANY.WE8MSWIN1252</td>
</tr>
<tr>
<td>Greek</td>
<td>GREEK_GREECE.EL8MSWIN1253</td>
</tr>
<tr>
<td>Hebrew</td>
<td>HEBREW_ISRAEL.IW8MSWIN1255</td>
</tr>
<tr>
<td>Hungarian</td>
<td>HUNGARIAN_HUNGARY.EE8MSWIN1250</td>
</tr>
<tr>
<td>Icelandic</td>
<td>ICELANDIC_ICELAND.WE8MSWIN1252</td>
</tr>
<tr>
<td>Indonesian</td>
<td>INDONESIAN_INDONESIA.WE8MSWIN1252</td>
</tr>
<tr>
<td>Italian (Italy)</td>
<td>ITALIAN_ITALY.WE8MSWIN1252</td>
</tr>
<tr>
<td>Japanese</td>
<td>JAPANESE_JAPAN.JA16SJIS</td>
</tr>
<tr>
<td>Korean</td>
<td>KOREAN_KOREA.KO16MSWIN949</td>
</tr>
<tr>
<td>Latvian</td>
<td>LATVIAN_LATVIA.BLT8MSWIN1257</td>
</tr>
<tr>
<td>Lithuanian</td>
<td>LITHUANIAN_LITHUANIA.BLT8MSWIN1257</td>
</tr>
<tr>
<td>Norwegian</td>
<td>NORWEGIAN_NORWAY.WE8MSWIN1252</td>
</tr>
<tr>
<td>Polish</td>
<td>POLISH_POLAND.EE8MSWIN1250</td>
</tr>
<tr>
<td>Portuguese (Brazil)</td>
<td>BRAZILIAN PORTUGUESE_BRAZIL.WE8MSWIN1252</td>
</tr>
<tr>
<td>Portuguese (Portugal)</td>
<td>PORTUGUESE_PORTUGAL.WE8MSWIN1252</td>
</tr>
<tr>
<td>Romanian</td>
<td>ROMANIAN_ROMANIA.EE8MSWIN1250</td>
</tr>
<tr>
<td>Russian</td>
<td>RUSSIAN_CIS.CL8MSWIN1251</td>
</tr>
<tr>
<td>Slovak</td>
<td>SLOVAK_SLOVAKIA.EE8MSWIN1250</td>
</tr>
<tr>
<td>Spanish (Spain)</td>
<td>SPANISH_SPAIN.WE8MSWIN1252</td>
</tr>
<tr>
<td>Swedish</td>
<td>SWEDISH_SWEDEN.WE8MSWIN1252</td>
</tr>
<tr>
<td>Thai</td>
<td>THAI_THAILAND.TH8TISASCII</td>
</tr>
</tbody>
</table>
2.6. List of common NLS_LANG's used in the Command Prompt (DOS box)

**Note:** this is the correct setting for the **DOS Box** SQL*Plus version (sqlplus.exe)

<table>
<thead>
<tr>
<th>Operating System Locale</th>
<th>Oracle Client character set (3rd part of NLS_LANG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>AR8ASMO8X</td>
</tr>
<tr>
<td>Catalan</td>
<td>WE8PC850</td>
</tr>
<tr>
<td>Chinese (PRC)</td>
<td>ZHS16GBK</td>
</tr>
<tr>
<td>Chinese (Taiwan)</td>
<td>ZHT16MSWIN950</td>
</tr>
<tr>
<td>Czech</td>
<td>EE8PC852</td>
</tr>
<tr>
<td>Danish</td>
<td>WE8PC850</td>
</tr>
<tr>
<td>Dutch</td>
<td>WE8PC850</td>
</tr>
<tr>
<td>English (United Kingdom)</td>
<td>WE8PC850</td>
</tr>
<tr>
<td>English (United States)</td>
<td>US8PC437</td>
</tr>
<tr>
<td>Finnish</td>
<td>WE8PC850</td>
</tr>
<tr>
<td>French</td>
<td>WE8PC850</td>
</tr>
<tr>
<td>German</td>
<td>WE8PC850</td>
</tr>
<tr>
<td>Greek</td>
<td>EL8PC737</td>
</tr>
<tr>
<td>Hebrew</td>
<td>IW8PC1507</td>
</tr>
<tr>
<td>Hungarian</td>
<td>EE8PC852</td>
</tr>
<tr>
<td>Italian</td>
<td>WE8PC850</td>
</tr>
<tr>
<td>Japanese</td>
<td>JA16SJS</td>
</tr>
<tr>
<td>Korean</td>
<td>KO16MSWIN949</td>
</tr>
<tr>
<td>Norwegian</td>
<td>WE8PC850</td>
</tr>
<tr>
<td>Polish</td>
<td>EE8PC852</td>
</tr>
<tr>
<td>Portuguese</td>
<td>WE8PC850</td>
</tr>
<tr>
<td>Romanian</td>
<td>EE8PC852</td>
</tr>
<tr>
<td>Russian</td>
<td>RU8PC866</td>
</tr>
<tr>
<td>Slovak</td>
<td>EE8PC852</td>
</tr>
<tr>
<td>Slovenian</td>
<td>EE8PC852</td>
</tr>
<tr>
<td>Spanish</td>
<td>WE8PC850</td>
</tr>
<tr>
<td>Swedish</td>
<td>WE8PC850</td>
</tr>
<tr>
<td>Turkish</td>
<td>TR8PC857</td>
</tr>
</tbody>
</table>

2.7. How Windows uses Fonts to display the different character sets

We assume you have an UTF8 database with correctly stored UTF8 codepoints.

On Windows there are two kinds of tools / applications:

1) A fully Unicode enabled applications which accepts Unicode codepoints and which can render them. It's the application that needs to deal with the Unicode, Windows provides the Unicode API but the **GUI** system itself is **NOT** Unicode "by nature".

A fully Unicode application can only show one glyph for a given Unicode code point. So there is **NO** confusion possible here, this application will need to use a full Unicode font. If you have a full Unicode application, then you need to set the NLS_LANG to UTF8.
Windows NLS Considerations

Note that there are currently NOT many applications like this and if it's not explicitly mentioned by the vendor it's most likely an ANSI application (see below). So DON'T set the NLS_LANG to UTF8 if you are not sure!

The only Unicode capable client that is included in the database is iSQL*PLUS. See Note 231231.1 Quick setup of iSQL*Plus as Unicode client on windows, for a guide on how to setup this.

2) A standard ANSI application (like sqlplusw.exe) cannot use Unicode code points. So the Unicode code point stored in the database needs to be CONVERTED to an ANSI code point. This is done by setting NLS_LANG (as described in further on in this note and in Note 158577.1. This allows oracle to map the Unicode point to the character set of the client, (and here comes the tricky part) but this is NOT the same as a font.

If you want to display Arabic for example then you need to set the Windows character set to Arabic. That way Windows knows what are valid code points and can use the FONT engine to DISPLAY the code points (this results in glyphs). Windows passes the code point and the "page" to the rendering engine. This "page" defines the glyphs for the code points for a certain character set/code page.

Because there are only 256 possible positions for an ANSI application, and one font contains normally glyphs for different languages this "page" is used to select from a FONT that has (for example) all the glyphs for Cyrillic, Arabic and West-European the "page" for Arabian.

So lets say you have an Arabic setup that works, you change manually the "Page" of a FONT and ask to display the glyph for ANSI code point XX. Now 1 of 2 things can happen:

1) There is a character defined on that position for the CHARACTERSET of that "Page", so the creator of the font has foreseen a glyph and this is displayed (but this is NOT the character expected or wanted as its stored as a different character in the database!).

2) There is NO character defined on that position for the CHARACTERSET of that "Page" so the creator of the font has NOT foreseen a glyph and you get "garbage" or black squares (normally you should see a black square but a ? or ? are also possible, this depends on the error handling defined in the FONT).

The above is also possible if you have a non-Unicode character set for the database.

2.8. What about Multiple Homes on Windows?

There is nothing special with NLS_LANG and the multiple homes on Windows. The parameter taken into account is the one specified in the ORACLE_HOME registry key used by the executable. Again, if set in the environment, it takes precedence over the value in the registry and is used for ALL Oracle Homes on the server/client (!).

The NLS_LANG can be found in these registry keys:
HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE
or
HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\HOMEx

3. Advanced NLS Research

3.1. What is Locale Builder?

The Locale Builder offers an easy and efficient way to access and define NLS locale data definitions. It provides a graphical user interface through which you can easily view, modify, and define locale-specific data. It extracts data from the text and binary definition files and presents them in a readable format, so you can process the information without worrying about the specific definition formats used in these files.
The Locale Builder handles four types of locale definitions: language, territory, character set, and linguistic sort (Collation). It also supports user-defined characters and customized linguistic rules. You can view definitions in existing text and binary definition files and make changes to them or create your own definitions.

Locale Builder gives you the ability to build your own locale (Language, Territory, Character Set and Collation) to be used within Oracle server. This ability does not include creating your own message files. It can be used to modify a large number of locales provided with Oracle9i like Arabic, Dutch, etc.

For more details, see Metalink Note: 227339.1 - Locale Builder - Frequently Asked Questions and Metalink Note: 223706.1 - Using Locale Builder to view the definition of character sets

3.2. How to see what's really stored in the database?

To find the real numeric value for a character stored in the database is to use the dump command:

**Introduction**

This note looks at the sql DUMP function and especially it's usage in debugging NLS problems. The first part of the note focuses on using DUMP on (VAR)CHAR columns. In 2 sections at the end of the note we will also discuss how to use DUMP on DATE and NUMBER columns, and how to interpret the output of DUMP on those columns.

**Character storage**

Character data stored in Oracle is stored according to a certain character set. A character set is nothing more than a numbered list of characters. Every character is assigned a number, or "code point". When a character is inserted into the database it is that codepoint that is actually stored. You can use the Oracle Locale Builder to 'view' the different character sets and see the codepoints for the different characters in these character sets: Note 223706.1 Using Locale Builder to view the definition of character sets

**Character conversion through NLS_LANG**

Character set conversion is automatically performed in Oracle when the client environment variable NLS_LANG is set to be different from the database character set. It should be set to accurately reflect the client's character encoding scheme. For details about that please see: Note 158577.1 NLS_LANG Explained (How does Client-Server Character Conversion Work?)

**Debugging NLS Issues with the DUMP sql command**

Sometimes the data that is inserted into the database is not what comes out when you select it back. There are generally 3 possible causes of this:
1. Environment settings (or other problems) when inserting
2. The database character set does not support the characters that are inserted
3. Environment settings (or other problems) when querying

Using Locale Builder (as mentioned earlier) we will know if the character that we want to store is supported by the database character set, and we will know what the codepoint is of the data we want to store. Using the SQL dump command we can check the actual code points of the data stored in the database. Therefore we can determine if the problem lies in the insert stage or the query stage.

**Basic DUMP usage for character data**

Assume a database is created with a WE8DEC character set and some some characters that contain a umlaut stored in there. You can check the result of your insert with

SQL> SELECT DUMP(col) FROM table;
you should get the following values with this dump:

- `<u umlaut>` -> 252
- `<o umlaut>` -> 246
- `<a umlaut>` -> 228

These are the code points of above characters in WE8DEC encoding scheme.

If the character set of the database would have been WE8PC850, and these same characters would have been stored, they would have been stored under different code points:

- `<u umlaut>` -> 129
- `<o umlaut>` -> 148
- `<a umlaut>` -> 132

You can use the DUMP command to show the values in either the decimal or HEX format. Many lists of code points use HEX so it's useful to select the HEX code points that are stored. So if we take the same 3 characters, but this time in a AL32UTF8 database we can select the HEX codes like this:

```sql
SQL> SELECT DUMP(col, 16) FROM table;
```

This would show the characters stored as:

- `<u umlaut>` -> 0xC3BC
- `<o umlaut>` -> 0xC3B6
- `<a umlaut>` -> 0xC3A4

### Advanced DUMP usage for character data

Character conversion can be tested using the SQL 'dump', 'chr' and 'convert' functions by specifying byte values as defined by the character set encoding scheme. This testing technique has the advantage that a terminal configured for a given encoding scheme is not necessary, for example:

```sql
SQL> select dump(convert(chr(231),'<target_char_set>','<source_char_set>')) from dual;
```

This technique can also be used to test case conversion, for example:

```sql
SQL> select dump(upper(chr(231))) from dual;
```

and to directly insert and select character data, independently of character set conversion, for example:

```sql
SQL> insert into <table> values (chr(231)||chr(232)||chr(233));
SQL> select dump(<column>) from <tab>;
```

For full details see: Metalink Note 13854.1

### 3.3. How to define NLS_LANG with OEM (enterprise manager) version 2 and 9?

#### Symptom(s)

The Enterprise manager client tools are not handling extended ASCII characters correctly. For example SQL Worksheet might not handle the Euro-sign properly, etc. etc.

+ You have set all the client settings (NLS_LANG, dbappscfg.properties file).

---

Document : Windows NLS Considerations – version 2.1

Page 14 of 21

Issue date 21.06.2005

---
Connections directly to the target database work ok. Only when connected through the OMS this problem persists.

Change(s)

None, this problem would occur when trying to set up the EM tools to connect to the target database through an OMS.

Cause

This problem is caused by the fact that if you run EM tools through an OMS, the connections to the database are made from the OMS environment. So you need to set up the OMS environment to be able to support these characters.

Fix

Because the sessions to the database are created on the OMS machine and not on the client it is important to have the correct settings on the OMS server. That means we have to think about 3 things:

1. The NLS_LANG
As in every other setup, the oracle environment variable NLS_LANG needs to be set up to reflect the OS environment. So, if we extend the example from point 1, the NLS_LANG should now be set to:

   NLS_LANG=FRENCH_FRANCE.WE8ISO8859P15

   For more information on setting NLS_LANG see:

2. The dbappscfg.properties file.
This file is used to set certain variables for the EM tools. Specifically for sqlplus worksheet we need to make sure that the SQLPLUS_NLS_LANG variable is set here. Again, like in the previous points, we need to set this to reflect the situation ON THIS SERVER and not on the client. So in the file on the server $ORACLE_HOME/sysman/admin set:

   SQLPLUS_NLS_LANG=FRENCH_FRANCE.WE8ISO8859P15

   After these settings have been made you should stop and restart the OMS and the client tools will now handle the extended ASCII characters properly.

Specific cases

The choice of OS locale is very important and can have a serious affect on the characters that are and aren’t displayed properly.

4. Appendixes

Appendix A: List of all supported CharacterSets for European Languages in Oracle9i

Oracle9i Globalization Support Guide, Appendix A, Locale Data

European Language Character Sets

Table A-5 lists the Oracle character sets that can support European languages.

Table A-5 European Language Character Sets

---

Document : Windows NLS Considerations – Page 15 of 21
version 2.1  Issue date 21.06.2005
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>US7ASCII</td>
<td>ASCII 7-bit American</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>SF7ASCII</td>
<td>ASCII 7-bit Finnish</td>
<td>SB</td>
</tr>
<tr>
<td>YUG7ASCII</td>
<td>ASCII 7-bit Yugoslavian</td>
<td>SB</td>
</tr>
<tr>
<td>RU8BESTA</td>
<td>BESTA 8-bit Latin/Cyrillic</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>EL8GCOS7</td>
<td>Bull EBCDIC GCOS7 8-bit Greek</td>
<td>SB</td>
</tr>
<tr>
<td>WE8GCOS7</td>
<td>Bull EBCDIC GCOS7 8-bit West European</td>
<td>SB</td>
</tr>
<tr>
<td>EL8DEC</td>
<td>DEC 8-bit Latin/Greek</td>
<td>SB</td>
</tr>
<tr>
<td>TR7DEC</td>
<td>DEC VT100 7-bit Turkish</td>
<td>SB</td>
</tr>
<tr>
<td>TR8DEC</td>
<td>DEC 8-bit Turkish</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>TR8EBCDIC1026</td>
<td>EBCDIC Code Page 1026 8-bit Turkish</td>
<td>SB</td>
</tr>
<tr>
<td>TR8EBCDIC1026S</td>
<td>EBCDIC Code Page 1026 Server 8-bit Turkish</td>
<td>SB</td>
</tr>
<tr>
<td>TR8PC857</td>
<td>IBM-PC Code Page 857 8-bit Turkish</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>TR8MAC TURKISH</td>
<td>MAC Client 8-bit Turkish</td>
<td>SB</td>
</tr>
<tr>
<td>TR8MAC TURKISHS</td>
<td>MAC Server 8-bit Turkish</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>TR8MSWIN1254</td>
<td>MS Windows Code Page 1254 8-bit Turkish</td>
<td>SB, ASCII, EURO</td>
</tr>
<tr>
<td>WE8BS2000L5</td>
<td>Siemens EBCDIC.DF.L5 8-bit West European/Turkish</td>
<td>SB</td>
</tr>
<tr>
<td>WE8DEC</td>
<td>DEC 8-bit West European</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>D7DEC</td>
<td>DEC VT100 7-bit German</td>
<td>SB</td>
</tr>
<tr>
<td>F7DEC</td>
<td>DEC VT100 7-bit French</td>
<td>SB</td>
</tr>
<tr>
<td>S7DEC</td>
<td>DEC VT100 7-bit Swedish</td>
<td>SB</td>
</tr>
<tr>
<td>E7DEC</td>
<td>DEC VT100 7-bit Spanish</td>
<td>SB</td>
</tr>
<tr>
<td>NDK7DEC</td>
<td>DEC VT100 7-bit Norwegian/Danish</td>
<td>SB</td>
</tr>
<tr>
<td>I7DEC</td>
<td>DEC VT100 7-bit Italian</td>
<td>SB</td>
</tr>
<tr>
<td>NL7DEC</td>
<td>DEC VT100 7-bit Dutch</td>
<td>SB</td>
</tr>
<tr>
<td>CH7DEC</td>
<td>DEC VT100 7-bit Swiss (German/French)</td>
<td>SB</td>
</tr>
<tr>
<td>SF7DEC</td>
<td>DEC VT100 7-bit Finnish</td>
<td>SB</td>
</tr>
<tr>
<td>WE8DG</td>
<td>DG 8-bit West European</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>WE8EBCDIC37C</td>
<td>EBCDIC Code Page 37 8-bit Oracle/c</td>
<td>SB</td>
</tr>
<tr>
<td>WE8EBCDIC37</td>
<td>EBCDIC Code Page 37 8-bit West European</td>
<td>SB</td>
</tr>
<tr>
<td>D8EBCDIC273</td>
<td>EBCDIC Code Page 273/1 8-bit Austrian German</td>
<td>SB</td>
</tr>
<tr>
<td>DK8EBCDIC277</td>
<td>EBCDIC Code Page 277/1 8-bit Danish</td>
<td>SB</td>
</tr>
<tr>
<td>S8EBCDIC278</td>
<td>EBCDIC Code Page 278/1 8-bit Swedish</td>
<td>SB</td>
</tr>
<tr>
<td>I8EBCDIC280</td>
<td>EBCDIC Code Page 280/1 8-bit Italian</td>
<td>SB</td>
</tr>
<tr>
<td>WE8EBCDIC284</td>
<td>EBCDIC Code Page 284 8-bit Latin American/Spanish</td>
<td>SB</td>
</tr>
<tr>
<td>WE8EBCDIC285</td>
<td>EBCDIC Code Page 285 8-bit West European</td>
<td>SB</td>
</tr>
<tr>
<td>WE8EBCDIC924</td>
<td>Latin 9 EBCDIC 924</td>
<td>SB, EBCDIC</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Comments</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>WE8EBCDIC1047</td>
<td>EBCDIC Code Page 1047 8-bit West European SB</td>
<td></td>
</tr>
<tr>
<td>WE8EBCDIC1047E</td>
<td>Latin 1/Open Systems 1047 SB, EBCDIC, EURO</td>
<td></td>
</tr>
<tr>
<td>WE8EBCDIC1140</td>
<td>EBCDIC Code Page 1140 8-bit West European SB, EURO</td>
<td></td>
</tr>
<tr>
<td>WE8EBCDIC1140C</td>
<td>EBCDIC Code Page 1140 Client 8-bit West European SB, EURO</td>
<td></td>
</tr>
<tr>
<td>WE8EBCDIC1145</td>
<td>EBCDIC Code Page 1145 8-bit West European SB, EURO</td>
<td></td>
</tr>
<tr>
<td>WE8EBCDIC1146</td>
<td>EBCDIC Code Page 1146 8-bit West European SB, EURO</td>
<td></td>
</tr>
<tr>
<td>WE8EBCDIC1148</td>
<td>EBCDIC Code Page 1148 8-bit West European SB, EURO</td>
<td></td>
</tr>
<tr>
<td>WE8EBCDIC1148C</td>
<td>EBCDIC Code Page 1148 Client 8-bit West European SB, EURO</td>
<td></td>
</tr>
<tr>
<td>F8EBCDIC297</td>
<td>EBCDIC Code Page 297 8-bit French SB</td>
<td></td>
</tr>
<tr>
<td>WE8EBCDIC500C</td>
<td>EBCDIC Code Page 500 8-bit Oracle/c SB</td>
<td></td>
</tr>
<tr>
<td>WE8EBCDIC500</td>
<td>EBCDIC Code Page 500 8-bit West European SB</td>
<td></td>
</tr>
<tr>
<td>EE8EBCDIC870</td>
<td>EBCDIC Code Page 870 8-bit East European SB</td>
<td></td>
</tr>
<tr>
<td>EE8EBCDIC870C</td>
<td>EBCDIC Code Page 870 Client 8-bit East European SB</td>
<td></td>
</tr>
<tr>
<td>EE8EBCDIC870S</td>
<td>EBCDIC Code Page 870 Server 8-bit East European SB</td>
<td></td>
</tr>
<tr>
<td>WE8EBCDIC871</td>
<td>EBCDIC Code Page 871 8-bit Icelandic SB</td>
<td></td>
</tr>
<tr>
<td>EL8EBCDIC875</td>
<td>EBCDIC Code Page 875 8-bit Greek SB</td>
<td></td>
</tr>
<tr>
<td>EL8EBCDIC875R</td>
<td>EBCDIC Code Page 875 Server 8-bit Greek SB</td>
<td></td>
</tr>
<tr>
<td>CL8EBCDIC1025</td>
<td>EBCDIC Code Page 1025 8-bit Cyrillic SB</td>
<td></td>
</tr>
<tr>
<td>CL8EBCDIC1025C</td>
<td>EBCDIC Code Page 1025 Client 8-bit Cyrillic SB</td>
<td></td>
</tr>
<tr>
<td>CL8EBCDIC1025R</td>
<td>EBCDIC Code Page 1025 Server 8-bit Cyrillic SB</td>
<td></td>
</tr>
<tr>
<td>CL8EBCDIC1025S</td>
<td>EBCDIC Code Page 1025 Server 8-bit Cyrillic SB</td>
<td></td>
</tr>
<tr>
<td>BLT8EBCDIC1112</td>
<td>EBCDIC Code Page 1112 8-bit Baltic Multilingual SB</td>
<td></td>
</tr>
<tr>
<td>BLT8EBCDIC1112S</td>
<td>EBCDIC Code Page 1112 8-bit Server Baltic Multilingual SB</td>
<td></td>
</tr>
<tr>
<td>D8EBCDIC1141</td>
<td>EBCDIC Code Page 1141 8-bit Austrian German SB, EURO</td>
<td></td>
</tr>
<tr>
<td>DK8EBCDIC1142</td>
<td>EBCDIC Code Page 1142 8-bit Danish SB, EURO</td>
<td></td>
</tr>
<tr>
<td>S8EBCDIC1143</td>
<td>EBCDIC Code Page 1143 8-bit Swedish SB, EURO</td>
<td></td>
</tr>
<tr>
<td>I8EBCDIC1144</td>
<td>EBCDIC Code Page 1144 8-bit Italian SB, EURO</td>
<td></td>
</tr>
<tr>
<td>F8EBCDIC1147</td>
<td>EBCDIC Code Page 1147 8-bit French SB, EURO</td>
<td></td>
</tr>
<tr>
<td>EEC8EUROASCI</td>
<td>EEC Targon 35 ASCII West European/Greek SB</td>
<td></td>
</tr>
<tr>
<td>EEC8EUROPA3</td>
<td>EEC EUROPA3 8-bit West European/Greek SB</td>
<td></td>
</tr>
<tr>
<td>LA8PASSPORT</td>
<td>German Government Printer 8-bit All-European Latin SB, ASCII</td>
<td></td>
</tr>
<tr>
<td>WE8HP</td>
<td>HP LaserJet 8-bit West European SB</td>
<td></td>
</tr>
<tr>
<td>WE8ROMAN8</td>
<td>HP Roman8 8-bit West European SB, ASCII</td>
<td></td>
</tr>
<tr>
<td>HU8CW12</td>
<td>Hungarian 8-bit CWI-2 SB, ASCII</td>
<td></td>
</tr>
<tr>
<td>HU8ABMOD</td>
<td>Hungarian 8-bit Special AB Mod SB, ASCII</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Comments</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>LV8RST104090</td>
<td>IBM-PC Alternative Code Page 8-bit Latvian (Latin/Cyrillic)</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>US8PC437</td>
<td>IBM-PC Code Page 437 8-bit American</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>BG8PC437S</td>
<td>IBM-PC Code Page 437 8-bit (Bulgarian Modification)</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>EL8PC437S</td>
<td>IBM-PC Code Page 437 8-bit (Greek modification)</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>EL8PC737</td>
<td>IBM-PC Code Page 737 8-bit Greek/Latin</td>
<td>SB</td>
</tr>
<tr>
<td>LT8PC772</td>
<td>IBM-PC Code Page 772 8-bit Lithuanian (Latin/Cyrillic)</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>LT8PC774</td>
<td>IBM-PC Code Page 774 8-bit Lithuanian (Latin)</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>BLT8PC775</td>
<td>IBM-PC Code Page 775 8-bit Baltic</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>WE8PC850</td>
<td>IBM-PC Code Page 850 8-bit West European</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>EL8PC851</td>
<td>IBM-PC Code Page 851 8-bit Greek/Latin</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>EE8PC852</td>
<td>IBM-PC Code Page 852 8-bit East European</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>RU8PC855</td>
<td>IBM-PC Code Page 855 8-bit Latin/Cyrillic</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>WE8PC858</td>
<td>IBM-PC Code Page 858 8-bit West European</td>
<td>SB, ASCII, EURO</td>
</tr>
<tr>
<td>WE8PC860</td>
<td>IBM-PC Code Page 860 8-bit West European</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>IS8PC861</td>
<td>IBM-PC Code Page 861 8-bit Icelandic</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>CDN8PC863</td>
<td>IBM-PC Code Page 863 8-bit Canadian French</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>N8PC865</td>
<td>IBM-PC Code Page 865 8-bit Norwegian</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>RU8PC866</td>
<td>IBM-PC Code Page 866 8-bit Latin/Cyrillic</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>EL8PC869</td>
<td>IBM-PC Code Page 869 8-bit Greek/Latin</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>LV8PC1117</td>
<td>IBM-PC Code Page 1117 8-bit Latvian</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>US8ICL</td>
<td>ICL EBCDIC 8-bit American</td>
<td>SB</td>
</tr>
<tr>
<td>WE8ICL</td>
<td>ICL EBCDIC 8-bit West European</td>
<td>SB</td>
</tr>
<tr>
<td>WE8ISOICIUK</td>
<td>ICL special version ISO8859-1</td>
<td>SB</td>
</tr>
<tr>
<td>WE8ISO8859P1</td>
<td>ISO 8859-1 West European</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>EE8ISO8859P2</td>
<td>ISO 8859-2 East European</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>SE8ISO8859P3</td>
<td>ISO 8859-3 South European</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>NEE8ISO8859P4</td>
<td>ISO 8859-4 North and North-East European</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>CL8ISO8859P5</td>
<td>ISO 8859-5 Latin/Cyrillic</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>AR8ISO8859P6</td>
<td>ISO 8859-6 Latin/Arabic</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>EL8ISO8859P7</td>
<td>ISO 8859-7 Latin/Greek</td>
<td>SB, ASCII, EURO</td>
</tr>
<tr>
<td>IW8ISO8859P8</td>
<td>ISO 8859-8 Latin/Hebrew</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>NE8ISO8859P10</td>
<td>ISO 8859-10 North European</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>BLT8ISO8859P13</td>
<td>ISO 8859-13 Baltic</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>CEL8ISO8859P14</td>
<td>ISO 8859-13 Celtic</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>WE8ISO8859P15</td>
<td>ISO 8859-15 West European</td>
<td>SB, ASCII, EURO</td>
</tr>
<tr>
<td>LA8ISO6937</td>
<td>ISO 6937 8-bit Coded Character Set for Text Communication</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Comments</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>IW7IS960</td>
<td>Israeli Standard 960 7-bit Latin/Hebrew</td>
<td>SB</td>
</tr>
<tr>
<td>AR8ARABICMAC</td>
<td>Mac Client 8-bit Latin/Arabic</td>
<td>SB</td>
</tr>
<tr>
<td>EE8MACCE</td>
<td>Mac Client 8-bit Central European</td>
<td>SB</td>
</tr>
<tr>
<td>EE8MACROATIAN</td>
<td>Mac Client 8-bit Croatian</td>
<td>SB</td>
</tr>
<tr>
<td>WE8MACROMAN8</td>
<td>Mac Client 8-bit Extended Roman8 West European</td>
<td>SB</td>
</tr>
<tr>
<td>EL8MACGREEK</td>
<td>Mac Client 8-bit Greek</td>
<td>SB</td>
</tr>
<tr>
<td>IS8MACICELANDIC</td>
<td>Mac Client 8-bit Icelandic</td>
<td>SB</td>
</tr>
<tr>
<td>CL8MACCYRILLIC</td>
<td>Mac Client 8-bit Latin/Cyrillic</td>
<td>SB</td>
</tr>
<tr>
<td>AR8ARABICMACS</td>
<td>Mac Server 8-bit Latin/Arabic</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>EE8MACCES</td>
<td>Mac Server 8-bit Central European</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>EE8MACROATIANS</td>
<td>Mac Server 8-bit Croatian</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>WE8MACROMAN8S</td>
<td>Mac Server 8-bit Extended Roman8 West European</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>CL8MACCYRILLICS</td>
<td>Mac Server 8-bit Latin/Cyrillic</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>EL8MACGREEKS</td>
<td>Mac Server 8-bit Greek</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>IS8MACICELANDICS</td>
<td>Mac Server 8-bit Icelandic</td>
<td>SB</td>
</tr>
<tr>
<td>BG8MSWIN</td>
<td>MS Windows 8-bit Bulgarian Cyrillic</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>LT8MSWIN921</td>
<td>MS Windows Code Page 921 8-bit Lithuanian</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>ET8MSWIN923</td>
<td>MS Windows Code Page 923 8-bit Estonian</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>EE8MSWIN1250</td>
<td>MS Windows Code Page 1250 8-bit East European</td>
<td>SB, ASCII, EURO</td>
</tr>
<tr>
<td>CL8MSWIN1251</td>
<td>MS Windows Code Page 1251 8-bit Latin/Cyrillic</td>
<td>SB, ASCII, EURO</td>
</tr>
<tr>
<td>WE8MSWIN1252</td>
<td>MS Windows Code Page 1252 8-bit West European</td>
<td>SB, ASCII, EURO</td>
</tr>
<tr>
<td>EL8MSWIN1253</td>
<td>MS Windows Code Page 1253 8-bit Latin/Greek</td>
<td>SB, ASCII, EURO</td>
</tr>
<tr>
<td>BLT8MSWIN1257</td>
<td>MS Windows Code Page 1257 8-bit Baltic</td>
<td>SB, ASCII, EURO</td>
</tr>
<tr>
<td>BLT8CP921</td>
<td>Latvian Standard LV88-92(1) Windows/Unix 8-bit Baltic</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>LV8PC8LR</td>
<td>Latvian Version IBM-PC Code Page 866 8-bit Latin/Cyrillic</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>WE8NCR4970</td>
<td>NCR 4970 8-bit West European</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>WE8NEXTSTEP</td>
<td>NeXTSTEP PostScript 8-bit West European</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>CL8ISOIR111</td>
<td>ISOIR111 Cyrillic</td>
<td>SB</td>
</tr>
<tr>
<td>CL8KO18R</td>
<td>RELCOM Internet Standard 8-bit Latin/Cyrillic</td>
<td>SB, ASCII</td>
</tr>
<tr>
<td>CL8KO18U</td>
<td>KO18 Ukrainian Cyrillic</td>
<td>SB</td>
</tr>
<tr>
<td>US8BS2000</td>
<td>Siemens 9750-62 EBCDIC 8-bit American</td>
<td>SB</td>
</tr>
<tr>
<td>DK8BS2000</td>
<td>Siemens 9750-62 EBCDIC 8-bit Danish</td>
<td>SB</td>
</tr>
<tr>
<td>F8BS2000</td>
<td>Siemens 9750-62 EBCDIC 8-bit French</td>
<td>SB</td>
</tr>
<tr>
<td>D8BS2000</td>
<td>Siemens 9750-62 EBCDIC 8-bit German</td>
<td>SB</td>
</tr>
<tr>
<td>E8BS2000</td>
<td>Siemens 9750-62 EBCDIC 8-bit Spanish</td>
<td>SB</td>
</tr>
<tr>
<td>S8BS2000</td>
<td>Siemens 9750-62 EBCDIC 8-bit Swedish</td>
<td>SB</td>
</tr>
</tbody>
</table>
## Windows NLS Considerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>DK7SIEMENS9780X</td>
<td>Siemens 97801/97808 7-bit Danish</td>
<td>SB</td>
</tr>
<tr>
<td>F7SIEMENS9780X</td>
<td>Siemens 97801/97808 7-bit French</td>
<td>SB</td>
</tr>
<tr>
<td>D7SIEMENS9780X</td>
<td>Siemens 97801/97808 7-bit German</td>
<td>SB</td>
</tr>
<tr>
<td>I7SIEMENS9780X</td>
<td>Siemens 97801/97808 7-bit Italian</td>
<td>SB</td>
</tr>
<tr>
<td>N7SIEMENS9780X</td>
<td>Siemens 97801/97808 7-bit Norwegian</td>
<td>SB</td>
</tr>
<tr>
<td>E7SIEMENS9780X</td>
<td>Siemens 97801/97808 7-bit Spanish</td>
<td>SB</td>
</tr>
<tr>
<td>S7SIEMENS9780X</td>
<td>Siemens 97801/97808 7-bit Swedish</td>
<td>SB</td>
</tr>
<tr>
<td>EE8BS2000</td>
<td>Siemens EBCDIC.DF.04 8-bit East European</td>
<td>SB</td>
</tr>
<tr>
<td>WE8BS2000</td>
<td>Siemens EBCDIC.DF.04 8-bit West European</td>
<td>SB</td>
</tr>
<tr>
<td>WE8BS2000E</td>
<td>Siemens EBCDIC.DF.04 8-bit West European</td>
<td>SB, EURO</td>
</tr>
<tr>
<td>CL8BS2000</td>
<td>Siemens EBCDIC.EHC.LC 8-bit Cyrillic</td>
<td>SB</td>
</tr>
<tr>
<td>AL16UTF16</td>
<td>See &quot;Universal Character Sets&quot; for details</td>
<td>MB, EURO, FIXED</td>
</tr>
<tr>
<td>AL32UTF8</td>
<td>See &quot;Universal Character Sets&quot; for details</td>
<td>MB, ASCII, EURO</td>
</tr>
<tr>
<td>UTF8</td>
<td>See &quot;Universal Character Sets&quot; for details</td>
<td>MB, ASCII, EURO</td>
</tr>
<tr>
<td>UTFE</td>
<td>See &quot;Universal Character Sets&quot; for details</td>
<td>MB, EURO</td>
</tr>
</tbody>
</table>

### Appendix B: References

2. NLS_LANG Explained (How does Client-Server Character Conversion Work?) - [http://metalink.oracle.com/metalink/plsql/showdoc?db=NOT&id=158577.1&blackframe=1](http://metalink.oracle.com/metalink/plsql/showdoc?db=NOT&id=158577.1&blackframe=1)
9. Good ole' ASCII - [http://czyborra.com/charsets/iso646.html](http://czyborra.com/charsets/iso646.html)