Windows NLS Considerations
version 2.1

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

Contents

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.

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 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.

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:

NLS_LANG = 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:

* 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** assumes 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, WE8ISO89859P1, 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 characterset you have chosen).

The same problem appears if you add some Windows clients who are using another characterset 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 charactersets 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 characterset 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:
URL for Windows 2000: http://www.microsoft.com/globaldev/reference/oslocversion.mspx#win2k
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:

Windows 2000				
Language Pro	LCID		Code Page	
	hex	dec	ANSI	OEM
Arabic	0x0401	1025	1256	720
Brazilian	0x0416	1046	1252	850
Chinese (Simplified)	0x0804	2052	936	936
Chinese (Traditional)	0x0404	1028	950	950
Chinese (Hong Kong)	0x0c04	3076	950	950
Czech	0x0405	1029	1250	852
Danish	0x0406	1030	1252	850
Dutch	0x0413	1043	1252	850
English	0x0409	1033	1252	437
Finnish	0x040B	1035	1252	850
French	0x040C	1036	1252	850
German	0x0407	1031	1252	850
Greek	0x0408	1032	1253	737
Hebrew	0x040D	1037	1255	862
Hungarian	0x040E	1038	1250	852
Italian	0x0410	1040	1252	850
Japanese	0x0411	1041	932	932
Korean	0x0412	1042	949	949
Norwegian	0x0414	1044	1252	850
Polish	0x0415	1045	1250	852
Portuguese	0x0816	2070	1252	850
Russian	0x0419	1049	1251	866
Spanish	0x0C0A	3082	1252	850
Swedish	0x041D	1053	1252	850
Turkish	0x041F	1055	1254	857

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 characterset used by the GUI sqlplus (sqlplusW.exe) that you start trough 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:

MS-DOS code page	Oracle Client character set (3rd part of NLS_LANG)
437	US8PC437
737	EL8PC737
850	WE8PC850
852	EE8PC852
857	TR8PC857
858	WE8PC858

861	IS8PC861
862	IW8PC1507
865	N8PC865
866	RU8PC866

For tools like SQL*Loader you need to set the NLS_LANG to the characterset 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!

Operating System Locale	**NLS_LANG Value**
Arabic (U.A.E.)	ARABIC_UNITED ARAB EMIRATES.AR8MSWIN1256
Bulgarian	BULGARIAN_BULGARIA.CL8MSWIN1251
Catalan	CATALAN_CATALONIA.WE8MSWIN1252
Chinese (PRC)	SIMPLIFIED CHINESE_CHINA.ZHS16GBK
Chinese (Taiwan)	TRADITIONAL CHINESE_TAIWAN.ZHT16MSWIN950
Chinese (Hong Kong HKCS)	TRADITIONAL CHINESE_HONG KONG.ZHT16HKSCS
Chinese (Hong Kong HKCS2001)	TRADITIONAL CHINESE_HONG KONG.ZHT16HKSCS2001 (new in 10gR1) (See http://www.microsoft.com/hk/hkscs/ how Microsoft supports HKCS and http://www.info.gov.hk/digital21/eng/structure/cli_main.html)
Croatian	CROATIAN_CROATIA.EE8MSWIN1250
Czech	CZECH_CZECH REPUBLIC.EE8MSWIN1250
Danish	DANISH_DENMARK.WE8MSWIN1252
Dutch (Netherlands)	DUTCH_THE NETHERLANDS.WE8MSWIN1252
Dutch (Belgium)	DUTCH_BELGIUM.WE8MSWIN1252
English (United Kingdom)	ENGLISH_UNITED KINGDOM.WE8MSWIN1252
English (United States)	AMERICAN_AMERICA.WE8MSWIN1252
Estonian	ESTONIAN_ESTONIA.BLT8MSWIN1257
Finnish	FINNISH_FINLAND.WE8MSWIN1252
French (Canada)	CANADIAN FRENCH_CANADA.WE8MSWIN1252
French (France)	FRENCH_FRANCE.WE8MSWIN1252
German (Germany)	GERMAN_GERMANY.WE8MSWIN1252
Greek	GREEK_GREECE.EL8MSWIN1253
Hebrew	HEBREW_ISRAEL.IW8MSWIN1255
Hungarian	HUNGARIAN_HUNGARY.EE8MSWIN1250
Icelandic	ICELANDIC_ICELAND.WE8MSWIN1252
Indonesian	INDONESIAN_INDONESIA.WE8MSWIN1252
Italian (Italy)	ITALIAN_ITALY.WE8MSWIN1252
Japanese	JAPANESE_JAPAN.JA16SJIS
Korean	KOREAN_KOREA.KO16MSWIN949
Latvian	LATVIAN_LATVIA.BLT8MSWIN1257
Lithuanian	LITHUANIAN_LITHUANIA.BLT8MSWIN1257
Norwegian	NORWEGIAN_NORWAY.WE8MSWIN1252
Polish	POLISH_POLAND.EE8MSWIN1250
Portuguese (Brazil)	BRAZILIAN PORTUGUESE_BRAZIL.WE8MSWIN1252
Portuguese (Portugal)	PORTUGUESE_PORTUGAL.WE8MSWIN1252
Romanian	ROMANIAN_ROMANIA.EE8MSWIN1250
Russian	RUSSIAN_CIS.CL8MSWIN1251
Slovak	SLOVAK_SLOVAKIA.EE8MSWIN1250
Spanish (Spain)	SPANISH_SPAIN.WE8MSWIN1252
Swedish	SWEDISH_SWEDEN.WE8MSWIN1252
Thai	THAI_THAILAND.TH8TISASCII

Spanish (Mexico)	MEXICAN SPANISH_MEXICO.WE8MSWIN1252
Spanish (Venezuela)	LATIN AMERICAN SPANISH_VENEZUELA.WE8MSWIN1252
Turkish	TURKISH_TURKEY.TR8MSWIN1254
Ukrainian	UKRAINIAN_UKRAINE.CL8MSWIN1251
Vietnamese	VIETNAMESE_VIETNAM.VN8MSWIN1258

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)

Operating System Locale	**Oracle Client character set (3rd part of NLS_LANG)**
Arabic	AR8ASMO8X
Catalan	WE8PC850
Chinese (PRC)	ZHS16GBK
Chinese (Taiwan)	ZHT16MSWIN950
Czech	EE8PC852
Danish	WE8PC850
Dutch	WE8PC850
English (United Kingdom)	WE8PC850
English (United States)	US8PC437
Finnish	WE8PC850
French	WE8PC850
German	WE8PC850
Greek	EL8PC737
Hebrew	IW8PC1507
Hungarian	EE8PC852
Italian	WE8PC850
Japanese	JA16SJIS
Korean	KO16MSWIN949
Norwegian	WE8PC850
Polish	EE8PC852
Portuguese	WE8PC850
Romanian	EE8PC852
Russian	RU8PC866
Slovak	EE8PC852
Slovenian	EE8PC852
Spanish	WE8PC850
Swedish	WE8PC850
Turkish	TR8PC857

2.7. How Windows uses Fonts to display the different charactersets

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.

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 characterset 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 characterset to Arabic. That way Windows knows what are valid codepoints and can use the **FONT** engine to **DISPLAY** the codepoints (this results in glyphs). Windows passes the codepoint and the "page" to the rendering engine. This "page" defines the glyphs for the codepoints for a certain characterset/codepage.

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** codepoint 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 characterset 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 characterset. 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 generaly 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> 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> 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> select dump(upper(chr(231))) from dual:

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

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).
~~~~~~~~~~

+ 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***
~~~~

Name	Description	Comments
US7ASCII	ASCII 7-bit American	SB, ASCII
SF7ASCII	ASCII 7-bit Finnish	SB
YUG7ASCII	ASCII 7-bit Yugoslavian	SB
RU8BESTA	BESTA 8-bit Latin/Cyrillic	SB, ASCII
EL8GCOS7	Bull EBCDIC GCOS7 8-bit Greek	SB
WE8GCOS7	Bull EBCDIC GCOS7 8-bit West European	SB
EL8DEC	DEC 8-bit Latin/Greek	SB
TR7DEC	DEC VT100 7-bit Turkish	SB
TR8DEC	DEC 8-bit Turkish	SB, ASCII
TR8EBCDIC1026	EBCDIC Code Page 1026 8-bit Turkish	SB
TR8EBCDIC1026S	EBCDIC Code Page 1026 Server 8-bit Turkish	SB
TR8PC857	IBM-PC Code Page 857 8-bit Turkish	SB, ASCII
TR8MACTURKISH	MAC Client 8-bit Turkish	SB
TR8MACTURKISHS	MAC Server 8-bit Turkish	SB, ASCII
TR8MSWIN1254	MS Windows Code Page 1254 8-bit Turkish	SB, ASCII, EURO
WE8BS2000L5	Siemens EBCDIC.DF.L5 8-bit West European/Turkish	SB
WE8DEC	DEC 8-bit West European	SB, ASCII
D7DEC	DEC VT100 7-bit German	SB
F7DEC	DEC VT100 7-bit French	SB
S7DEC	DEC VT100 7-bit Swedish	SB
E7DEC	DEC VT100 7-bit Spanish	SB
NDK7DEC	DEC VT100 7-bit Norwegian/Danish	SB
I7DEC	DEC VT100 7-bit Italian	SB
NL7DEC	DEC VT100 7-bit Dutch	SB
CH7DEC	DEC VT100 7-bit Swiss (German/French)	SB
SF7DEC	DEC VT100 7-bit Finnish	SB
WE8DG	DG 8-bit West European	SB, ASCII
WE8EBCDIC37C	EBCDIC Code Page 37 8-bit Oracle/c	SB
WE8EBCDIC37	EBCDIC Code Page 37 8-bit West European	SB
D8EBCDIC273	EBCDIC Code Page 273/1 8-bit Austrian German	SB
DK8EBCDIC277	EBCDIC Code Page 277/1 8-bit Danish	SB
S8EBCDIC278	EBCDIC Code Page 278/1 8-bit Swedish	SB
I8EBCDIC280	EBCDIC Code Page 280/1 8-bit Italian	SB
WE8EBCDIC284	EBCDIC Code Page 284 8-bit Latin American/Spanish	SB
WE8EBCDIC285	EBCDIC Code Page 285 8-bit West European	SB
WE8EBCDIC924	Latin 9 EBCDIC 924	SB, EBCDIC

Name	Description	Comments
WE8EBCDIC1047	EBCDIC Code Page 1047 8-bit West European	SB
WE8EBCDIC1047E	Latin 1/Open Systems 1047	SB, EBCDIC, EURO
WE8EBCDIC1140	EBCDIC Code Page 1140 8-bit West European	SB, EURO
WE8EBCDIC1140C	EBCDIC Code Page 1140 Client 8-bit West European	SB, EURO
WE8EBCDIC1145	EBCDIC Code Page 1145 8-bit West European	SB, EURO
WE8EBCDIC1146	EBCDIC Code Page 1146 8-bit West European	SB, EURO
WE8EBCDIC1148	EBCDIC Code Page 1148 8-bit West European	SB, EURO
WE8EBCDIC1148C	EBCDIC Code Page 1148 Client 8-bit West European	SB, EURO
F8EBCDIC297	EBCDIC Code Page 297 8-bit French	SB
WE8EBCDIC500C	EBCDIC Code Page 500 8-bit Oracle/c	SB
WE8EBCDIC500	EBCDIC Code Page 500 8-bit West European	SB
EE8EBCDIC870	EBCDIC Code Page 870 8-bit East European	SB
EE8EBCDIC870C	EBCDIC Code Page 870 Client 8-bit East European	SB
EE8EBCDIC870S	EBCDIC Code Page 870 Server 8-bit East European	SB
WE8EBCDIC871	EBCDIC Code Page 871 8-bit Icelandic	SB
EL8EBCDIC875	EBCDIC Code Page 875 8-bit Greek	SB
EL8EBCDIC875R	EBCDIC Code Page 875 Server 8-bit Greek	SB
CL8EBCDIC1025	EBCDIC Code Page 1025 8-bit Cyrillic	SB
CL8EBCDIC1025C	EBCDIC Code Page 1025 Client 8-bit Cyrillic	SB
CL8EBCDIC1025R	EBCDIC Code Page 1025 Server 8-bit Cyrillic	SB
CL8EBCDIC1025S	EBCDIC Code Page 1025 Server 8-bit Cyrillic	SB
CL8EBCDIC1025X	EBCDIC Code Page 1025 (Modified) 8-bit Cyrillic	SB
BLT8EBCDIC1112	EBCDIC Code Page 1112 8-bit Baltic Multilingual	SB
BLT8EBCDIC1112S	EBCDIC Code Page 1112 8-bit Server Baltic Multilingual	SB
D8EBCDIC1141	EBCDIC Code Page 1141 8-bit Austrian German	SB, EURO
DK8EBCDIC1142	EBCDIC Code Page 1142 8-bit Danish	SB, EURO
S8EBCDIC1143	EBCDIC Code Page 1143 8-bit Swedish	SB, EURO
I8EBCDIC1144	EBCDIC Code Page 1144 8-bit Italian	SB, EURO
F8EBCDIC1147	EBCDIC Code Page 1147 8-bit French	SB, EURO
EEC8EUROASCI	EEC Targon 35 ASCI West European/Greek	SB
EEC8EUROPA3	EEC EUROPA3 8-bit West European/Greek	SB
LA8PASSPORT	German Government Printer 8-bit All-European Latin	SB, ASCII
WE8HP	HP LaserJet 8-bit West European	SB
WE8ROMAN8	HP Roman8 8-bit West European	SB, ASCII
HU8CWI2	Hungarian 8-bit CWI-2	SB, ASCII
HU8ABMOD	Hungarian 8-bit Special AB Mod	SB, ASCII

Name	Description	Comments
LV8RST104090	IBM-PC Alternative Code Page 8-bit Latvian (Latin/Cyrillic)	SB, ASCII
US8PC437	IBM-PC Code Page 437 8-bit American	SB, ASCII
BG8PC437S	IBM-PC Code Page 437 8-bit (Bulgarian Modification)	SB, ASCII
EL8PC437S	IBM-PC Code Page 437 8-bit (Greek modification)	SB, ASCII
EL8PC737	IBM-PC Code Page 737 8-bit Greek/Latin	SB
LT8PC772	IBM-PC Code Page 772 8-bit Lithuanian (Latin/Cyrillic)	SB, ASCII
LT8PC774	IBM-PC Code Page 774 8-bit Lithuanian (Latin)	SB, ASCII
BLT8PC775	IBM-PC Code Page 775 8-bit Baltic	SB, ASCII
WE8PC850	IBM-PC Code Page 850 8-bit West European	SB, ASCII
EL8PC851	IBM-PC Code Page 851 8-bit Greek/Latin	SB, ASCII
EE8PC852	IBM-PC Code Page 852 8-bit East European	SB, ASCII
RU8PC855	IBM-PC Code Page 855 8-bit Latin/Cyrillic	SB, ASCII
WE8PC858	IBM-PC Code Page 858 8-bit West European	SB, ASCII, EURO
WE8PC860	IBM-PC Code Page 860 8-bit West European	SB. ASCII
IS8PC861	IBM-PC Code Page 861 8-bit Icelandic	SB, ASCII
CDN8PC863	IBM-PC Code Page 863 8-bit Canadian French	SB, ASCII
N8PC865	IBM-PC Code Page 865 8-bit Norwegian	SB. ASCII
RU8PC866	IBM-PC Code Page 866 8-bit Latin/Cyrillic	SB, ASCII
EL8PC869	IBM-PC Code Page 869 8-bit Greek/Latin	SB, ASCII
LV8PC1117	IBM-PC Code Page 1117 8-bit Latvian	SB, ASCII
US8ICL	ICL EBCDIC 8-bit American	SB
WE8ICL	ICL EBCDIC 8-bit West European	SB
WE8ISOICLUK	ICL special version ISO8859-1	SB
WE8ISO8859P1	ISO 8859-1 West European	SB, ASCII
EE8ISO8859P2	ISO 8859-2 East European	SB, ASCII
SE8ISO8859P3	ISO 8859-3 South European	SB, ASCII
NEE8ISO8859P4	ISO 8859-4 North and North-East European	SB, ASCII
CL8ISO8859P5	ISO 8859-5 Latin/Cyrillic	SB, ASCII
AR8ISO8859P6	ISO 8859-6 Latin/Arabic	SB, ASCII
EL8ISO8859P7	ISO 8859-7 Latin/Greek	SB, ASCII, EURO
IW8ISO8859P8	ISO 8859-8 Latin/Hebrew	SB, ASCII
NE8ISO8859P10	ISO 8859-10 North European	SB, ASCII
BLT8ISO8859P13	ISO 8859-13 Baltic	SB, ASCII
CEL8ISO8859P14	ISO 8859-13 Celtic	SB, ASCII
WE8ISO8859P15	ISO 8859-15 West European	SB, ASCII, EURO
LA8ISO6937	ISO 6937 8-bit Coded Character Set for Text Communication	SB, ASCII

Name	Description	Comments
IW7IS960	Israeli Standard 960 7-bit Latin/Hebrew	SB
AR8ARABICMAC	Mac Client 8-bit Latin/Arabic	SB
EE8MACCE	Mac Client 8-bit Central European	SB
EE8MACCROATIAN	Mac Client 8-bit Croatian	SB
WE8MACROMAN8	Mac Client 8-bit Extended Roman8 West European	SB
EL8MACGREEK	Mac Client 8-bit Greek	SB
IS8MACICELANDIC	Mac Client 8-bit Icelandic	SB
CL8MACCYRILLIC	Mac Client 8-bit Latin/Cyrillic	SB
AR8ARABICMACS	Mac Server 8-bit Latin/Arabic	SB, ASCII
EE8MACCES	Mac Server 8-bit Central European	SB, ASCII
EE8MACCROATIANS	Mac Server 8-bit Croatian	SB, ASCII
WE8MACROMAN8S	Mac Server 8-bit Extended Roman8 West European	SB, ASCII
CL8MACCYRILLICS	Mac Server 8-bit Latin/Cyrillic	SB, ASCII
EL8MACGREEKS	Mac Server 8-bit Greek	SB, ASCII
IS8MACICELANDICS	Mac Server 8-bit Icelandic	SB
BG8MSWIN	MS Windows 8-bit Bulgarian Cyrillic	SB, ASCII
LT8MSWIN921	MS Windows Code Page 921 8-bit Lithuanian	SB, ASCII
ET8MSWIN923	MS Windows Code Page 923 8-bit Estonian	SB, ASCII
EE8MSWIN1250	MS Windows Code Page 1250 8-bit East European	SB, ASCII, EURO
CL8MSWIN1251	MS Windows Code Page 1251 8-bit Latin/Cyrillic	SB, ASCII, EURO
WE8MSWIN1252	MS Windows Code Page 1252 8-bit West European	SB, ASCII, EURO
EL8MSWIN1253	MS Windows Code Page 1253 8-bit Latin/Greek	SB, ASCII, EURO
BLT8MSWIN1257	MS Windows Code Page 1257 8-bit Baltic	SB, ASCII, EURO
BLT8CP921	Latvian Standard LVS8-92(1) Windows/Unix 8-bit Baltic	SB, ASCII
LV8PC8LR	Latvian Version IBM-PC Code Page 866 8-bit Latin/Cyrillic	SB, ASCII
WE8NCR4970	NCR 4970 8-bit West European	SB, ASCII
WE8NEXTSTEP	NeXTSTEP PostScript 8-bit West European	SB, ASCII
CL8ISOIR111	ISOIR111 Cyrillic	SB
CL8KOI8R	RELCOM Internet Standard 8-bit Latin/Cyrillic	SB, ASCII
CL8KOI8U	KOI8 Ukrainian Cyrillic	SB
US8BS2000	Siemens 9750-62 EBCDIC 8-bit American	SB
DK8BS2000	Siemens 9750-62 EBCDIC 8-bit Danish	SB
F8BS2000	Siemens 9750-62 EBCDIC 8-bit French	SB
D8BS2000	Siemens 9750-62 EBCDIC 8-bit German	SB
E8BS2000	Siemens 9750-62 EBCDIC 8-bit Spanish	SB
S8BS2000	Siemens 9750-62 EBCDIC 8-bit Swedish	SB

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

Appendix B: References

1. The correct NLS_LANG in a Windows Environment - http://metalink.oracle.com/metalink/plsql/ml2_documents.showDocument?p_database_id=NOT&p_id=179133.1
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
3. Character Sets, Code Pages, Fonts and the NLS_LANG Value - http://metalink.oracle.com/metalink/plsql/ml2_documents.showDocument?p_id=137127.1&p_database_id=NOT
4. How to change the ANSI Code Page (ACP) on Windows NT 4.0 and Windows 2000 / XP - http://metalink.oracle.com/metalink/plsql/ml2_documents.showDocument?p_id=199926.1&p_database_id=NOT
5. Dump SQL Command for NLS Debugging - http://metalink.oracle.com/metalink/plsql/ml2_documents.showDocument?p_id=13854.1&p_database_id=NOT
6. Locale Builder - Frequently Asked Questions - http://metalink.oracle.com/metalink/plsql/ml2_documents.showDocument?p_id=227339.1&p_database_id=NOT
7. Non-ASCII characters are not handled properly in EM client tools when connected through OMS - http://metalink.oracle.com/metalink/plsql/ml2_documents.showDocument?p_id=245281.1&p_database_id=NOT
8. The ISO 8859 Alphabet Soup - http://czyborra.com/charsets/iso8859.html
9. Good ole' ASCII - http://czyborra.com/charsets/iso646.html
10. The Cyrillic Charset Soup - http://czyborra.com/charsets/cyrillic.html