NLS_LANG Explained (How does Client-Server Character Conversion Work?) [ID 158577.1]

Modified: Sep 4, 2012 Type: BULLETIN Status: PUBLISHED Priority: 1

In this Document

Purpose
Scope
Details
1.1 What is Oracle Globalization Support ?
1.2 What is this NLS_LANG thing anyway?
2.1 What is a Characterset or Code Page?
2.2 So Why Are There Different Charactersets?
2.3 What's the Difference Between 7 bit, 8 bit Charactersets and Unicode ?
3.1 Why Should I bother setting the correct NLS_LANG? It seams to work now.
3.2 A detailed example of a *wrong* nls setup to understand what's going on:
3.3 How to see what's really stored in the database?
4.1 So What Should I Do To Have A Correct Setup?
4.1.1 Identify the characterset/codepage used by your clients.
4.1.2 Set the NLS_LANG on the client to the corresponding Oracle characterset.
4.1.3 Create your database with a characterset that supports ALL characters used by your various clients.
4.1.4 Set NLS_LANG on the server ALSO to the characterset used by the OS (terminal type) of the server.
4.2 How can I Check the Client's NLS_LANG Setting?
4.2.1 On Unix:
4.2.2 On Windows:
4.3 Where is the Character Conversion Done?
4.4 NLS_LANG default value:
5.1 My windows sqlplus is not showing all my extended characters.
5.2 I get an (inverted) question mark (? or ¿) when selecting back the just inserted character.
5.3 What about sql*loader, import, export, my tool?
5.4 What about database links?
5.5 What about webclients (browsers) and webservers connecting to Oracle?
5.6 What about Multiple Homes on Windows?
5.7 Is there an Oracle Unicode Client on Windows?
5.8 UTL_FILE is writing / reading incorrect characters.
5.9 DBMS_LOB is not loading flat text files correctly.
5.10 Loading (XML) files as XMLtypes stores incorrect characters.
5.11 ODBC and NLS_LANG.
5.12 everything works except cut-and-paste from txt or word file to sqlplus:
5.13 What's the use of putting ('ENVS= NLS_LANG=....') in the listener.ora?
References

Applies to:

Oracle Server - Enterprise Edition - Version 8.0.3.0 and later
Information in this document applies to any platform.

Purpose

To provide a basic understanding of what is going on if you set NLS_LANG, how the conversion is done and how to set up a correct configuration.
If you think or notice that you have problems with character conversion the please *do* go first of all through this note, so that you have a good understanding what NLS_LANG actually is, if needed create a new db and test.

To change the NLS_CHARACTERSET please see Note 225912.1 Changing the Database Character Set but *please* don't start changing you database characterset without knowing what you are doing. And ALWAYS test it on a backup of your enviroment.

Scope

Anyone configuring a system to handle languages other than English.

Details

1.1 What is Oracle Globalization Support ?

Globalization support enables Oracle software to support different languages and different national conventions in date and monetary formatting.
It's also used to convert the charactersets of different clients to the characterset of the database. The name 'Globalization support' is the new name from Oracle9i onwards for 'National Language Support (NLS)'.

1.2 What is this NLS_LANG thing anyway?

NLS_LANG consist of: NLS_LANG=<Language>_<Territory>.<clients characterset>

This note covers mainly the <clients characterset> part of NLS_LANG. NLS_LANG is set in the enviroment for Linux/Unix and in the registry for windows

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

* Using for the the <clients characterset> part of NLS_LANG the same value as the NLS_CHARACTERSET 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. The used NLS_LANG value has as such no relation with the NLS_CHARACTERSET. If they happen to be the same, fine but they are simply 2 different things. See point 4.1.1 and 4.1.2

* 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, so Oracle can do the proper conversion. You cannot change the characterset of your client by using a different NLS_LANG! To change the client characterset you need to change the OS configuration, not a Oracle parameters.

* 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!

* If the NLS_LANG is the same as the database characterset then Oracle OCI currently (for performance reasons) will do NO validation of the codes passed trough against the configured characterset. See Also "3.2 A detailed example of a *wrong* nls setup to understand what's going on".

Note that while this behavior is quite widely know and there are currently no plans to change this, Oracle does not warrant this will not may change in the future. This is not *documented behavior* that is warranted to stay this way, it is simply a result of how the current client/server conversion is *implemented*. This "switch off" trick is currently already not possible anymore with the JDBC and ODBC interfaces.

* 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 define Japanese characters. But a NLS_LANG set to AMERICAN_AMERICA.JA16SJIS *will* allow you to store Japanese (if the database is also using a NLS_CHARACTERSET that can store Japanese like UTF8 or JA16SJIS and the client is indeed a Japanse windows system)

2.1 What is a Characterset or Code Page?

A characterset is just an agreement on what numeric value a symbol has.
A computer does not know ' A ' or ' B ', it only knows the (binary) numeric value for that symbol, defined in the characterset used by its Operating System (OS) or in hardware (firmware) for terminals. A computer can only manipulate numbers which is why there is a need for charactersets.

An example is 'ASCII', an old 7 bit characterset (US7ASCII in Oracle) , 'ISO-8859-1' an 8 bit characterset on Unix (WE8ISO8859P1 in

Oracle) or 'UTF-8' a Unicode multibyte characterset (UTF8/AL32UTF8 in Oracle).

A code page is the name for the Windows/DOS encoding schemes, for Oracle NLS you can consider it the same as a characterset.

You also have to distinguish between a FONT and a characterset/codepage.
A font is used by the OS to convert a numeric value into a graphical 'print' on screen. The Wingdings Font on Windows is the best example of a font where an ' A ' is NOT shown as an ' A ' on screen, but for the OS the numeric value represents an ' A '. So you don't SEE it as an ' A ', but for Windows it's an ' A ' and will be saved (or used) as an ' A '.

To better understand the above, just open MS Word, choose the Wingdings Font, type your name (you will see symbols) and save this as html, if you open the html file with Notepad you will see that in the <style> section the fonts are declared and lower in the <body> section you will find your name in plain text but with style='font-family:Wingdings' attribute. If you open it in Internet Explorer or Firefox, you will again see the Wingdings symbols. It's the *display* of the data that changes, not *the data itself*.

It's also possible that you don't see with a particular font ALL the symbols defined in the codepage you are using, just because the creator of the FONT did not include a graphical representation for all the symbols in that font. That's why you get sometimes black squares on the screen if you change fonts. On Windows you can use the 'Character Map' tool to see all the symbols defined in a font (!font, not characterset!). See Note 137127.1 for a more in-depth overview of fonts

2.2 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.

Unicode (UTF-8, UTF-16) is a relatively new standard that aims to provide one characterset that defines all characters used in the world, which is an ongoing mission of course.

2.3 What's the Difference Between 7 bit, 8 bit Charactersets and Unicode ?

A 7 bit characterset only knows 128 symbols (2^7) , for example US7ASCII.
An 8 bit characterset knows 256 symbols (2^8) , for example WE8MSWIN1252.
Unicode is a standard aiming to define every language known and has the capability to define over a million characters.
http://www.unicode.org/standard/WhatIsUnicode.html . There is even a proposal to add Tolkiens Tengwar script to the Unicode standard, not yet implemented however ;-).
Oracle has several revisions of the Unicode standard implemented during the years, the current Unicode *characterset* implementation is AL32UTF8 Note:260893.1 Unicode character sets in the Oracle database. More information about Unicode in Oracle is found here : Note:788156.1 AL32UTF8 / UTF8 (Unicode) Database Character Set Implications

3.1 Why Should I bother setting the correct NLS_LANG? It seams to work now.

You can store data with the wrong setup, but that's up to you to take the risk. A common INCORRECT setup is storing 8 bit characters in a 7 bit database or to store non-western languages (like Hebrew, Arabian,...) in an database using a West European character set (ex: WE8ISO8859P1). This is not supported and you will lose data when interfacing (using database links, replication, exp/imp) to other systems.This may render some tools or applications pretty useless at some point. Correcting this is most of the time only possible for a certain percentage of the dataset, meaning that at one point in time you will actually loose data or will need to invest a huge amount of manual intervention to correct the situation. So having a CORRECT NLS setup is really in your best interest.

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

You have created a database on your Unix box with US7ASCII NLS_CHARACTERSET. Your Windows clients work with the MSWIN1252 characterset (regional settings -> western europe = ACP 1252) and you, as dba, use sqlplus on the Unix box (SHH/Telnet) to work on the database from your windows client. 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 enviroment! *********

> 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 deliverd by the client , bit by bit, simply seen there is no way that Oracle can find out if you "lie" by using an incorrect setup :-).

Let's do something now:

You insert an ' é '(LATIN SMALL LETTER E WITH ACUTE) into a table NLS_TEST containing one column 'TEST' of the type 'varchar2(10)'.

As long as you insert into and select from the column on Windows clients with the WE8MSWIN1252 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 ' õ ' (LATIN CAPITAL LETTER O WITH TILDE) instead of the ' é '.

If you insert ' é ' on the unix server and you select the row inserted on the unix at the Windows client you get an ' Å ' (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 ' é ' 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 ' é ' has the hexadecimal value 'E9' and in the Roman8 characterset the hexadecimal value for ' é ' 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 ' õ ' because in its (Roman8) characterset the hexadecimal value 'E9'is representing the letter ' õ '. So instead of the ' é ' you get ' õ ' 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 defines 'é' (WE8MSWIN1252, WE8ISO89859P1, AL32UTF8, etc..) and setting the NLS_LANG on client to WE8MSWIN1252 and on the server to WE8ROMAN8. If you then insert an ' é ' on both sides, you will get an ' é ' 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 WE8MSWIN1252 client are both ' é ' and inserts ' é ' into the database (the code in the database depends on the NLS_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 problems.

Note 225938.1 Database Character Set Healthcheck gives more info on how to check if you can try to change you database characterset without losing data, even if you have stored WE8MSWIN1252 data in an us7ascii database or so.

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

To find the real numeric value for a character stored in the database use the dump command : Note 13854.1 Dump SQL Command for NLS Debugging
For a UTF8 database see also: Note 69518.1 Storing and Checking Character Codepoints in a UTF8/AL32UTF8 (Unicode) database

In general you are only interested to know if data is correct in the database or not. Then can simply use Sqldeveloper from a (Windows) client, this is a "know good client" that needs no NLS configuration. Download the latest version from http://www.oracle.com/technology/products/database/sql_developer/ Sqldeveloper does not need a client installation, for testing purposes we recommend to install this on a client that has no other Oracle software installed.
If new data inserted trough Sqldeveloper in a test table is displayed correctly in SQL Developer then you are sure the NLS_CHARACTERSET supports these characters the database side is fine and the problem is at the client side.
If existing application data is displayed correctly in SQL Developer then you are sure this data is correctly stored in the database and the problem is pure the selecting client side.
If existing application data is NOT correct in SQL Developer then the inserting client config is the first thing to debug.

4.1 So What Should I Do To Have A Correct Setup?

To have a proper NLS environment you have to observe these steps:

4.1.1 Identify the characterset/codepage used by your clients.

You may start with these notes for commmon OS environments:

* For Microsoft Windows platforms: Note 179133.1 The correct NLS_LANG in a Windows Environment

* For Unix platforms, search your OS documentation for "LANG" and "locale" as this is differently implemented by each vendor. Note 264157.1 The correct NLS_LANG setting in Unix Environments explains this further. Note the importance of the TELNET/SSH client configuration.

Please contact your OS vendor (Microsoft, HP, Sun...) if you have questions about your OS configuration.

4.1.2 Set the NLS_LANG on the client to the corresponding Oracle characterset.

If you use windows clients: Note 179133.1 The correct NLS_LANG in a Windows Environment
If you use Unix clients: Note 264157.1 The correct NLS_LANG setting in Unix Environments

For less common client OS's, you can look up what the "oracle name" for a characterset is in this note: Note 226492.1 Listing of Character Sets for 9.2, 9.0.1 and 8.1.7 including Language references

If you have a doubt, log a tar against RDBMS / NLS and provide the info requested in Note 226692.1 Finding out your NLS Setup.

****
It cannot be stressed enough that you need to set the NLS_LANG to the characterset that your client is actually *using*. Not to the characterset you *want* to use, if you need on your client another characterset (to display Cyrillic or so) the you need to see how you can change the characterset of the client on OS level.
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.
****

4.1.3 Create your database with a characterset that supports ALL characters used by your various clients.

The only NLS_CHARACTERSET that will always be able to store ALL characters is an Unicode charterset (UTF8 / AL32UTF8): Note 788156.1 AL32UTF8 / UTF8 (Unicode) Database Character Set Implications

In general the "windows" charactersets are the best choice for an non-Unicode NLS_CHARACTERSET simply because Windows is the dominant *client* platform: Note:264294.1 Choosing from WE8ISO8859P1, WE8ISO8859P15 or WE8MSWIN1252 as db character set.

You can create on Unix a database with a "Windows" characterset like WE8MSWIN1252. Oracle is not depending on the OS for the DATABASE (national) characterset. The only restriction is that you cannot use EBCDIC charactersets (like used on AS400 ea) on ASCII based platforms (like used on Unix and Windows) (or inverse) for the database characterset.

This select gives all the known charactersets for that release of Oracle on that platform.

```
select unique VALUE from V$NLS_VALID_VALUES where PARAMETER ='CHARACTERSET';
```

This select gives the current database (national) characterset:

```
select * from nls_database_parameters where parameter like '%CHARACTERSET%';
```

One can use locale builder (from 9i onwards) to view what characters are defined Note 223706.1 Using Locale Builder to view the definition of character Sets

or use this note for a quick overview: Note 226492.1 Listing of Character Sets for 9.2, 9.0.1 and 8.1.7 including Language references

or check this note to see what characters are known in most common charactersets Note 282336.1 Charts of most current mono-byte Character sets

All charactersets that Oracle implements are based on industry standards, see these web sites for more info about the definition of a characterset:
http://www.unicode.org/
http://www.unicode.org/Public/MAPPINGS/
http://msdn.microsoft.com/en-us/goglobal/bb964653

4.1.4 Set NLS_LANG on the server ALSO to the characterset used by the OS (terminal type) of the server.

in fact the same as sections 4.1.1 and 4.1.2 ...

4.2 How can I Check the Client's NLS_LANG Setting?

To be 100% sure about the value used by the client, you can use these methods to get back the value of NLS_LANG:

4.2.1 On Unix:

```
SQL>HOST ECHO $NLS_LANG
```

This returns the value of the parameter as defined in the CURRENT env.

To double check get also the value known by the sqlplus executable:

```
SQL>@.[$NLS_LANG].
```

If you get something like:

```
SQL>@.[$NLS_LANG].
SP2-0310: unable to open file ".[ENGLISH_UNITED KINGDOM.WE8ISO8859P1]..sql"
```

the "file name" between the '[]' is the value of the NLS_LANG parameter.

If you get this as result:

```
SQL>@.[$NLS_LANG].
SP2-0310: unable to open file ".[$NLS_LANG]..sql"
```

then the parameter NLS_LANG is not set. If the "HOST ECHO .." did returned a value then the NLS_LANG was not exported to the environment.

4.2.2 On Windows:

On Windows you have two possible options, normally the NLS_LANG is set in the registry, but it can also be set in the environment, however this is not often done. The value in the environment takes precedence over the value in the registry and is used for ALL Oracle_Homes on the server(!).
Also note that any USER enviroment variable is taking precedence over any SYSTEM enviroment variable (this is windows behavior, nothing to do with Oracle) if set.

To check if it's set in the environment:

```
SQL>HOST ECHO %NLS_LANG%
```

If this reports just %NLS_LANG% back, the variable is not set in the environment.

```
SQL>HOST ECHO %NLS_LANG%
%NLS_LANG%
```

If it's set it reports something like ENGLISH_UNITED KINGDOM.WE8PC850

```
SQL>HOST ECHO %NLS_LANG%
ENGLISH_UNITED KINGDOM.WE8PC850
```

If NLS_LANG is not set in the enviroment, check the value in the registry:

```
SQL>@.[%NLS_LANG%].
```

If you get something like:

```
SQL>@.[%NLS_LANG%].
SP2-0310: unable to open file ".[ENGLISH_UNITED KINGDOM.WE8MSWIN1252]..sql"
```

the "file name" between the '[]' is the value of the registry parameter.

If you get this as result:

```
SQL>@.[%NLS_LANG%].
SP2-0310: unable to open file ".[%NLS_LANG%]..sql"
```

then the parameter NLS_LANG is also not set in the registry.

Note: the @.[%NLS_LANG%]. "trick" reports the NLS_LANG known by the sqlplus executable, it will not read the registry itself. But then you are not sure if the variable is set in the environment or in the registry. That's the reason of checking with the host command first..

All other NLS parameters for this client connection can be retrieved by a

```
SQL> SELECT * FROM NLS_SESSION_PARAMETERS;
```

See also Note 241047.1 The Priority of NLS Parameters Explained.

Note: SELECT USERENV ('language') FROM DUAL; gives the session's <Language>_<territory>
but the **DATABASE** character set, so the value returned is **not** the client's NLS_LANG setting!

If you log a tar regarding an NLS issue please provide this information: Note 226692.1 Finding out your NLS Setup.

4.3 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.

Note that there is
* pre-9i systems have problems with connecting to AL32UTF8 databases Note:237593.1 Problems connecting to AL32UTF8 databases from older versions (8i and lower)
* a known problem with connecting with an V7 client to an UTF8 9.2 database. This will not fail with an error but any non-us7ascii character will be incorrectly stored. There is no fix for this as this is an non-supported configuration.

4.4 NLS_LANG default value:

If NLS_LANG is not defined the <clients characterset> will default to US7ASCII , see Note 241047.1 The Priority of NLS Parameters Explained.

5.1 My windows sqlplus is not showing all my extended characters.

You see black squares instead of the characters. That's because sqlplusw.exe uses per default the Fixedsys font on windows and this font is not containing all characters, so it may have problems displaying some characters like the euro symbol.
The character is not defined in the font, so windows cannot display it, hence it uses a black square to display something. Note that the character is correctly *stored* in the database and is know at the client (sqlplus) side. If you copy paste the output to notepad or so it will show up correctly.

To change the sqlplus font you need to define

SQLPLUS_FONT -> Courier New
SQLPLUS_FONT_SIZE -> 16

Using choose any fixed-pitch TrueType font available in your Windows system such as "Courier New" or "Lucida Console" should work for most cases.

Do NOT change the sqlplus font to "make a client understanding <insert a language>" by defining the odl SQLPLUS_FONT_CHARSET . If you want to change a windows client from a West European to (for example) a Japanse system you need to change the windows Regional Settings. Changing the SQLPLUS_FONT_CHARSET for sqlplus may APPEAR to work but your data WILL be INCORRECTLY stored. Do not set this parameter.

If you choose a proportional pitch font such as Arial or Times New Roman, or if you enter an unavailable font, the registry entry is ignored and the default font and size, Fixedsys 16, are used.
If you choose an unavailable font size, the default font size, 16, is used.
You need to specify the name in the correct upper/lower case mix. (Courier New not COURIER NEW or so and without extra spaces (!) or qoutes around.)

see http://download.oracle.com/docs/cd/B19306_01/server.102/b14357/ch1.htm#sthref166 for more information.

The same problem occurs with the dos version (sqlplus.exe), here change the properties of the dos box / cmd.exe / command prompt used. Under the Font tab choose there Lucida Console.

As stated before you can see with the windows "Character Map" system tool what characters are known by a font.

Using iSqlPlus or SQL developer (see point 5.7) might be a better solution.

5.2 I get an (inverted) question mark (? or ¿) when selecting back the just inserted character.

This is most likely intended behavior, when your client send characters to the database and the character you are inserting is not *known* by the database characterset then Oracle stores a "replacement" character.

An example: you have a Arabian windows client (AR8MSWIN1256 codepage), so you set your NLS_LANG to ARABIC_BAHRAIN.AR8MSWIN1256 and you are connecting to an database with an EE8MSWIN1250 database characterset.

Now, as long as you use "ascii" type characters like 'T' or some extended characters like 'é' this will not provoke a problem seen both T or é are known/defined in both charactersets.
But from the moment you insert Arabian then you get an " ¿ " because Oracle cannot find a matching letter in the 1250 characterset - 1250 does not define Arabian, if no matching character can be found then a "replacement character" is stored/used. " ¿ " is used in the microsoft/windows charactersets and in the ISO charactersets as replacement character, in US7ASCII this is "?".

It's also possible, but this less visible and not so common, that Oracle is doing a remapping to *another* character that resembles somewhat to the one inserted: a plain " e " for an " é " or so , but this you can check with locale builder in the "Replacement Characters" tab for the characterset you are using on that database.

Of course using an AL32UTF8 database characterset solves all this. A database using an AL32UTF8 NLS_CHARTERSET can store all characters know by windows clients.

There is no such thing as a "compatible" NLS_LANG setting for a certain NLS_CHARACTERSET. Even in above example setting the NLS_LANG to ARABIC_BAHRAIN.AR8MSWIN1256 and connecting to an database with an EE8MSWIN1250 database characterset is from a technical point of view 100% correct. The fact that you cannot store certain characters is a result of a **poorly choosen** setup, but is not a "wrong" or "incompatible" setup.

5.3 What about sql*loader, import, export, my tool?

Basicly it's always the same for any tool used to input data:

- you have INPUT to a client:

* For sqlloader this is the txt/xls/html file you read. -> point 18 in Note 227330.1 Character Sets & Conversion - FAQ
* For import the dump file your read -> Note 227332.1
* For sqlplus is this your command line or GUI enviroment "driving" your keyboard. -> Note 179133.1
* When running a script in sqlplus this is the encoding of the *script*.

* For a JDBC driver the java program that calls it. -> Note 115001.1 NLS_LANG Client Settings and JDBC Drivers
* For webservers see Note 229786.1 NLS_LANG and webservers explained.
* For forms (windows runtime): Note 105809.1 Character Set Support for Developer Tools
* Web Forms 6i/9i is Unicode enabled .
* etc

- You have to set the character set part of the NLS_LANG to the characterset of the INPUT telling Oracle what characterset you are using so the Oracle client can do the conversion to the database.

For output, the same thing, if the tool you are using is capable of outputting the characterset wanted (!) you just tell Oracle to convert it to that characterset by setting the NLS_LANG on the client side.

5.4 What about database links?

The NLS_LANG on the server (or client) has no influence on characterset conversion trough a database link.
Oracle will do
* for CHAR, VARCHAR2 , LONG and CLOB datatypes the conversion from the NLS_CHARACTERSET of the source database to the NLS_CHARACTERSET of the target database (or inverse).
* for NCHAR, NVARCHAR2 and NCLOB datatypes the conversion from the NLS_NCHAR_CHARACTERSETof the source database to the NLS_NCHAR_CHARACTERSET of the target database (or inverse).

in both cases the NLS_LANG is not used in any way.

5.5 What about webclients (browsers) and webservers connecting to Oracle?

See Note 229786.1 NLS_LANG and webservers explained and Note 115001.1 NLS_LANG Client Settings and JDBC Drivers

5.6 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(!). Note 179133.1 The correct NLS_LANG in a Windows Environment

5.7 Is there an Oracle Unicode Client on Windows?

Windows provide an API to the Unicode layer of the OS (this is used by MS office 2000 / XP for example) but Oracle uses the "old" WIN32 API for sqlplus(w).exe. So you CANNOT use sqlplus(w).exe with a NLS_LANG set to UTF8. If this "works" then you have a 100% ***incorrect*** setup :) .

If you need an Unicode client then you might want to use
* SQL developer, this is a "know good client" that needs no NLS configuration (but need unicode font support - so for some languages you need to check if your OS has the correct Unicoee fonts installed) .

Please download the latest version from http://www.oracle.com/technology/products/database/sql_developer/ , do not use the odler version included in Oracle client installations.

* iSQL*PLUS the browser based version of sqlplus.
Note 231231.1 Quick setup of iSQL*Plus 9.2 as unicode (UTF8) client on windows.
Note 281847.1 How do I configure or test iSQL*Plus 10i?

5.8 UTL_FILE is writing / reading incorrect characters.

See Note 227531.1 Character set conversion when using UTL_FILE

5.9 DBMS_LOB is not loading flat text files correctly.

See Note 267356.1 Character set conversion when using DBMS_LOB

5.10 Loading (XML) files as XMLtypes stores incorrect characters.

See Note 229291.1 XDB (xmltype) and NLS related issues for 9.2 and up

5.11 ODBC and NLS_LANG.

See Note 231953.1 ODBC and NLS Related Things to Know

5.12 everything works except cut-and-paste from txt or word file to sqlplus:

See Note 226558.1 An example inserting Cyrillic data into a database on West European windows for a step by step example en explanation what's happening when inserting data from txt or word files containing data in another code page than you are normally using (that note uses Cyrillic as example but you can easily use it for other languages.).

5.13 What's the use of putting ('ENVS= NLS_LANG=....') in the listener.ora?

This is mainly used on Unix and by Oracle Applications. This is used to define the NLS_LANG in the serverprocess environment for callbacks. This has no use for client-server, there the NLS_LANG on the client is used.

References

NOTE:115001.1 - NLS_LANG Client Settings and JDBC Drivers
NOTE:15095.1 - Old Exp/Imp (not datapump) and NLS Considerations
NOTE:179133.1 - The correct NLS_LANG in a Windows Environment
NOTE:223706.1 - Using Locale Builder to view the definition of character sets
NOTE:225912.1 - Changing the Database Character Set (NLS_CHARACTERSET)
NOTE:226558.1 - An example inserting cyrillic data into a database on west european windows
NOTE:231953.1 - ODBC and NLS Related Things to Know
NOTE:264157.1 - The correct NLS_LANG setting in Unix Environments
NOTE:60134.1 - Globalization (NLS) - Frequently Asked Questions
NOTE:788156.1 - AL32UTF8 / UTF8 (Unicode) Database Character Set Implications