The correct NLS_LANG setting in Unix Environments [ID 264157.1]

Modified: Jun 26, 2012 Type: HOWTO Status: PUBLISHED Priority: 2

In this Document

Applies to:

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

Goal

To explain how to set the NLS_LANG correctly in Unix environments.

Fix

To debug display problems (you see "funny" symbols or characters become "?", "¿" or "ÃÂÃÂ") then please also select the existing (application) data in SQL Developer *and* test with new data in a test table inserted trough 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 even 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, hence the database side is fine and the problem is at the client side. This new data, if shown correctly in Sqldeveloper, can then be used as a "reference" to check your client's configuration.
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 before trying to configure anything else.

1- Check which locale is set and correct this if needed.

The example used here is to configure your Unix environment so that you can use Unicode (UTF-8) in the shell on your Unix box , configure your Telnet/SSH software and then use a NLS_LANG set to UTF8 / AL32UTF8 for sqlplus. More information on using Unicode on database level is found in Note:788156.1 AL32UTF8 / UTF8 (unicode) Database Character Set Implications

To see your current setup, use the "locale" command like this:

```
$ locale

example of output:

LANG=fr_FR
LC_CTYPE="fr_FR.iso885915@euro"
LC_COLLATE="fr_FR.iso885915@euro"
LC_MONETARY="fr_FR.iso885915@euro"
```

```
LC_NUMERIC="fr_FR.iso885915@euro"
LC_TIME="fr_FR.iso885915@euro"
LC_MESSAGES="fr_FR.iso885915@euro"
LC_ALL=fr_FR.iso885915@euro
```

Most Unix versions have this as default:

```
$ locale

LANG=
LC_CTYPE="C"
LC_COLLATE="C"
LC_MONETARY="C"
LC_NUMERIC="C"
LC_TIME="C"
LC_MESSAGES="C"
LC_ALL=
```

"C" means US7ASCII , this implies that you can only display a-z, A-Z and 0-9 and nothing else.

We recommend to use UTF-8 when possible, this should look like:

```
$ locale

LANG=en_US
LC_CTYPE="en_US.UTF-8"
LC_COLLATE="en_US.UTF-8"
LC_MONETARY="en_US.UTF-8"
LC_NUMERIC="en_US.UTF-8"
LC_TIME="en_US.UTF-8"
LC_MESSAGES="en_US.UTF-8"
LC_ALL=en_US.UTF-8
```

When you have chosen a value, for example "en_US.UTF-8" on Linux, you can set it like this:

```
$ export LC_ALL=en_US.UTF-8

or

% setenv LC_ALL en_US.UTF-8
```

Please note that the output/syntax of this command is not exactly the same on all the Unix environments. If you have questions on how to configure an user environment on your specific Unix/Linux flavor please consult your OS vendor.

2- Check if the chosen locale is indeed installed and install this if needed.

To see all installed locales issue:

```
$ locale -a

sample output:

$ locale -a

POSIX
common
en_US.UTF-8
C
```

```
iso_8859_1
iso_8859_15
en_CA
en_CA.ISO8859-1
en_US
en_US.ISO8859-1
en_US.ISO8859-15
en_US.ISO8859-15@euro
fr_CA
fr_CA.ISO8859-1
th
th_TH
th_TH.TIS620
ja
...
```

This will list all **installed** locales for the Unix box.
If for example you want to use "fr_FR.iso885915@euro" but this is missing in the list then you need to install this first.
If you set the user environment to a locale that is not installed you will not get an error but it may not work.
Please also note that you need to have installed the exact locale, if you have "fr_FR.UTF-8" or "UTF-8" installed but want to use "en_US.UTF-8" you need to install "en_US.UTF-8".

The Locale parameter syntax is "language(_territory)(.encoding)(@modifier)". The language(_territory) part determines the default format of dates, the interface language and so on OS level (for more info see your system documentation) , so this is something we can't decide on what you actually would like to have, the last (.encoding) part needs to be UTF-8 however if you want to have an Unicode shell environment. Note that the Language and Territory setting of the Unix Locale have nothing to do with the ability to use ***characters*** in the shell. A Locale set to ja_JA.ISO8859-1 will not allow you to use Japanese because ISO8859-1 doesn't know Japanese characters. But a Locale set to en_US.UTF-8 will allow you to use Japanese (assuming your telnet/ssh client settings are correct) in the shell.

In the example "en_US.UTF-8" is listed , so we can use it on this server.

3- Check if you telnet/ssh software is properly configured.

You will need to check that your telnet/ssh software is correctly configured. It's your telnet/ssh software who is responsible for the conversion of the Unix locale to the client's environment (most likely a Windows system).

We suggest that you try first with the free PUTTY client which is to the best of our knowledge working pretty well with Unicode. You can download Putty from this site: http://www.chiark.greenend.org.uk/~sgtatham/putty/

When using the "en_US.UTF-8" like in our example on the Unix side change the following setting in Putty:

Open the configuration window, navigate to "Window" then to "Translation" and set "Received data assumed to be in which character set" to "UTF-8". This needs to match the encoding on the Unix shell side.

Then navigate to "Window" then to "Appearance" and choose a font in the "Font used in the terminal window" that supports the languages you want to use/see.

Note the fundamental difference between a encoding/characterset which defines what code is what character and a font which used by the OS to "draw" the character on the screen that is defined by the encoding. With text based emulators like Telnet/ssh the remote side does not send/receive "characters" to/from the client but **codes in a certain characterset**, then the client OS uses fonts on the Telnet/ssh client side to **draw** these characters on the screen. This implies that the client and remote side first need to "agree" on the used encoding before any fonts come into the picture.

On windows clients for most non-Asian languages the default "Courier New" windows font can be used, a more complete font is "Arial Unicode MS". The "Arial Unicode MS" is normally available on every windows client that has Office 2002 or later installed and supports a wide range of characters. See http://support.microsoft.com/kb/q287247/ for what languages are supported by this font. If you do not have office 2002 installed you can try the GNU FreeFont http://www.gnu.org/software/freefont/index.html collection.. Another excellent resource for Unicode fonts is Alan Wood's website . For example the Shareware "Code2000" Font is one of the most complete "generic" fonts available.

On windows the windows tool "character map" can be used to see what characters are known in a font or alternatively enter characters and choose the font in an editor like wordpad and see if the characters show up correctly.

Please DO check your telnet/ssh client configuration, in almost all cases where there are problems with displaying characters in an "Unix prompt" the problem is due an incorrect configured telnet/ssh client or using a telnet/ssh client that simply cannot be configured like the standard Windows telnet.

If it works with Putty but not with your telnet/ssh package then please consult the vendor of your telnet/ssh software.
If you do not use a telnet/ssh client but a "real" Unix display please see Note 265090.1 How to check Unix terminal Environments for the capability to display extended characters.

4- Set the NLS_LANG and Test.

Once you have
1) configured LC_ALL correctly
2) have checked that the used locale is really installed
3) configured your telnet/ssh client
then you can use a NLS_LANG matching the locale.

NLS_LANG consist of: NLS_LANG=<NLS_LANGUAGE>_<NLS_TERRITORY>.<clients characterset>

In the "en_US.UTF-8" example this means setting NLS_LANG=AMERICAN_AMERICA.AL32UTF8 , if the setting would be "fr_FR.UTF-8" then the corresponding NLS_LANG setting would be FRENCH_FRANCE.AL32UTF8.

please note:
* the difference in notation between UTF-8 (Unix) and UTF8 /AL32UTF8 (Oracle)
* That the NLS_LANGUAGE and NLS_TERRITORY setting of the NLS_LANG have **nothing to do with the ability to "see" characters on a client/store characters in a database**. A NLS_LANG set to JAPANESE_JAPAN.WE8ISO8859P15 will not allow you to store Japanese because WE8ISO8859P15 doesn't know Japanese characters. But a NLS_LANG set to AMERICAN_AMERICA.UTF8 will allow you to use/store Japanese (assuming your telnet/ssh client and Locale settings are correct and your database can store Japanese of course - for example a database using an UTF8 or AL32UTF8 NLS_CHARACTERSET)
* To match the NLS_LANGUAGE and NLS_TERRITORY to the language(_territory) value from LC_ALL look up the corresponding oracle language and territory values in the database globalizaton support guide appendix A.

So, login with your Unix user and then

a) Check with locale if LC_ALL is correctly set (assuming here en_US.UTF-8)

b) Double check the Telnet/ssh client configuration (assuming here Putty configured as in point 3)

c) Set the NLS_LANG to match the locale settings

```
$ export NLS_LANG=AMERICAN_AMERICA.AL32UTF8

or

% setenv NLS_LANG AMERICAN_AMERICA.AL32UTF8
```

d) Connect with sqlplus to your database and select some data.

An excellent check to do is to issue "select UNISTR('\20AC') from dual;" This will give the euro symbol when selecting using a 9i or up database that can handle the Euro (AL32UTF8,WE8MSWIN1252,...) and a correct UTF-8 or ISO8859-15 Unix enviroment. If the euro symbol is seen but EXISITING data is not showing up properly then the client is correctly configured but the EXISTING data in the database is/was incorrectly stored. You then will need to correct the EXISTING data in the database before it can be seen in a correctly configured client Note:225938.1 Database Character Set Healthcheck.

If this works, then set the NLS_LANG also in the profile of your user used for sqlplus.

If this is now done then you have a correct setup for interactive inserting and selecting data in sqlplus.

This however does not mean this setting can be used for ALL applications.
The NLS_LANG is used to let the oracle client know what encoding/character set data is delivered to the client libraries. If you have a web application running on this Unix box and that is ouputting iso-8859-1 data then the correct NLS_LANG , even if your Unix locale is UTF-8, is WE8ISO8859P1 for **that** application

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

5- What to do if this is not working?

If you do not see the expected characters then please double check your settings. Most common problem left is that your Telnet/ssh emulator is still NOT correctly configured.
However it's also possible that you have wrong data in your database.

The easy way to check:

Use a windows client, download and install SqlDeveloper from http://www.oracle.com/technology/products/database/sql_developer/index.html , connect to your database and see if your data is correctly displayed in that tool.
If the data is visible in SqlDeveloper then the data is correct in the database and the problem is on the client side.
If the data is not visible in SqlDeveloper then you have wrong data in your database, which means it will not show up correctly, even if your client is correctly configured.

The harder way:

If for example "select ename from scott.emp where empno='7369';" is a select that returns one row of your data then do "select dump(ename,1016),ename from scott.emp where empno='7369';".
You can then look up if the codes match the characters you expect for your database characterset (select value from NLS_DATABASE_PARAMETERS where parameter='NLS_CHARACTERSET';) in Note 282336.1 Charts of most current mono-byte Character sets or in case you have a (AL32)UTF8 database use Note 69518.1 Storing and Checking Character Codepoints in a UTF8/AL32UTF8 (Unicode) database

If you can't figure out what's wrong log a tar, refer to this note and provide:
* the info asked in Note 226692.1 Finding out your NLS Setup.
* a spool (!not copy paste!) of the output of your version of the "select dump(ename,1016),ename from scott.emp where empno='7369';" select .

6- More in depth debugging.

The steps 1-4 should be enough for 99% of the cases, the rest of the note is more in depth debugging

On some platforms, it can be useful to use the following syntax to have more details about
the codepage really used:

```
$ locale LC_CTYPE | head
```

example of output in a HP-UX env:

```
""
""
"iso885915"
""
```

example of output in a Linux env:

```
upper;lower;alpha;digit;xdigit;space;print;graph;blank;cntrl;punct;alnum;combining;combining_level3
toupper;tolower;totitle
16
1
ISO-8859-15
70
84
1
0
1
```

```
$ locale LC_CTYPE | head
upper;lower;alpha;digit;xdigit;space;print;graph;blank;cntrl;punct;alnum;combining;combining_level3
toupper;tolower;totitle
16
```

```
6
UTF-8
70
84
1
0
1
```

On Solaris, AIX, TRU64, this syntax doesn't give interesting complementary information.
To find more details about these settings:
on Solaris, have a look in /usr/lib/locale.
on AIX, have a look in /usr/lib/nls/README
on TRU64, have a look in /usr/lib/nls
on HP-UX, have a look in /usr/lib/nls/config
on Linux, have a look in /usr/share/locale/locale.alias

How to check the codepoints managed by the O.S.:

To know which code point is generated for a character in a Unix Environment,
you can use the "od" command like this (examples with a iso-8859-1 locale):

```
$ od -xc
é
0000000 00e9
351 \0
0000001
```

as you can see the hexa-decimal code point e9 is corresponding to the "é" (lower e acute)
351 is the corresponding Octal value (Octal is the native mode of the od command).

You can also check the character corresponding to a code point using the "echo" command like this:

for Solaris, AIX, HP-UX, TRU64:

```
$echo '\0351'
é
```

for Linux:

```
$echo -e '\0351'
é
```

As you can see, echo uses the Octal value, so you need to convert in octal the value you want to check.

7- Do the Locale and NLS_LANG *need* to match the database characterset?

No, the Locale and NLS_LANG setting (and if applicable the telnet/ssh config) need to match, but they are ALL technically unrelated to the database characterset and they are only relevant for that client environment.

Assume you have an AL32UTF8 NLS_CHARACTERSET database. If you configure your Unix env as (for example) fr_FR.iso885915@euro (the matching NLS_LANG is then FRENCH_FRANCE.WE8ISO8859P15) then you will be able to see/insert in the Unix shell the Euro symbol and "French" or "Spanish" but not for example "Russian" or "Chinese" (= any language not defined in iso885915).
However if you then use a remote client (example a Windows client using Sqldeveloper) that is able to store/process those languages you will be able to insert/select those languages into/from the AL32UTF8 database. This is simply because the server NLS_LANG/Locale setting is only relevant when using the server itself as client.

A lot of customers tend to go trough great pain to "correct" the Unix shell when changing the database NLS_CHARACTERSET to (AL32)UTF8, but it's not relevant as long as you are not using the Unix server itself as client *for data entry*. You should worry more about the clients where your end users are actually inserting the data. It may be nice to be able to see/insert Chinese in sqlplus on the server, but even if this is working it has no relevance for (for example) your Windows clients seen the Windows client setup is totally unrelated to the Unix side. For Windows clients see Note 179133.1 The correct NLS_LANG in a Windows Environment

Note that when loading flat txt files on the server the characterset for sqlldr is depending on the characterset of the txt / flat file not the database characterset or used Unix locale, see: Note 227330.1 Character Sets & Conversion - Frequently Asked Questions / point 18.

What is the best way to load non-US7ASCII characters using SQL*Loader or External Tables?

References

NOTE:158577.1 - NLS_LANG Explained (How does Client-Server Character Conversion Work?)
NOTE:179133.1 - The correct NLS_LANG in a Windows Environment
NOTE:227330.1 - Character Sets & Conversion - Frequently Asked Questions
NOTE:265090.1 - How to check Unix terminal Environments for the capability to display extended characters.
NOTE:229786.1 - NLS_LANG and webservers explained.
NOTE:115001.1 - NLS_LANG Client Settings and JDBC Drivers