

AL32UTF8 / UTF8 (Unicode) Database Character Set Implications [ID 788156.1]

Modified: Aug 29, 2012 Type: BULLETIN Status: PUBLISHED Priority: 1

In this Document[Purpose](#)[Scope](#)[Details](#)[A\) Often asked questions:](#)[A.1\) Do I need to use Nchar, Nvarchar2 or Nclob?](#)[A.2\) Does my Operating System need to support Unicode or do I need to install character sets?](#)[A.3\) What are the Unicode character sets and the Unicode versions in Oracle RDBMS?](#)[A.4\) Is -insert language or character here- supported/defined/known in an Oracle AL32UTF8/UTF8 database?](#)[A.5\) I also want to upgrade to a new Oracle version, do I go to AL32UTF8 before or after the upgrade?](#)[B\) Server side implications.](#)[B.1\) Storage.](#)[B.2\) How much will my database grow when going to AL32UTF8?](#)[B.3\) Codepoints for characters may change in AL32UTF8.](#)[B.4\) The meaning of SP2-0784, ORA-29275 and ORA-600 \[kole t2u\], \[34\] errors / losing characters when using convert.](#)[B.5\) Going to AL32UTF8 from another character set.](#)[B.6\) ORA-01401 / ORA-12899 while importing data in an AL32UTF8 database \(or move data using dblinks\).](#)[B.7\) Object and user names using non-US7ASCII characters.](#)[B.8\) The password of an user can only contain single byte data in 10g and below.](#)[B.9\) When using DBMS_LOB.LOADFROMFILE.](#)[B.10\) When using UTL_FILE](#)[B.11\) When using sqlldr or external tables.](#)[B.12\) Make sure you do not store "binary" \(pdf , doc, docx, jpeg, png , etc files\) or Encrypted data in character datatypes \(CHAR, VARCHAR2, LONG, CLOB\).](#)[B.13\) String functions work with characters not byte \(length,like,substr ...\).](#)[B.14\) LPad and Rpad count in "display units" not characters.](#)[B.15\) Using LIKE and INSTR.](#)[B.16\) Character functions that are returning character values might silently truncate data.](#)[B.17\) Column size triple when using Materialized Views / CTAS through database link.](#)[B.18\) When fetching data from non-AL32UTF8 databases using cursors \(PL/SQL\)](#)[B.19\) When using HTMLDB.](#)[B.20\) When using non-US7ASCII names in directory's or file names.](#)[B.21\) When using XDB \(xmltype\).](#)[B.22\) Upper and NLS_upper give unexpected results on the Micro symbol or turkish i and l characters.](#)[B.23\) Lower and NLS_lower do not handle Greek Sigma Uppercase / capital Σ to lowercase conversion based on position of the Sigma symbol.](#)[B.24\) After going to AL32UTF8 ORA-24816: Expanded non LONG bind data supplied after actual LONG or LOB column error may be seen](#)[B.25\) What is the impact on CPU and memory usage?](#)[C\) The Client side.](#)[C.1\) Common misconceptions about NLS_LANG.](#)[C.2\) Configuring your UNIX client to be an UTF-8 \(Unicode\) client.](#)[C.3\) Configuring your Windows client to be an UTF-8 \(Unicode\) client.](#)[C.4\) The default column width of output in sqlplus will change.](#)[C.5\) Configuring your web based client to be a Unicode client.](#)[C.6\) Using Sqlplus to run scripts inserting non-US7ASCII data](#)[C.7\) Spooling files using sqlplus is much slower using NLS_LANG set to UTF8 or AL32UTF8](#)

[C.8\) Using Oracle Applications.](#)

[C.9\) Using Portal.](#)

[C.10\) Oracle Forms PDF and Unicode](#)

[C.11\) Changing a database to AL32UTF8 hosting an OracleAS 10g Metadata Repository.](#)

[D\) Known Issues](#)

[E\) Other useful references](#)

[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 some practical hints on how to deal with the effects of moving to an AL32UTF8 database character set and using Unicode clients.

While this note is written for going to AL32UTF8/UTF8 most of the facts are also applicable when changing to any other Multibyte character set (ZHS16GBK, ZHT16MSWIN950, ZHT16HKSCS, ZHT16HKSCS31, KO16MSWIN949, JA16SJIS ...), simply substitute AL32UTF8 with the xx16xxx target character set. But in that case going to AL32UTF8 would be simply a far better idea. [Note 333489.1](#) Choosing a database character set means choosing Unicode

Scope

This note ignores any difference between AL32UTF8 and UTF8 and uses AL32UTF8, all information in this note is however the same for UTF8.

Choosing between UTF8 and AL32UTF8 (in 9i and up) is discussed here: [Note 237593.1](#) Problems connecting to AL32UTF8 databases from older versions (8i and lower)

Basically, if your setup is completely (!) (all clients and all servers) 9i or higher, use AL32UTF8 as NLS_CHARACTERSET (unless there are restrictions posed by the Application layer/vendor like for example Oracle Applications lower than Version 12).

If there are older 8i or lower clients use UTF8 and not AL32UTF8 as NLS_CHARACTERSET.

IMPORTANT: Do NOT use Expdp/Impdp when going to (AL32)UTF8 or an other multibyte character set on ALL 10g versions lower than 10.2.0.4 (including 10.1.0.5). Also 11.1.0.6 is affected.

It will provoke data corruption unless you applied [Patch 5874989](#) on the Impdp side, Expdp is not affected. The "old" exp/imp tools are not affected. This problem is fixed in the 10.2.0.4 and 11.1.0.7 patch set.

For windows the fix is included in

10.1.0.5.0 Patch 20 (10.1.0.5.20P) or later, see [Note 276548.1](#)

10.2.0.3.0 Patch 11 (10.2.0.3.11P) or later, see [Note 342443.1](#)

Details

A) Often asked questions:

A.1) Do I need to use Nchar, Nvarchar2 or Nclob?

People often think that data types like NCHAR, NVARCHAR2 or NCLOB (NLS_NCHAR_CHARACTERSET / National Character set datatypes) need to be used to have UNICODE support in Oracle.

This is simply not true.

The NLS_NCHAR_CHARACTERSET (used for NCHAR, NVARCHAR2 and NCLOB columns) is in 9i and up always Unicode (see [Note 276914.1](#) The National Character Set in Oracle 9i 10g and 11g) but "normal" CHAR, VARCHAR2, LONG and CLOB columns can be used for storing Unicode. In that case an AL32UTF8 NLS_CHARACTERSET database is needed.

```
SQL> select value from NLS_DATABASE_PARAMETERS where parameter='NLS_CHARACTERSET'

VALUE
-----
AL32UTF8
```

And then all "normal" CHAR, VARCHAR2, LONG and CLOB datatypes are "Unicode".

It is also **not** possible to use AL16UTF16 as NLS_CHARACTERSET, AL16UTF16 can **only** be used as NLS_NCHAR_CHARACTERSET, see [Note:276914.1](#) The National Character Set in Oracle 9i 10g and 11g

A.2) Does my Operating System need to support Unicode or do I need to install character sets?

For an Unicode database Oracle does not need "Unicode support" from the OS where the database is running on because the Oracle AL32UTF8 implementation is not depending on OS features.

It's for example perfectly possible to run/use an AL32UTF8 database on an Unix system that has not installed any UTF-8 locale. It's however advisable to configure you OS to use UTF-8 so that you can use this environment as UTF-8 *client*.

There is also no need to "install Unicode" or so for the Oracle database/client software, all character sets known in a database version, and this includes Unicode character sets, are **always** installed. You simply cannot choose to not install them.

A.3) What are the Unicode character sets and the Unicode versions in Oracle RDBMS?

For information on the Unicode character sets in Oracle and the versions of Unicode supported please see: [Note 260893.1](#) Unicode character sets in the Oracle database

Choosing between UTF8 and AL32UTF8 (in 9i and up) is discussed here: [Note 237593.1](#) Problems connecting to AL32UTF8 databases from older versions (8i and lower)

Basically, if your setup is completely (!) (all clients and all servers) 9i or higher, use AL32UTF8 as NLS_CHARACTERSET (unless there are restrictions posed by the Application layer/vendor like for example Oracle Applications lower then Version 12). If there are older 8i or lower clients use UTF8 and not AL32UTF8 as NLS_CHARACTERSET.

A.4) Is -insert language or character here- supported/defined/known in an Oracle AL32UTF8/UTF8 database?

The short answer, when using AL32UTF8, is "yes".

For some languages like HKCSC2004 UTF8 may not be ideal. If you want to be 100% sure check the Unicode version of the Oracle release and then have a look at <http://www.unicode.org> or [Note 1051824.6](#) What languages are supported in an Unicode (UTF8/AL32UTF8) database?

Most likely it's a bigger question if the client environment can support the language in question then an AL32UTF8 database.

Note that the support to store a language in an Unicode database is not related and has nothing to with the Oracle Installer "Product Language" choice. The installed "Product Language" refers to translation of the database messages. See [note 985974.1](#) Changing the Language of RDBMS (Error) Messages

Even if you only install the English "product language" you can **store any language** in an AL32UTF8 database.

A.5) I also want to upgrade to a new Oracle version, do I go to AL32UTF8 before or after the upgrade?

If your current Oracle version is 8.1.7 or lower then it's best to upgrade first to a higher release, mainly because a) you then can use AL32UTF8 (not possible in 8i) and b) Cscan has a few issues in 817 who might provoke confusion.

If your current Oracle version is 9i or up then both (before or after) are a good choice, it simply depends on your preference or needed application changes. We would however advice to not do the upgrade and the character set change at the same time, simply to be able to trace issues who might arise to or the upgrade or the character set change. Since doing an upgrade or character set change NEEDS proper testing and Q&A this is of course less relevant for production systems, the changes are then already well tested.

Please do **not** try to run cscan in the lower version, upgrade and then run csalter. After the upgrade you **need** to run csminst.sql and cscan again.

Upgrading to 11.2.0.3 (or higher) before doing the migration to AL32UTF8 might be a good idea so the new Database Migration Assistant for Unicode (DMU) can be used instead of Cscan/csalter seen DMU can do the conversion to AL32UTF8 without need to export /import (a part of) the dataset.

The DMU tool is supported against Oracle 11.2.0.3 and higher and selected older versions and platform combinations.

For more information please see [Note 1272374.1](#) The Database Migration Assistant for Unicode (DMU) Tool and [the DMU pages on OTN](#).

From Oracle 12c onwards, the DMU will be the only tool available to migrate to Unicode.

B) Server side implications.**B.1) Storage.**

AL32UTF8 is a varying width character set, which means that the code for 1 character can be 1, 2, 3 or 4 bytes long. This is a big difference with character sets like WE8ISO8859P1 or WE8MSWIN1252 where 1 character is always 1 byte.

US7ASCII characters (A-Z,a-z,0-9 and ./?,*# etc..) are in AL32UTF8 always 1 byte, so for most West European languages the impact is rather **limited for the whole dataset** as only "special" characters will use more bytes than in a 8 bit character set and they are not that often used (compared to A-Z) in most Western Languages..

When converting a Cyrillic or Arabic system to AL32UTF8, seen **all** the Cyrillic or Arabian data will take considerable more bytes to store the impact **on the whole dataset** will be bigger.

note that ANY character OTHER than US7ASCII (A-Z,a-z,0-9 and ./?,*# ..) will take more "bytes" to store the same character, so on COLUMN level this may have a big impact

columns need to be big enough to store the additional bytes. By default the column size is defined in BYTES and not in CHARACTERS. By default a "create table <name> (<colname> VARCHAR2 (2000));" means that that column can store 2000 **bytes**.

From 9i onwards it's possible to define the column length with the number of characters you want to store, regardless of the character set. How this works, what the limits and current known problems are is explained in [Note 144808.1](#) Examples and limits of BYTE and CHAR semantics usage

More info on how AL32UTF8 encoding works can be found in [Note 69518.1](#) Storing and Checking Character Codepoints in a UTF8/AL32UTF8 (Unicode) database.

note: UTF8 can be 1, 2, 3 or 6 bytes / character. any AL32UTF8 character that is 4 bytes will be stored as 2 time 3 bytes in UTF8. The amount of "real life" characters that will be 6 bytes is limited and will only exist when using (some) chinese characters - see [Note 69518.1](#) and [note 787371.1](#)

B.2) How much will my database grow when going to AL32UTF8?

The biggest expansion will be seen with CLOB's, if the source database is a 8 bit character set (WE8ISO8859P1, WE8MSWIN1252 etc) then populated Clob columns will double in disk size. [Note 257772.1](#) CLOBs and NCLOBs character set storage in Oracle Release 8i, 9i and 10g

An Estimation of the expansion is listed in the Cscan .txt file output under the Expansion header. See [Note 444701.1](#) Cscan output explained

We advice to use Cscan *always* when going to AL32UTF8, see point B.5).

For non-CLOB the expansion is typically a few % for West European databases seen most characters are actually US7ASCII characters. Databases storing other language groups like Arabic, Cyrillic etc will see an overall higher amount of data expansion than West European databases.

B.3) Codepoints for characters may change in AL32UTF8.

There is a common misconception that a character is always the same code, for example the pound sign is often referred as "code 163" character. This is *not* correct, a character is a certain code only in a certain character set (!). The code itself means nothing if you do not know what character set you are using.

The difference may look small, but it's not.

The pound sign for example is indeed "code 163" (A3 in hex) in the WE8ISO8859P1 and WE8MSWIN1252 character sets, but in AL32UTF8 the pound sign is code 49827 (C2 A3 in hex).

When using chr(163) in a AL32UTF8 database the 163 code is a illegal character, as 163 simply does not exist in UTF8, the pound sign is chr(49827) in an UTF8/AL32UTF8 system.

So be careful when using for example the CHR(<code>) function, the code for a character depends on the database character set!

Instead of CHR() it's far better to use Unistr('\<Unicode codepoint>'). Unistr() (a 9i new feature) works always on every character set that knows the character. There is for example no need to change the Unistr value for the Euro symbol when changing from WE8MSWIN1252 to AL32UTF8.

For more info on how to check/find the code for a character in AL32UTF8 and using Unistr please see [Note 69518.1](#) Storing and Checking Character Codepoints in an UTF8/AL32UTF8 (Unicode) database

Only US7ASCII (A-Z,a-z,0-9) characters have the same codepoints in AL32UTF8 as in US7ASCII, WE8ISO8859P1, AR8MSWIN1256

etc. meaning that using chr() for any value above 128 should be best avoided.

B.4) The meaning of SP2-0784, ORA-29275 and ORA-600 [kole_t2u], [34] errors / losing characters when using convert.

If you receive errors like **SP2-0784: Invalid or incomplete character beginning 0xC4 returned** or **ORA-29275: partial multibyte character** or **ORA-600 [kole_t2u], [34]** then this means that you are storing data in a character datatype that is NOT using AL32UTF8 encoding.

Not often seen but an ORA-00911: invalid character or ORA-24812: character set conversion to or from UCS2 failed when using an AL32UTF8 db means in most cases the same thing as the other errors.

As can be seen in [Note 69518.1](#) Storing and Checking Character Codepoints in an UTF8/AL32UTF8 (Unicode) database an UTF8 code sequence cannot start with C4 for example.

The ORA-29275: partial multibyte character error is a result of Oracle is doing "sanity" checks on character strings on very low level to see if the code sequence is valid AL32UTF8, the checks are (for performance reasons) not yet 100%, so some illegal code sequences may not be detected, but they are enhanced in every version, the checks in 11g are far better/stricter than in 9i for example. This is done to avoid wrong result sets from functions and to reduce the risk of injection problems leading to security problems.

With Clob data you will not encounter ORA-29275 but ORA-600 [kole_t2u], [34]. See [Note 734474.1](#) ORA-600 [kole_t2u], [34] - description, bugs, and reasons

SP2-0784 is a pure client side error/warning returned by sqlplus, it means the same as ORA-29275.

Note that those errors cannot be "turned off" and nor should they be. They error indicate a serious problem with your setup which **needs** to be resolved.

Any character datatype like CHAR, VARCHAR2, LONG and CLOB expect the data to be in the encoding defined by the NLS_CHARACTERSET. Storing data in an encoding that is not the NLS_CHARACTERSET is not supported. Any data using an encoding different from the NLS_CHARACTERSET should be considered BINARY and a BINARY datatypes like RAW or BLOB should be used to store and process (!) this.

Most of the time this is seen with "encrypted" (passwords etc) data stored in a VARCHAR2. If this is happening with "encrypted" data then see [Note 1297507.1](#) Problems with (Importing) Encrypted Data After Character Set Change Using Other NLS_CHARACTERSET Database or Upgrading the (client) Oracle Version

This is also true when using the "convert" function by the way, any conversion from the NLS_CHARACTERSET to a other character set should be considered as binary data as the result is not in the NLS_CHARACTERSET. When using the convert function you might even see errors like ORA-12703: this character set conversion is not supported.

Note that this is **expected** behaviour and that [the convert function should not be used in normal application logic](#). If data needs to be stored in non-UTF8 encoding UTL_RAW.CAST_TO_RAW and UTL_RAW.CONVERT should be used and the result should be stored as RAW or BLOB.

There is only one solution and that is to use CHARACTER datatypes for what they are designed for, store data in the NLS_CHARACTERSET encoding.

If you want to write out files in an other character set then AL32UTF8 then use UTL_FILE, see point B.10)

To find all *stored* data that might give these errors then a Unicode client like Sqldeveloper can be used to check the data and update it. For finding non-AL32UTF8 codes in a database Cscan can be used.

install this first

[Note 458122.1](#) Installing and configuring CSSCAN in 8i and 9i

[Note 745809.1](#) Installing and configuring CSSCAN in 10g and 11g

The run Cscan with this syntax:

```
$ cscan \"sys/<syspassword>@<TNSAlias> as sysdba\" FULL=Y FROMCHAR=<current NLS_CHARACTERSET> TOCHAR=<current NLS_CHARACTERSET> LOG=dbcheck CAPTURE=N ARRAY=1000000 PROCESS=2
```

* Always run Cscan connecting with a 'sysdba' connection/user, do not use "system" or "csmig" user.

* The <current NLS_CHARACTERSET> is seen in NLS_DATABASE_PARAMETERS.

select value from NLS_DATABASE_PARAMETERS where parameter='NLS_CHARACTERSET';

* the TOCHAR=<current NLS_CHARACTERSET> is not a typo, the idea is to check the CURRENT character set for codes who are not defined in this NLS_CHARACTERSET

* Above syntax will do a FULL database scan, if you want to scan only certain tables or users then please see " E) Do I need to always run a full database scan?" in [Note 444701.1](#) Cscan output explained

To have an overview of the Cscan output and what it means please see [Note 444701.1](#) Cscan output explained

Any "Lossy" is data that will give ORA-29275 and/or ORA-600 [kole_t2u], [34],

For encrypted data

If this data is not "encrypted" data but "normal" data then it means it is stored using non-UTF8 codes (= bad client config when the data was loaded) this data needs to be CORRECTED through reload in a correct manner, manual update or, if the actual encoding can be found, using UTL_RAW.CAST_TO_RAW, then UTL_RAW.CONVERT and UTL_RAW.CAST_TO_VARCHAR2 and then update the row.

It might not be possible to correct all data.

Known Oracle Bugs who can give ORA-29275 or ORA-600 [KOLE_T2U]

[Bug 4562807](#) ORA-600 [KOLE_T2U] - only related to oracle text, happens on bad input data

Fixed in 10.2.0.4 patchset and 11.1 and higher base release.

[Bug 6268409](#) ORA-29275 ERROR WHEN QUERYING THE SQL_REDO/UNDO COLUMNS IN V\$LOGMNR_CONTENTS

Fixed in 10.2.0.5, 11.1.0.7 and up

[Bug 5915741](#) ORA-29275 selecting from V\$SESSION with Multibyte DB

Fixed in 10.2.0.5, 11.1.0.6 and up

[Bug 10334711](#) ORA-600 [KOLE_T2U] when using EXTENDED feature of AUDIT_TRAIL

Fixed in 11.2.0.3 patchset and 12.1 base release.

B.5) Going to AL32UTF8 from another character set.

To change a database NLS_CHARACTERSET to AL32UTF8 using cscan/csalter or export/import into a new AL32UTF8 db we suggest to follow [Note 260192.1](#) Changing the NLS_CHARACTERSET to AL32UTF8/UTF8 (Unicode)

Please note that it's strongly recommended to follow above note when changing a WHOLE database to AL32UTF8, even when using full exp/imp into a new AL32UTF8 db.

If you are exporting/importing only certain users or table(s) between existing databases and the target database is an UTF8 or AL32UTF8 database then please see:

[Note 1297961.1](#) ORA-01401 / ORA-12899 While Importing Data In An AL32UTF8 / UTF8 (Unicode) Or Other Multibyte NLS_CHARACTERSET Database.

For migration to AL32UTF8 (and the deprecated UTF8), there is a new tool available called the Database Migration Assistant for Unicode (DMU). The DMU is a unique next-generation migration tool providing an end-to-end solution for migrating your databases from legacy encodings to Unicode. DMU's intuitive user-interface greatly simplifies the migration process and lessens the need for character set migration expertise by guiding the DBA through the entire migration process as well as automating many of the migration tasks. The DMU tool is supported against Oracle 11.2.0.3 and higher and selected older versions and platform combinations.

For more information please see [Note 1272374.1](#) The Database Migration Assistant for Unicode (DMU) Tool and [the DMU pages on OTN](#).

From Oracle 12c onwards, the DMU will be the only tool available to migrate to Unicode.

Please see the following note for an Oracle Applications database: [Note 124721.1](#) Migrating an Applications Installation to a New Character Set.

This is the only way supported by Oracle applications. If you have any doubt log an Oracle Applications SR for assistance.

B.6) ORA-01401 / ORA-12899 while importing data in an AL32UTF8 database (or move data using dblinks).

If import gives errors like

```
IMP-00019: row rejected due to ORACLE error 1401
IMP-00003: ORACLE error 1401 encountered
ORA-01401: inserted value too large for column
```

or from 10g onwards:

```
ORA-02374: conversion error loading table "TEST"."NTEST"
ORA-12899: value too large for column COMMENT (actual: 6028, maximum: 4000)
```

or when using a dblink from a non AL32UTF8 db to move data gives ORA-01401 / ORA-12899

then this indicates that the columns cannot handle the "grow in bytes" of the data.

If you are

*exporting/importing certain users or table(s) between existing databases and one database is an UTF8 or AL32UTF8 database

* moving data from a nonAL32UTF8 db to an AL32UTF8 db using dblink

then please see:

[Note 1297961.1](#) ORA-01401 / ORA-12899 While Importing Data In An AL32UTF8 / UTF8 (Unicode) Or Other Multibyte NLS_CHARACTERSET Database.

If you are trying to convert a WHOLE database to AL32UTF8 using export/import then please follow:

[Note 260192.1](#) Changing the NLS_CHARACTERSET to AL32UTF8/UTF8 (Unicode)

B.7) Object and user names using non-US7ASCII characters.

We strongly suggest to never use non-US7ASCII names for a database name and a database link name. See also the limitations listed in the documentation set [Restrictions on Character Sets Used to Express Names](#). This means that for any type of names that has "no" under the "Variable Width" column you NEED to use only US7ASCII characters (a-z,A-Z, 1-0) in an AL32UTF8 database. Using non-US7ASCII for these names is **not supported**. This has to be corrected before going to AL32UTF8. In general avoiding to use non-US7ASCII characters for database objects whenever possible is a very good idea.

(See "Schema Object Naming Rules " in the "Database SQL Reference" part of the docset)

Names can be from 1 to 30 **bytes** long with these exceptions:

* Names of databases are limited to 8 **bytes**.

* Names of database links can be as long as 128 **bytes**.

http://docs.oracle.com/docs/cd/B19306_01/server.102/b14200/sql_elements008.htm#i27570

This select

```
SQL> select object_name from dba_objects where object_name <>
convert(object_name, 'US7ASCII');
```

will return all objects having a non-US7ASCII name.

Using CHAR semantics is not supported in the SYS schema and that's where the database object and user name is stored. If there are column names, schema objects or comments with non-US7ASCII names that take more then 30 bytes in AL32UTF8 there is no alternative besides renaming the affected objects or user to use a name that will occupy maximum 30 bytes.

* An username can be max 30 **bytes** long

http://docs.oracle.com/docs/cd/B19306_01/server.102/b14200/statements_8003.htm#SQLRF01503

and "...his name can contain only characters from your database character set and must follow the rules described in the section "Schema Object Naming Rules". Oracle recommends that the user name contain at least one single-byte character regardless of whether the database character set also contains multibyte characters."

This select

```
SQL> select username from dba_users where username <>
convert(username, 'US7ASCII');
```

will return all users having a non-US7ASCII name.

B.8) The password of an user can only contain single byte data in 10g and below.

http://docs.oracle.com/docs/cd/B19306_01/server.102/b14200/statements_8003.htm#SQLRF01503

Passwords can contain only single-byte characters from the database character set regardless of whether the character set also contains multibyte characters. This means that in an AL32UTF8 database the user password can only contain US7ASCII characters as this are the only single byte characters in AL32UTF8.

This may provoke a problem, if you migrate from (for example) a CL8MSWIN1251 database then your users can use Cyrillic in their passwords, in CL8MSWIN1251 Cyrillic letters are one single byte, in AL32UTF8 they're not.

Passwords are stored in a hashed way, meaning this will **not** be seen in the Cscan result. You will need to reset for those clients the password to an US7ASCII string.

This restriction is lifted in 11g, there multi byte characters can be used as password string. Please note that they need to be updated in 11g before they use the new 11g hashing system. Please see [Note 429465.1](#) 11g R1 New Feature Case Sensitive Passwords and Strong User Authentication

B.9) When using DBMS_LOB.LOADFROMFILE.

When using DBMS_LOB.LOADFROMFILE then please read [Note 267356.1](#) Character set conversion when using DBMS_LOB.

B.10) When using UTL_FILE

When using UTL_FILE then please read [Note 227531.1](#) Character set conversion when using UTL_FILE.

B.11) When using sqlldr or external tables.

When using Sqlldr or external tables make sure to define the correct character set of the file in the control file. The character set of the database has in no direct relation with the encoding used in the file, in other words, it's not because the database is using an AL32UTF8 character set that using AL32UTF8 as NLS_LANG or as character set in the control file is always correct. You need to specify **the encoding of the file** sqlldr is loading.

Please read [Note 227330.1](#) Character Sets & Conversion - Frequently Asked Questions

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

B.12) Make sure you do not store "binary" (pdf , doc, docx, jpeg, png , etc files) or Encrypted data in character datatypes (CHAR, VARCHAR2, LONG, CLOB).

If binary data (like PDF , doc, docx, jpeg, png , etc files or encrypted data) is stored/handled as a CHAR, VARCHAR2, LONG or CLOB datatype then data loss **is expected**, especially when using an AL32UTF8 database (even without using exp/imp). Or errors like ORA-29275 or ORA-600 [kole_t2u], [34] may appear.

The ONLY supported datatypes to store "raw" binary data (like PDF , doc, docx, jpeg, png , etc files or encrypted data) are LONG RAW or BLOB.

If you want to store binary data (like PDF , doc, docx, jpeg, png , etc files or encrypted data) in CHAR, VARCHAR2, LONG or CLOB datatype then this must be converted to a "character set safe" representation like base64 in the application layer.

See [Note 1297507.1](#) Problems with (Importing) Encrypted Data After Character Set Change Using Other NLS_CHARACTERSET Database or Upgrading the (client) Oracle Version

[Note 1307346.1](#) DBMS_LOB Loading and Extracting Binary File To Oracle Database

B.13) String functions work with characters not byte (length,like,substr ...).

Functions like Length and Substr count in characters, not bytes. So in an AL32UTF8 database the result of Length or substr will be different then the amount of BYTES this string uses.

Functions like Substr and Length, who are often used to prepare or limit a string this means application logic, NEED to be checked.

There are of course exceptions like lengthB , substrB and instrB who explicitly deal with bytes. The exact length of a string in BYTES in an AL32UTF8 environment is never known *upfront*, hence operations based on BYTE length should be avoided.

Note that substrB might generate a different result then expected in a UTF8 env:

```
-- the euro symbol € ( U+20AC) is 3 bytes in UTF8
-- a ( U+0061) is one byte
SQL> select dump(to_char(UNISTR('\0061\20AC\0061')),1016) from dual;

DUMP(TO_CHAR(UNISTR( '\0061\20AC\0061' )), 1016)
-----
Typ=1 Len=5 CharacterSet=AL32UTF8:  61, e2, 82, ac, 61

SQL> select dump(substrB(to_char(UNISTR( '\0061\20AC\0061' )),1,4),1016) from
dual;

DUMP(SUBSTRB(TO_CHAR(UNISTR( '\0061\20AC\0061'  ))
```



```

-----
Typ=1 Len=4 CharacterSet=AL32UTF8: 61,e2,82,ac

SQL> select dump(substrB(to_char(UNISTR('\0061\20AC\0061')),1,3),1016) from
dual;

DUMP(SUBSTRB(TO_CHAR(UNISTR('\0061\20AC\006
-----
Typ=1 Len=3 CharacterSet=AL32UTF8: 61,20,20

SQL> select dump(substrB(to_char(UNISTR('\0061\20AC\0061')),1,2),1016) from
dual;

DUMP(SUBSTRB(TO_CHAR(UNISTR('\0061\20AC\
-----
Typ=1 Len=2 CharacterSet=AL32UTF8: 61,20

```

The point here is that SubstrB will replace the codes sequences that are illegal in AL32UTF8 with spaces (=code 20). For example "e2,82,ac" is the Euro in AL32UTF8 encoding, but if you strip the first byte you have "82,ac" which is an illegal code sequence in AL32UTF8, hence the "82,ac" is replaced with "20,20", which is legal (= 2 times a space), otherwise SubstrB would generate strings who can provoke an ORA-29275.

Besides R/Lpad (see below) other functions to check are "REPLACE", "TRANSLATE" and "CHR"/"ASCII". AL32UTF8 "CHR" and "ASCII" values are different then what sometimes is assumed (see point B.3). Also NLS_INITCAP, NLS_LOWER and NLS_UPPER may return multi byte characters, hence the return string may be longer in bytes than the input string.

note also that the default column width of output in sqlplus will change, see point C4)

B.14) LPad and Rpad count in "display units" not characters.

http://download.oracle.com/docs/cd/B28359_01/server.111/b28286/functions087.htm#sthref1621

This means that

```

select lengthb(rpad(bytestst ,10,'x')) lengthb, lengthc(rpad(bytestst ,10,'x'))
lengthc, rpad(bytestst ,10,'x') from
(select UNISTR('\00E9') bytestst from dual);

```

will result in 10 characters but 11 bytes ! (\00E9 - the character é - is in AL32UTF8 2 bytes).

In complex scripts where one "character on the display" may be composed by several combined characters the difference (nr of chars vs bytes) may be even bigger.

With some scripts the number of characters returned may also vary, for example most Asian characters are 2 "display units" wide, so the Japanese 住 character (U+4F4F which is a FULL WIDTH (or zenkaku) character) will "count for 2" in L/RPAD.

```

select lengthc(rpad(japanesechar ,10,'x')) lengthc,
rpad(japanesechar ,10,'x') from (select UNISTR('\4F4F')
japanesechar from dual);

```

Lengthc will return 9 characters.

The thing is that the Japanese character 住 (U+4F4F) takes 2 times the display width of a ascii character like x hence the RPAD will make the string 9 characters in total (2 time display for one 住 + 8 times display for 8 x = 10 display units).

To have a certain number of *characters* or *bytes* one can use something like:

```

RPAD ( str , n - LENGTHC(str), 'c' )

```

Use LENGTHB if the requested width is in *bytes*, use LENGTHC if if the requested width is in *characters*.

```
SUBSTR( str || RPAD( 'c', n, 'c' ), 1, n )
```

Use SUBSTRB if the requested width is in *bytes*, use SUBSTR if the requested width is in *characters*.

In above

* str is the string to be padded.

* 'c' is the fill character -- this must be a single-byte char, usually "space" is used.

* n is the requested width in bytes or characters

There were some issues in 9i and 10g with Lpad/rpad and East_Asian_Width Ambiguous character behavior <http://unicode.org/reports/tr11/#Ambiguous>

"Ambiguous width characters are all those characters that can occur as fullwidth characters in any of a number of East Asian legacy character encodings. They have a "resolved" width of either narrow or wide depending on the context of their use. If they are not used in the context of the specific legacy encoding to which they belong, their width resolves to narrow."

For example the character ↵ 'SOUTH EAST ARROW' (U+2198) is an "East_Asian_Width Ambiguous" character <http://unicode.org/cldr/utility/character.jsp?a=2198>

In 9i you had different behavior for MOST East_Asian_Width Ambiguous characters between UTF8 and AL32UTF8, in 9i UTF8 MOST Ambiguous width characters are always "narrow" (= 1 display place - rpad('↵',5,'x') gives ↵xxxx) and the same Ambiguous width character was in AL32UTF8 "wide" (= 2 display places - rpad('↵',5,'x') gives ↵xxx).

This was not true for every character however (some characters in the "Ambiguous width character" list were not correctly listed as such in oracle's UTF8/AL32UTF8 definition, leading to different l/rpad behavior)

In 11gR1 both UTF8 and AL32UTF8 were updated use the Unicode "Ambiguous width character" list and use "narrow" (= 1 display place - rpad('↵',5,'x') gives ↵xxxx). The "context aware" behavior of "East_Asian_Width Ambiguous" characters could not be implemented due architectural restrictions.

AL16UTF16 implements the "context aware" behavior of "East_Asian_Width Ambiguous" characters.

Note: this needs a NLS_NCHAR_CHARACTERSET set to AL16UTF16

```
alter session set NLS_LANGUAGE='dutch';
select rpad('↵',5,'x') from dual;
select rpad(to_nchar('↵'),5,to_nchar('x')) from dual;
alter session set NLS_LANGUAGE='simplified chinese';
select rpad('↵',5,'x') from dual;
select rpad(to_nchar('↵'),5,to_nchar('x')) from dual;

-- will give

session SET altered.
RPAD('↵',5,'X')
-----
↵XXXX

RPAD(TO_NCHAR('↵'),5,TO_NCHAR('X'))
-----
↵XXXX                <<<<<---- here ↵is 1 display widths because the context
(NLS_LANGUAGE) is Dutch

session SET altered.
RPAD('↵',5,'X')
-----
↵XXXX

RPAD(TO_NCHAR('↵'),5,TO_NCHAR('X'))
-----
↵XXX                <<<<<---- here ↵is 2 display widths because the context
(NLS_LANGUAGE) is Chinese
```

This can be used to force "East_Asian_Width Ambiguous" characters to use a 2 display width:

```

create or replace function Rpad_amb ( p_str varchar2, p_occ number , p_pad
varchar2 default ' ') return varchar2 is
v_nls_language varchar2(40);
v_Rapd_n Nvarchar2(2000);
v_pad_n Nvarchar2(2000);
begin
v_Rapd_n := to_nchar(p_str);
v_pad_n := to_nchar(p_pad);
select value into v_nls_language from nls_session_parameters where parameter =
'NLS_LANGUAGE';
execute immediate ('alter session set NLS_LANGUAGE='''simplified chinese''');
v_Rapd_n := Rpad(v_Rapd_n,p_occ,v_pad_n);
execute immediate ('alter session set NLS_LANGUAGE='''||v_nls_language||''');
return to_char(v_Rapd_n);
end;
/

```

B.15) Using LIKE and INSTR.

When using the LIKE operator it might be an idea to have a look at [Note 232085.1](#) comparison of LIKE2, LIKE4 and LIKEC operators. The same is true for the INSTR/INSTR2/INSTR4/INSTRC functions.

B.16) Character functions that are returning character values might silently truncate data.

Be aware that character functions that are returning character values might silently truncate data , please see

http://docs.oracle.com/docs/cd/B28359_01/server.111/b28286/functions001.htm#sthref928

These functions are: CHR, CONCAT, INITCAP, LOWER, LPAD, LTRIM, NLS_INITCAP, NLS_LOWER, NLSSORT, NLS_UPPER, REGEXP_REPLACE, REGEXP_SUBSTR, REPLACE, RPAD, RTRIM, SOUNDIX, SUBSTR, TRANSLATE, TREAT, TRIM and UPPER.

An example with replace to illustrate this can be found in [Bug 4185519](#) . Please note that this is NOT a bug but intended behaviour.

B.17) Column size triple when using Materialized Views / CTAS through database link.

When using Materialized Views or doing CTAS of a table through a database link it is strongly suggested to use CHAR semantics on BOTH sides to avoid problem with columns size and to avoid ORA-1401 or ORA-12899 errors.

From 8 bit (WE8xxxx) source to (AL32)UTF8 (CTAS target/MV side) database:

When not using CHAR semantics but BYTE semantics the column size will triple (a CHAR 20 BYTE will show up as 60 BYTE in the AL32UTF8/UTF8 db) .

Note: The max byte length of a character is 3 in UTF8 and 4 in AL32UTF8, however there is no 8bit character set that defines a character that takes more than 3 bytes in AL32UTF8 (for example the Euro symbol is max 3 bytes in UTF8/AL32UTF8), so the maximum possible BYTE expansion of data is 3 (for any data that is encoded in a 8bit character set) regardless if the target db is UTF8 or AL32UTF8.

There is enhancement request [Bug 9901628](#) - add parameter to control expansion factor of columns for al32utf8

If it's really needed to use BYTE then the `_keep_remote_column_size=true` parameter can be set at the CTAS/MV side, this will very likely provoke ORA-1401 or ORA-12899 errors as the data will expand when there are non-US7ASCII characters.

To avoid running into ORA-1401 or ORA-12899 during the refresh when using CHAR semantics on both sides make sure the 8 bit side has no columns bigger than CHAR (666 CHAR) or VARCHAR2 (1333 CHAR).

* Avoid to use a mixture of BYTE and CHAR semantics in the same table
 * `_keep_remote_column_size=true` is NOT compatible with using CHAR semantics.

From AL32UTF8 to 8 bit (xx8xxxx) (CTAS target/MV side) database:

The BYTE length of the columns will not be adapted. The byte length of the *data* will remain the same or decrease no issues are to be expected except data that is NOT supported in the target character set will be lost. This is possible since the 8 bit character set only support a subset of AL32UTF8.

Example: when creating a MV in a WE8MSWIN1252 database and the base table in the AL32UTF8 database contains Arabic or Japanese then the Arabic or Japanese will be replaced by a inverse question mark. This is normal as WE8MSWIN1252 only defines West European languages not Arabic or Japanese.

Note that you also will see this factor 3 expansion when going from UTF8 to AL32UTF8 database.
The best solution is here to use AL32UT8 on both sides or use CHAR semantics for the column definitions.
If it's really needed to use BYTE then the `_keep_remote_column_size=true` parameter can be set at the CTAS/MV side

B.18) When fetching data from non-AL32UTF8 databases using cursors (PL/SQL)

Please see [Note 269381.1](#) ORA-01406 or ORA-06502 in PLSQL when querying data in (AL32)UTF8 db from remote non-UTF8 db using cursor

B.19) When using HTMLDB.

When you are using HTMLDB there will be a problem with the passwords after the change to (AL32)UTF8 log a tar (5 / RDBMS / NLS) and refer to this note.

B.20) When using non-US7ASCII names in directory's or file names.

This is basically a pretty bad idea on windows platforms. Also on Unix platforms there are possible problems. More information is in [Note 738758.1](#) Using Non US7ASCII characters in filenames with Extproc, Bfile and other callouts.

Note that using non-US7ASCII characters for datafile names/directory's in an AL32UTF8 database is NOT supported.

B.21) When using XDB (xmltype).

Please see [Note 229291.1](#) XDB (xmltype) and NLS related issues for 9.2 and up

B.22) Upper and NLS_upper give unexpected results on the Micro symbol or turkish i and I characters.

See [Note 1148599.1](#) Upper and NLS_Upper on the (Micro) symbol in an Unicode (AL32UTF8 or AL16UTF16 database) and [Note 329828.1](#) upper and lower function don't work as expected in Turkish for i and I characters

B.23) Lower and NLS_lower do not handle Greek Sigma Uppercase / capital Σ to lowercase conversion based on position of the Sigma symbol.

The Greek letter Sigma exists in three versions: 'GREEK CAPITAL LETTER SIGMA' (Σ): U+03A3 , 'GREEK SMALL LETTER SIGMA' (σ): U+03C3 , 'GREEK SMALL LETTER FINAL SIGMA' (ς): U+03C2, This last one is used only at the end of a word.

Lower and NLS_lower always convert the Sigma Capital (Σ) to Sigma Lowercase (σ): U+03C3 regardless of the position of the Sigma (Σ) symbol . This is not limited to AL32UTF8 databases and is current behavior of Oracle RDBMS, there is enhancement request [Bug 411067](#) to implement this in a future version.

B.24) After going to AL32UTF8 ORA-24816: Expanded non LONG bind data supplied after actual LONG or LOB column error may be seen

There are 2 things who combined provoke this ORA-24816: Expanded non LONG bind data supplied after actual LONG or LOB column error

* When a client connects to a multibyte database (AL32UTF8 for example) and the string , once converted to the multibyte database encoding becomes longer than 4000 BYTES Oracle transforms this internally into a LONG as the data is then longer than the 4000 bytes max value for Varchar2.

* OCI has a requirement that any LOB and LONG needs to be the last of the bind list, this includes that "real" LONG/LOB need to be the last and after any VARCHAR2 that is expanded to over 4000 BYTE due to the character set conversion

http://docs.oracle.com/docs/cd/E11882_01/appdev.112/e10646/oci05bnd.htm#LNOCI16372

The ORA-24816 error itself is introduced in 10.1 and will not be seen in 9i

So in summary, the ora-24816 error is an expected error from database side, when using non LONG or LOB binds of greater than 4000

bytes in size, the "expanded" bind must be placed before any LONG or LOB in the bind list

B.25) What is the impact on CPU and memory usage?

In general going to AL32UTF8 itself (!) will not have a huge impact on RAM and CPU usage.

On memory level the thing that has the most effect is using CHAR semantics, seen for example a 80 CHAR variable allocate 4 times this size in bytes (320 bytes), so when using CHAR semantics the size of user processes and data buffers might indeed grow. Note that this is only a part of the used memory consumption, so it's not that your memory consumption will be 4 times higher. Typical total (!) memory expansion ranges seen are 20 to 30 % when using CHAR semantics and AL32UTF8, but this are just ballpark figures.

On CPU level CHAR semantics / AL32UTF8 itself is not having a huge impact, typical it will be a few % more, on some operations like pattern matching the cpu usage when using AL32UTF8 might be higher.

However when going to AL32UTF8 a lot of the times Linguistic sorting is also used (NLS_COMP = LINGUISTIC and NLS_SORT set to GENERIC_M or other sorts - see [Note:227335.1](#) Linguistic Sorting - Frequently Asked Questions) , if this is used then one will see a big increase in cpu usage.

Linguistic sorting is very CPU intensive compared to BINARY sorting, to reduce the impact it's mandatory to use correct linguistic indexes and having correct statistics but Linguistic sorting will always use a lot more cpu than binary sorting - how much is rather impossible to predict, that depends highly on the dataset and what amount of linguistic sorting is actually used in the application.

If you really want to know the impact on your specific situation , then there is basically only one way, do the migration of a copy of your production system on a test system and do proper load testing with full datasets and the same amount of user activity.

C) The Client side.

C.1) Common misconceptions about NLS_LANG.

It's common to think that the NLS_LANG should be UTF8 or AL32UTF8 when connecting to a AL32UTF8 database. This is not necessarily true, the NLS_LANG has in fact no relation with the database character set. It's purpose is to let oracle know what the **client** character set is, so that Oracle can do the needed conversion.

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

1.2 What is this NLS_LANG thing anyway?

Please make the difference between a client who can **connect** to a Unicode database (which is any 8.0 and up client for an UTF8 database and any 9i and up client for an AL32UTF8 database) and a "real" Unicode client.

A "real Unicode client" means a client who can display/insert all characters know by Unicode without need to recompile or change (Operating system) settings.

There is also often confusion about *where* NLS parameters are define , this is discussed in [Note:241047.1](#) The Priority of NLS Parameters Explained

There is a webcast that might be of interest:

Advisor Webcast - Client side NLS: NLS_LANG and other client NLS parameters explained

<https://oracleaw.webex.com/oracleaw/lr.php?AT=pb&SP=EC&riD=63151037&rKey=c44abd1e6fe0f764%20>

It has a overview of the key concepts and notes of setting NLS parameters and then gives some examples about client config and conversion between different OS environments like using sqlplus and sqldeveloper in a mixed windows / Unix environment.

C.2) Configuring your UNIX client to be an UTF-8 (Unicode) client.

To have a Unix "Unicode client" you need to configure your Unix environment first to use UTF-8, then you have to check your telnet/ssh software to be able to use Unicode and then (as last step) you can use a NLS_LANG set to for example AMERICAN_AMERICA.AL32UTF8 and start sqlplus.

This is explained in [Note 264157.1](#) The correct NLS_LANG setting in Unix Environments

Please do not forget to configure your telnet/ssh client. Using a incorrect telnet/ssh config is the reason for the majority of SR's logged for "display problems in the unix shell".

C.3) Configuring your Windows client to be an UTF-8 (Unicode) client.

On Windows sqlplusw.exe or sqlplus.exe *cannot* be used as an Unicode / AL32UTF8 client to display or insert data interactively (= type things and select /update data by "typing" in sqlplus), in almost ALL cases using sqlplus(w).exe with a nls_lang set to AL32UTF8 is totally

incorrect. This is explained in [Note 179133.1](#) The correct NLS_LANG in a Windows Environment.

The ONLY situation where you can use AL32UTF8 as nls_lang is to

- * run a script that is encoded in UTF-8.
- * create a spool file in UTF-8

Please do note that this is about the encoding/character set of the .sql or spool file, see point C6).

There are 2 Oracle provided AL32UTF8 (Unicode) sql clients for windows systems:

* iSqlplus

[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?

* Oracle SQL Developer

This is also an Unicode client which can be downloaded for free from: http://www.oracle.com/technology/products/database/sql_developer/index.html

To debug display problems (you see "funny" symbols or characters become "?" or an inverted ?) then please select the **existing** (application) data in SQL Developer and test with **new** data in a test table inserted through Sqldeveloper from a (Windows) client. 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 through 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.

* NOT correct in SQL Developer then the **inserting** client config is the first thing to debug before trying to configure anything else.

It still might be needed to install OS support for certain languages, even if sqldeveloper is an Unicode client as it is at the end the OS that displays the data.

Typically this means you will see "white squares" when selecting the data, note that this has as such no relation with "missing characters", the characters is there in the tool, the CLIENT system simply does not know how to display it.

Non-Asian Windows XP installations for example often do not provide standard support for Asian languages

To add this to windows xp:

1. Open Regional and Language Options in Control Panel. (click Start, click Control Panel, click Date, Time, Language, and Regional Options, and then click Regional and Language Options.)
2. On the Languages tab, under Supplemental language support, select the "Install files for East Asian languages" and "the Install files for complex script and right-to-left languages" check box.
3. Click OK or Apply.

You will be prompted to insert the Windows CD-ROM or point to a network location where the files are located.

4. After the files are installed, you must restart your computer.

note:

The East Asian languages include Chinese, Japanese, and Korean.

The complex script and right-to-left languages include Arabic, Armenian, Georgian, Hebrew, the Indic languages, Thai, and Vietnamese.

On windows clients for most non-Asian languages the default "Courier New" windows font can be used, but this might not display Asian languages. 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.

In sqldeveloper you change the font in "Tools" , "Preferences" then expand the "Code Editor" tree and choose "Fonts". In the drop down box choose the font you want.

Note that a missing font (you see squares) has as such no relation with "missing characters", the characters is there in the tool, the CLIENT system simply does not know how to display it.

But you can try to copy paste it for example to another tool like Microsoft Word.

When running into display problems it's always a good idea to check with a "know good client" and check if the data is correctly stored in the database. Oracle SQL Developer is the best tool to do this.

When using / writing an Unicode application on windows then this application should be specifically written to use the Unicode API of windows, setting the NLS_LANG to AMERICAN_AMERICA.AL32UTF8 is **not** enough to "make" an application Unicode. Please consult your application vendor or vendor of the development environment to check if and how to use this as an AL32UTF8 client or Unicode programming environment.

Older versions of the popular Toad tool are NOT able to run against AL32UTF8 databases. Please do **not** use the "workaround" of setting NLS_LANG to UTF8 for toad, data **will** get corrupted doing this.

Information about this is found on Quest's website [here](#) and [here](#) .

For current Unicode support questions about Toad, please contact Quest software.

All Oracle 9i and up Oracle clients are "compatible" with an AL32UTF8 database, even when using a non-AL32UTF8 NLS_LANG (see point C.1)).

It is perfectly correct to use sqlplusw.exe on (for example) a West European / US Windows client (using a NLS_LANG set to AMERICAN_AMERICA.WE8MSWIN1252, which is the correct value for a West European / US Windows system) to connect to a AL32UTF8 database.

However the Western sqlplus client will then only able to insert / view West European characters.

If another user, using sqlplus on a correctly configured Chinese Windows system, inserts data then this will not be visible in the West European sqlplus client. When updating the Chinese data using the West European client this will delete the Chinese data, it will become an inverted question mark.

C.4) The default column width of output in sqlplus will change.

An other often unnoticed side effect (until the migration is done) of going to AL32UTF8 is that the default column width of output will change in sqlplus.

The output in sqlplus when connected to a WE8MSWIN1252 or a other 8 bit character set database

```
SQL> select rpad(dummy,10,'x') from dual;
RPAD(DUMMY,10,'X')
-----
XXXXXXXXXX
```

becomes when connected to an AL32UTF8 database

```
SQL> select rpad(dummy,10,'x') from dual;
RPAD(DUMMY,10,'X')
-----
XXXXXXXXXX
```

For more information please see [Note 330717.1](#) Output widths change after upgrade, or change of character set.

C.5) Configuring your web based client to be a Unicode client.

When using web based applications it's advisable to read

[Note 229786.1](#) NLS_LANG and webservers explained.

[Note 115001.1](#) NLS_LANG Client Settings and JDBC Drivers

When running into display problems it's always a good idea to check with a "know good client" and check if the data is correctly stored in the database. Oracle SQL Developer is the best tool to do this.

C.6) Using Sqlplus to run scripts inserting non-US7ASCII data


When using sqlplus to run .sql file that contain non-US7ASCII characters the NLS_LANG needs to be the encoding of the file.

So, if you load for example a txt/sql flat file made on a windows box in notepad then by default this is WE8MSWIN1252, hence if you run this in sqlplus the NLS_LANG needs to be WE8MSWIN1252 , it doesn't matter if this is ran in sqlplus on unix or <insert any other os >.

An example:

* you followed [Note 264157.1](#) The correct NLS_LANG setting in Unix Environments and correctly configured your Unix environment, your telnet/ssh client and set the NLS_LANG to AMERICAN_AMERICA.AL32UTF8

* you checked that "select UNISTR('\20AC') from dual;" gives indeed the Euro symbol when using the Unix sqlplus.

* Now create (or [download from here](#) ) a test.sql file using notepad and save this as ANSI on an US/West European windows client (file - save as - choose ANSI) which contains:

```
Drop table scott.test;
create table scott.test (numcol number, testcol VARCHAR2(10 CHAR));
insert into scott.test values (1, '<insert here the Capital A circumflex symbol
>');
insert into scott.test values (2, UNISTR('\00c2'));
commit;
select * from scott.test;
```

* ftp this to your unix system and run the test.sql in sqlplus on the Unix server

* then select the data, you will see the Capital A circumflex character for row 2 but row 1 will not display correctly.

The problem here is that the encoding of the FILE is not AL32UTF8 (like your NLS_LANG is set) to but WE8MSWIN1252. The UNISTR('\00c2') rows "works" because this is inserting the data using the Unicode codepoint, not from an actual "physical" character stored in a flat text file.

To correctly load this file you need to run it using a NLS_LANG set to AMERICAN_AMERICA.WE8MSWIN1252, even if your Unix env is NOT a 1252 config. WE8MSWIN1252 is correct because it then reflects the encoding of the FILE.

To check the result however you need to set the NLS_LANG back to the encoding of your Unix and telnet/ssh client, AMERICAN_AMERICA.AL32UTF8.

In general most customers

or have a policy that you should NOT use any non-US7ASCII characters in sql code / scripts - in that case the used nls_lang is not an issue.

or have a policy to use UTF8 encoding without BOM (more on that later) for every flat txt/sql file

If you want to load [UTF-8 encoded .sql](#) files in sqlplus then the NLS_LANG needs to be set to <Language>_<Territory>.AL32UTF8 (for example AMERICAN_AMERICA.AL32UTF8) regardless of the platform (so also on windows) as it needs to match the encoding of the file. The problem is that if you use this setup on windows you will NOT be able to "see" the UTF8 data properly, also (assuming the .sql file contains proper UTF-8 data) it will be inserted correctly - you can check the result with Sqldveloper for example. Only on Unix platforms you can configure a system so that you can use a NLS_LANG set to UTF8/AL32UTF8 and also use this for both loading .sql files in UTF-8 and inserting/selecting UTF8 data "interactively".

When using UTF8/AL32UTF8 as NLS_LANG there is one other issue: the BOM (http://unicode.org/faq/utf_bom.html#BOM) .This is an (not mandatory but allowed) 3 bytes sequence at the start of a flat file to indicate a file is in UTF-8 encoding (products like notepad insert this). Most Microsoft product insert this by default to an UTF encoded file.

This gives a problem in sqlplus, assume 2 files saved in notepad as UTF8 (file save as - choose UTF8):

test1.sql contains on line:

```
select sysdate from dual;
```

test2.sql contains two lines:

```
-- this file needs to be saved as UTF-8
select sysdate from dual;
```

this gives in sqlplus.exe on windows when using a NLS_LANG set to UTF8/AL32UTF8:

```
SQL> @d:\test1.sql
```

```
SP2-0734: unknown command beginning "select sy..." - rest of line ignored.
SQL> @d:\test2.sql
SP2-0734: unknown command beginning "-- this f..." - rest of line ignored.

SYSDATE
-----
23-JUL-09

SQL>
```

The "workaround" of putting a comment line as first line is a bit "ugly", it gives SP2-0734, but it works.

An other solution is to use a script or so that removes the BOM (if present) from flat/txt files <http://www.w3.org/International/questions/qa-utf8-bom>

There is enhancement request [Bug 13515585](#) - add support for the utf-8 bom in sqlplus

In much the same way one can use the sqlplus (also on a windows systems) to create UTF-8 spool files. Note however that the output in sqlplus will NOT be correct, but the resulting data in the spool file will be. To check a UTF-8 file on Windows notepad can be used, file -open - choose "UTF-8" as "encoding". If you see squares instead of the characters then try an other font like "Arial Unicode MS".

C.7) Spooling files using sqlplus is much slower using NLS_LANG set to UTF8 or AL32UTF8

This is [Bug 6350579](#) - spooling with trimspool and linesize takes too long in 10.2.0.3

Fixed-Releases: 11.2.0.1, 10.2.0.5, Windows 10.2.0.4.0 Patch 14 and up, does not happen in 9i

Details: when using a NLS_LANG set to UTF8 or AL32UTF8 spooling a file will take a huge amount of time compared to 9i, the time is especially longer when using " trimspool on" in sqlplus

Workaround: use non-UTF8 nls_lang or use trimspool off

C.8) Using Oracle Applications.

Please see:

[Note 393861.1](#) Globalization Guide for Oracle Applications Release 12

[Note 393320.1](#) Internationalization Update Notes for Release 12

[Note 222663.1](#) Internationalization Update Notes for the Oracle E-Business Suite 11i

C.9) Using Portal.

You need to change after the database change the Portal dad to reflect the new character set. You can do this by:

1. Login to EM Application Server Control
2. Click on the midtier farm
3. Click on HTTP_Server
4. Click on Administration
5. Click on PL/SQL Properties
6. Scroll down to the DAD and click on it.

Within there you can modify the NLS Language to match your db. Press Apply after making the change.

C.10) Oracle Forms PDF and Unicode

[Note 97441.1](#) Does Oracle Reports Support Unicode (UTF8) Characters in PDF Output?

C.11) Changing a database to AL32UTF8 hosting an OracleAS 10g Metadata Repository.

Follow [Note 260192.1](#) Changing the NLS_CHARACTERSET to AL32UTF8/UTF8 (Unicode) and then follow the steps in the documentation set [Oracle Application Server Administrator's Guide 10g Release 2 \(10.1.2\)](#) , [6.5 Changing the Character Set of OracleAS Metadata Repository](#)

D) Known Issues

[Bug 6350579](#) - spooling with trimspool and linesize takes too long in 10.2.0.3

Fixed-Releases: 11.2.0.1, 10.2.0.5, Windows 10.2.0.4.0 Patch 14 and up,

Details: when using a NLS_LANG set to UTF8 or AL32UTF8 spooling a file will take a huge amount of time compared to 9i, the time is especially longer when using " trimspool on" in sqlplus

Workaround: use non-UTF8 nls_lang or do not use trimspool

[Bug 9727970](#) NLS_INITCAP RETURNS WRONG RESULTS WITH NLS_SORT='XTURKISH'

Fixed-Releases: not fixed yet

Details: NLS_INITCAP gives wrong for Turkish result in a non-Turkish character set like AL32UTF8, NLS_UPPER and NLS_LOWER work

Workaround: use NLS_INITCAP(NLS_LOWER(...) ...) instead.

[bug 5010130](#) LPAD/RPAD BEHAVIOR NOT CONSISTENT IN PRE 10GR2 RELEASES WITH AMBIGUOUS CHARS

Fixed-Releases: 11.1.0.6

Details: [Unicode Ambiguous characters](#) can have a display width of 1 or 2 depending on the context, resulting in different behavior when used in L/Rpad, AL16UTF16 implements the Unicode Ambiguous character behavior fully, in 10.2 UTF8 defines all Unicode Ambiguous characters as display width 1, AL32UTF8 defines Unicode Ambiguous characters as display width 2.

Fixed in 11.1.0.6 (both UTF8 and AL32UTF8 define now Unicode Ambiguous characters as display width 1, use AL16UTF16 as NLS_NCHAR_CHARACTERSET and N-types to have the full Unicode Ambiguous character behavior), backport to 10g possible.

The Unicode [EastAsianWidth.txt](#) shows what characters are "Ambiguous", they have "A" as East Asian Width property. Note that this includes non-Asian characters like some Greek or Cyrillic characters.

[Bug 5581731](#) Errors loading wrapped PLSQL in multibyte from client other than SQLPLUS

Fixed-Releases: 11.1.0.6, 10.2.0.4

Details: Attempting to load / compile a wrapped PLSQL input via a mechanism other than SQL*Plus (eg: OCI) if the database's character set is multi-byte can fail with PLS-103, ORA-24344 or other syntax-related compilation errors.

Workaround: Remove trailing blank lines and / from the input

E) Other useful references

[Note 1137194.1](#) Master Note For Bugs, Fixed Versions and Workarounds in PL/SQL

[Note 268895.1](#) Oracle Database Server Patchset Information, Versions: 8.1.7 to 11.2.0

References

[NOTE:115001.1](#) - NLS_LANG Client Settings and JDBC Drivers

[NOTE:237593.1](#) - Problems connecting to AL32UTF8 databases from older versions (8i and lower)

[NOTE:241047.1](#) - The Priority of NLS Parameters Explained (Where To Define NLS Parameters)

[NOTE:257772.1](#) - CLOBs and NCLOBs character set storage in Oracle Release 8i, 9i, 10g and 11g.

[NOTE:144808.1](#) - Examples and limits of BYTE and CHAR semantics usage (NLS_LENGTH_SEMANTICS)

[NOTE:69518.1](#) - Storing and Checking Character Codepoints in an UTF8/AL32UTF8 (Unicode) database

@ [BUG:2940345](#) - GOT ERRORS WHEN INSERTING >1333 CHARACTERS INTO CLOB/LONG WITH UTF8 DB CHARSET

[BUG:4185519](#) - REPLACE SILENTLY DROPS DATA IN IN UTF8 DATABASE WHEN COL IS REACHING 4K BYTES

[NOTE:260893.1](#) - Unicode Character Sets In The Oracle Database

[NOTE:264157.1](#) - The correct NLS_LANG setting in Unix Environments

[NOTE:267356.1](#) - Character set conversion when using DBMS_LOB

[NOTE:734474.1](#) - ORA-600 [kole_t2u], [34] - description, bugs, and reasons

[NOTE:738758.1](#) - Using Non US7ASCII characters in filenames with Extproc, Bfile and other callouts

[BUG:5010130](#) - LPAD/RPAD BEHAVIOR NOT CONSISTENT IN PRE 10GR2 RELEASES WITH AMBIGUOUS CHARS

[BUG:7448978](#) - ORA-24816 RESTRICTION ON LOB DML

[BUG:9727970](#) - NLS_INITCAP RETURNS WRONG RESULTS WITH NLS_SORT='XTURKISH'

[NOTE:260192.1](#) - Changing the NLS_CHARACTERSET to AL32UTF8 / UTF8 (Unicode)

[NOTE:393861.1](#) - Globalization Guide for Oracle Applications Release 12

@ [BUG:6350579](#) - SPOOLING WITH TRIMSPOOL AND LINESIZE TAKES TOO LONG IN 10.2.0.3

[NOTE:124721.1](#) - Migrating an Applications Installation to a New Character Set

[NOTE:281847.1](#) - How to Configure or Test iSQL*Plus 10g?

[NOTE:329828.1](#) - upper and lower function don't work as expected in Turkish for i and l characters

[NOTE:330717.1](#) - Output widths change after upgrade, or change of character set

[NOTE:333489.1](#) - Choosing a database character set means choosing Unicode

[NOTE:342443.1](#) - 10.2.0.x Oracle Database and Networking Patches for Microsoft Platforms

[NOTE:429465.1](#) - 11g R1 New Feature : Case Sensitive Passwords and Strong User Authentication

[NOTE:393320.1](#) - Internationalization Update Notes for Release 12

@[NOTE:367358.1](#) - What Needs to be Done to the WWV_FLOW_FND_USER Table if the Database Characterset Is Changed?

[BUG:13515585](#) - ADD SUPPORT FOR THE UTF-8 BOM IN SQLPLUS

[NOTE:227330.1](#) - Character Sets & Conversion - Frequently Asked Questions

[NOTE:227335.1](#) - Linguistic Sorting - Frequently Asked Questions

[NOTE:227531.1](#) - Character set conversion when using UTL_FILE

[NOTE:231231.1](#) - Quick setup of iSQL*Plus 9.2 as unicode (UTF8) client on windows.

[NOTE:229786.1](#) - NLS_LANG and web servers explained.

[NOTE:232085.1](#) - comparison of LIKE2, LIKE4 and LIKEC operators

[NOTE:985974.1](#) - Changing the Language of RDBMS (Error) Messages

[NOTE:276548.1](#) - 10.1.0.x Oracle Database and Networking Patches for Microsoft Platforms

[NOTE:268895.1](#) - Oracle Database Patchset Information, Versions: 10.1.0 to 11.2.0

[NOTE:269381.1](#) - ORA-01406 or ORA-06502 in PLSQL when querying data in (AL32)UTF8 db from remote non-UTF8 db using cursor

[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:745809.1](#) - Installing and configuring Cscan in 10g and 11g (Database Character Set Scanner)

[NOTE:97441.1](#) - Does Oracle Reports Support Unicode (UTF8) Characters in PDF Output?

[NOTE:1297507.1](#) - Problems with (Importing) Encrypted Data After Character Set Change Using Other NLS_CHARACTERSET Database or Upgrading the (client) Oracle Version

[NOTE:1297961.1](#) - ORA-01401 / ORA-12899 While Importing Data In An AL32UTF8 / UTF8 (Unicode) Or Other Multibyte NLS_CHARACTERSET Database.

[NOTE:1307346.1](#) - DBMS_LOB Loading and Extracting Binary File To Oracle Database

[BUG:9901628](#) - ADD PARAMETER TO CONTROL EXPANSION FACTOR OF COLUMNS FOR AL32UTF8

[NOTE:1051824.6](#) - What languages are supported in an Unicode (UTF8/AL32UTF8) database?

[NOTE:229291.1](#) - XDB (xmltype) and NLS related issues for 9.2 and up

[NOTE:222663.1](#) - Internationalization Update Notes for Oracle E-Business Suite 11i

[NOTE:276914.1](#) - The National Character Set (NLS_NCHAR_CHARACTERSET) in Oracle 9i, 10g and 11g

[NOTE:458122.1](#) - Installing and Configuring Cscan in 8i and 9i (Database Character Set Scanner)

[NOTE:444701.1](#) - Cscan output explained

[BUG:5581731](#) - WRAPPED PACKAGES WITH TRAILING BLANK LINES AND A / VS UNICODE D/B GIVES PLS-103

[BUG:6268409](#) - ORA-29275 ERROR WHEN QUERYING THE SQL_REDO/UNDO COLUMNS IN V\$LOGMNR_CONTENTS

[NOTE:1137194.1](#) - Master Note For Bugs, Fixed Versions and Workarounds in PL/SQL

[NOTE:1148599.1](#) - Upper and NLS_Upper on the μ (Micro) symbol in an Unicode (AL32UTF8 or AL16UTF16 database)

[BUG:5915741](#) - SELECT * FROM V\$SESSION; ENDS WITH ORA-29275