

# Globalization Support Oracle Unicode database support

*An Oracle White Paper  
August 2003*

# Oracle Unicode database support

INTRODUCTION .....	3
REQUIREMENT.....	3
WHAT IS UNICODE ? .....	4
Supplementary Characters .....	4
Unicode Encodings .....	5
UTF-8 Encoding .....	5
UCS-2 Encoding .....	6
UTF-16 Encoding.....	6
UNICODE AND ORACLE .....	7
AL24UTFFSS.....	7
UTF8 .....	8
UTFE.....	8
AL32UTF8 .....	8
AL16UTF16 .....	8
COMMON QUESTIONS AND ANSWERS.....	9
SUMMARY.....	13

## **INTRODUCTION**

This paper helps customers to understand the relationship between the different Unicode encodings and Oracle's character sets. It also contains the answers to some of most common questions regarding Oracle's Unicode implementation. This paper is the starting point for customers trying to find the right Unicode solution for meeting their globalization needs.

## **REQUIREMENT**

Dealing with many different languages in the same application or database has been complicated and difficult for a long time. There is no single character set that contains enough characters to deal with the requirements of day-to-day ebusiness operations. For example, the European Union requires several different national character sets just to cover all its languages. Even for a single language like English, we have no character set available which can encode all the letters, punctuation and technical symbols in general use.

The problem with 7-bit ASCII and other national character sets is that they are not universal. They do not contain enough characters for handling multiple languages. Most of the national character sets are created using ASCII as their base encoding. While you can store combinations like English and French in the same character set, there are some combinations that you can not store, such as German and Chinese, Russian and Korean, or even Traditional Chinese and Simplified Chinese. There are also potential conflicts between the character sets, because the same numeric code value can be used to represent different characters, or the same character is represented by different numeric code values in different character sets.

To overcome the limitations of existing character encodings, several organizations began working on the creation of a global character set in the late 1980s. The need for this became even greater with the development of the World Wide Web in the mid-1990s. The Internet has changed how we do business today, with an emphasis on the global market that has made a universal character set a major requirement. The requirements of this global character set are that it needs to contain all major living scripts, support legacy data and implementations, and be simple enough that a single implementation of a product is sufficient for worldwide use. Additionally it should also support multilingual users and organizations, conform to international

standards, and enable the worldwide interchange of data. This global character set exists and is in wide use; it's called Unicode.

## **WHAT IS UNICODE?**

Unicode is a universal encoded character set that allows you to store information from any language. Unicode defines properties for each character, standardizes script behavior, provides a standard algorithm for bi-directional text, and defines cross-mappings to other standards. Unicode provides a unique code value for every character, regardless of the platform, program, or language.

The Unicode standard primarily encodes scripts rather than languages. That is, where more than one language shares a set of symbols that have a historically related derivation, the union of the set of symbols of each such language is unified into a single collection identified as a single script. In many cases, a single script may serve to write tens or even hundreds of languages (e.g. the Latin script). In other cases only one language employs a particular script (e.g. Hangul).

Unicode defines codes for characters used in every major language written today. Unicode covers all the languages that can be written in the following scripts: Latin; Greek; Cyrillic; Armenian; Hebrew; Arabic; Syriac; Thaana; Devanagari; Bengali; Gurmukhi; Oriya; Tamil; Telegu; Kannada; Malayalam; Sinhala; Thai; Lao; Tibetan; Myanmar; Georgian; Hangul; Ethiopic; Cherokee; Canadian-Aboriginal Syllabics; Ogham; Runic; Khmer; Mongolian; Han (Japanese, Chinese, Korean ideographs); Hiragana; Katakana; Bopomofo and Yi .

It also includes punctuation marks, diacritics, mathematical symbols, technical symbols, musical symbols, arrows, dingbats, and so forth. In all, the Unicode Standard (as of version 3.2) provides support for over 95,000 characters from the world's alphabets, ideograph sets, and symbol collections.

Many software and hardware vendors have adopted the Unicode standard. Many operating systems and browsers now support Unicode. Unicode is required by modern standards such as XML, Java, JavaScript, LDAP, CORBA 3.0, WML etc. It is also synchronized with the ISO/IEC 10646 standard. For more information, please refer to the Unicode Standard official web site:

<http://www.unicode.org/unicode/standard/standard.html>

## **Supplementary Characters**

The initial version of Unicode used a 2-byte encoding format; by using 16 bits for every code point, a total of 65,536 characters could be represented. This is not sufficient to represent all characters currently in use worldwide. For example, the Chinese speaking community alone uses over 55,000 characters. For languages like Chinese, Japanese and Korean there are still tens of thousands of ideograms that are not yet encoded. And even though many of these are rarely used characters, they are still present in documents that need to be preserved electronically.

To meet this requirement, the Unicode Standard defines supplementary characters. The standard allocates 2048 code points in two groups of 1024 to be used in pairs to represent additional characters. By taking a pair of code points (also known as surrogate pairs), drawing one from each group of 1024 to represent a single character, an additional 1,048,576 characters can be defined. The first batch of the supplementary characters, 44,944 of them were added in the Unicode standard 3.1 (released in March 2001). Together with the 49,194 characters that existed in Unicode 3.0, a total of 95,221 characters are now encoded in Unicode 3.2. This introduced more complexity into the Unicode standard, but the complexity is still far less than managing a large number of different encoding schemes.

## Unicode Encodings

As with many technologies, Unicode has more than one implementation standard. Here are the common Unicode encoding formats, i.e. ways in which characters are represented by binary codes.

- UTF-8
- UCS-2
- UTF-16

Converting between the different Unicode encoding formats is a simple bit-wise operation that is defined in the Unicode standard. This conversion can be performed by using an algorithm, so an expensive mapping table is unnecessary.

### UTF-8 Encoding

UTF-8 is the 8-bit encoding of Unicode. It is a variable-width encoding and also a strict superset of 7-bit ASCII. A strict superset means that each and every character in 7-bit ASCII is available in UTF-8 with the same corresponding codepoint value. One Unicode character can be 1 byte, 2 bytes, 3 bytes or 4 bytes in this encoding. Characters from the European scripts are represented in either 1 or 2 bytes; characters from most Asian scripts are represented in 3 bytes, while supplementary characters are represented in 4 bytes.

UTF-8 is the Unicode encoding used on UNIX platforms, HTML and most Internet browsers.

The main benefits of UTF-8 are: -

- More compact storage requirement for European scripts. Since UTF-8 is a strict superset of 7-bit ASCII, in general European data will occupy less storage on disk and in memory.

- Ease of migration. Since 7-bit ASCII data remains the same in UTF-8, data conversion effort between ASCII based character set and UTF-8 are reduced significantly.

### UCS-2 Encoding

UCS-2 encoding is a fixed width 16-bit encoding of Unicode, where each character is 2 bytes in size regardless of the script. UCS-2 is the Unicode encoding used by Java and Microsoft Windows NT 4.0. UCS-2 can support Unicode characters defined up to Unicode standard 3.0 only, so there is no support for supplementary characters.

The main benefits of UCS-2 are: -

- More compact storage requirement for Asian scripts, because every character is represented in 2 bytes.
- String processing will be faster because all characters are of the same width.
- Better compatibility with Java and Microsoft clients

### UTF-16 Encoding

UTF-16 encoding is the 16-bit encoding of Unicode. UTF-16 is basically an extension of UCS-2, providing support for the new supplementary characters defined in Unicode 3.1 as pairs of UCS-2 code points.

One Unicode character can be 2 bytes or 4 bytes in this encoding. Characters from both European (including ASCII) and most Asian scripts are represented in 2 bytes. Supplementary characters are represented in 4 bytes. UTF-16 is the main Unicode encoding used by Microsoft Windows 2000.

The benefits of UTF-16 are: -

- More compact storage requirement for Asian scripts. Since the majority of the commonly used Asian characters are represented in 2 bytes in UTF-16 (whereas the same characters require 3 bytes each in UTF-8), UTF-16 will occupy less storage on disk and in memory.
- Better compatibility with Java and Microsoft clients.

Figure 1 shows some characters and their character codes in UTF-8, UCS-2 and UTF-16 encodings. The last character is a treble clef (a music symbol), a supplementary character that has been added to the Unicode 3.1 standard.

Character	UTF-8	UCS-2	UTF16
-----------	-------	-------	-------

A	41	0041	0041
c	63	0063	0063
Æ	C3 86	00C6	00C6
Ö	C3 B6	00F6	00F6
☺	DA B2	06B2	06B2
☹	E4 BA 9C	4E9C	4E9C
♫	F0 9D 84 9E	N/A	D834 DD1E

## UNICODE AND ORACLE

Oracle started supporting Unicode as a database character set in Oracle7. Here is a summary of the Unicode character sets supported in Oracle.

<i>Character Set</i>	<i>Supported in RDBMS</i>	<i>Unicode Encoding</i>	<i>Unicode Versions</i>	<i>Database Char set</i>	<i>National Char set</i>
AL24UTFFSS	7.2 - 8i	UTF-8	1.1	Y	N
UTF8	8.0 – 10g	UTF-8	2.1 <sup>1</sup> 3.0 <sup>2</sup>	Y	Y*
UTFE	8.0 – 10g	UTF-8	2.1 <sup>1</sup> 3.0 <sup>2</sup>	Y	N
AL32UTF8	9i – 10g	UTF-8	3.0 (9iR1) 3.1 (9iR2) 3.2 (10gR1)	Y	N
AL16UTF16	9i – 10g	UTF-16	3.0 (9iR1) 3.1 (9iR2) 3.2 (10gR1)	N	Y*

<sup>1</sup> RDBMS 8.0.x to 8.1.6

<sup>2</sup> RDBMS 8.1.7 to 9iR2

\* Oracle9i and Oracle Database 10g only

### AL24UTFFSS

AL24UTFFSS was the first Unicode character set supported by Oracle. It was introduced in RDBMS 7.2 as a Unicode database character set. AL24UTFFSS is an acronym for the multi-byte Unicode character encoding scheme UTF-FSS. Before version 7.2, UTF-8 was called UTF-2, and previously to that it was called UTF-FSS.

The AL24UTFFSS encoding scheme was based on the Unicode standard 1.1, which is now obsolete. AL24UTFFSS has been desupported in Oracle9i. The migration path for an existing AL24UTFFSS database is to upgrade to UTF8 prior to upgrading to Oracle9i.

## **UTF8**

UTF8 was the UTF-8 encoded character set in Oracle 8 and 8*i*. This was the first character set that broke the standard Oracle naming convention of <Language><bit size><encoding> (following the convention, this character set should have been called AL24UTF8 where AL stands for All Languages).

It was thought at the time that this was going to be the final UTF-8 encoded character set in Oracle. It followed the Unicode standard version 2.1 between Oracle 8.0 and 8.1.6, and was upgraded to Unicode version 3.0 in both 8.1.7 and 9*i*. To maintain compatibility with existing installations, this character set will remain at Unicode version 3.0 in future Oracle releases.

Although specific supplementary characters were not assigned to Unicode until version 3.1, the allocation for these characters were already defined in version 3.0. So if supplementary characters are inserted in a UTF8 database, it will not corrupt the actual data inside the database. They will be treated as two separate undefined characters occupying 6 bytes in storage. Oracle recommends that customers switch to AL32UTF8 or AL16UTF16 for full supplementary character support.

## **UTFE**

This is the UTF8 database character set for the EBCDIC platforms. It has the same properties as UTF8 on ASCII based platforms.

This EBCDIC Unicode transformation format is documented in Unicode Technical Report #16 - UTF-EBCDIC. A list of current Unicode Technical Reports is found on <http://www.unicode.org/unicode/reports/>

## **AL32UTF8**

This is the UTF-8 encoded character set introduced in Oracle9*i*. AL32UTF8 is the database character set that supports the latest version (3.2) of the Unicode Standard; it also provides support for the newly defined supplementary characters. All supplementary characters are stored as 4 bytes.

## **AL16UTF16**

This is the first UTF-16 encoded character set in Oracle. It was introduced in Oracle9*i* as the default national character set and continues to be in Oracle Database 10g. The National character set determines the character set of the SQL NCHAR datatypes (NCHAR, NVARCHAR2 and NCLOB), whereas the database character set governs the encoding of the SQL CHAR datatypes (CHAR, VARCHAR2, LONG, CLOB).

AL16UTF16 is the national character set that supports the latest version of the Unicode Standard (3.2); it also provides support to the newly defined supplementary characters. All supplementary characters are stored as 4 bytes.

## COMMON QUESTIONS AND ANSWERS

### *Is there a down side to Unicode?*

Although Unicode offers many benefits, there are some tradeoffs that come with its use. For example, if your data can be represented in a single byte character set, then use of the UTF-16 encoding format will double the storage requirements. Customers currently storing double byte Asian characters who migrate to UTF-8 will see storage requirements expand from 2 bytes / character to 3 bytes / character, an increase of 50%. In both cases the data expansion will increase the amount of data passed across the network, maintained in memory etc. The rate of data expansion can be reduced significantly by choosing the most appropriate Unicode encoding, depending on your language requirements.

Multi byte character sets such as Unicode can also impose some performance overhead in string manipulations. This is due to the nature of multi byte encoding; since the number of bytes required to represent a character varies in size, the character-processing algorithm needs to look ahead to determine where character boundaries lie. Fixed-width character sets, including all single-byte character sets, can always manipulate characters at known byte boundaries.

If you are already running with a multi byte database character set, the performance of Unicode processing should be similar.

### *When should Unicode be used?*

In general, you need to use Unicode if there is no other character set available that contains all the characters you need to represent.

For example, Unicode could be a good fit for a global company whose data is fragmented across multiple local systems. Using Unicode this company can deploy a single system and a single data center, reducing IT operations and allowing users to have access to complete global information. Another situation where Unicode would be useful is in the deployment of a corporate web site that must service customers from all over the world.

### *Should I deploy Unicode using Unicode database or Unicode datatype?*

You can create a Unicode database that allows you to store UTF-8 encoded characters as SQL CHAR datatypes (CHAR, VARCHAR2, CLOB, and LONG). If you prefer to implement Unicode support incrementally or you only need to support multilingual data in selective columns, you can store Unicode data in SQL NCHAR datatypes (NCHAR, NVARCHAR2, and NCLOB). The SQL NCHAR datatypes are called Unicode datatypes because they are used for storing Unicode data only.

There are many factors in determining the most appropriate Unicode solution to meet your business needs. Under some circumstances it may be beneficial to deploy both a Unicode database with Unicode datatypes in one database. For a

detailed discussion on this topic, please refer to, *Supporting Multilingual Databases with Unicode* in the Oracle Database 10g Globalization Support Guide.

***Can I store Unicode data using NCHAR prior to Oracle9i?***

No, NCHAR types in Oracle8 and Oracle8i were designed to support special Oracle specific fixed-width Asian character sets, that were introduced to provide higher performance processing of Asian character data. Examples of these character sets are: - JA16EUCFIXED, JA16SJISFIXED, ZHT32EUCFIXED etc.

No Unicode character set is supported as the national character set prior to Oracle9i.

***Which Unicode encoding should I pick for supporting Unicode datatype?***

For a detailed discussion on this topic, please refer to, *Supporting Multilingual Databases with Unicode* in the Oracle Database 10g Globalization Support Guide.

***What is the best way to migrate my existing columns to the Unicode datatype?***

This topic is covered in the white paper **Migration to Unicode Datatypes for Multilingual Databases and Applications in Oracle9i** available on OTN. <http://technet.oracle.com/tech/globalization/content.html>

***What is the difference between UCS-2 and UTF-16?***

UCS-2 is a fixed width Unicode encoding where each Unicode character is represented in 2 bytes. UTF-16 is a strict superset of UCS-2. It supports supplementary characters (surrogate pairs requiring 4 bytes per character) as defined in the latest Unicode standard 3.2.

***Why is AL16UTF16 not available as a database character set?***

The database character set is used to identify and to hold SQL, SQL metadata, and PL/SQL source code. It must have either single byte EBCDIC or single byte 7-bit ASCII as a subset, whichever is native to the deployment platform. Therefore, it is not possible to use a fixed-width, multi byte character set (such as UTF-16) as the database character set.

***What is the difference between UTF-8, UTF8 and AL32UTF8?***

UTF-8 is a variable-width Unicode encoding and also a strict superset of 7-bit ASCII. One Unicode character can be 1 byte, 2 bytes, 3 bytes or 4 bytes in UTF-8. UTF8 (without the dash) is the name of the Oracle character set introduced in Oracle 8.0 that follows the UTF-8 encoding. UTF8 supports UTF-8 encoding up to Unicode standard 3.0 only. AL32UTF8 is the Oracle Unicode character set that supports the supplementary characters defined in the latest Unicode standard 3.2.

***Why didn't Oracle upgrade the definition of UTF8 in Oracle9i to support supplementary characters, instead of creating yet another UTF-8 character set?***

UTF8 was originally designed back in Oracle8.0, when there was no concept of supplementary characters, and so has a maximum of 3 bytes per character.

If the definition of UTF8 were changed in Oracle9i, it would cause backward compatibility issues with pre-Oracle9i clients running on the old UTF8 definitions.

***Will I need to change my Unicode database character set again in the next major release of Oracle?***

This depends greatly on the future direction of the Unicode Standard. With the introduction of the surrogate pairs, over 1 million additional characters can be defined as supplementary characters. This should be enough to cover all the languages of the World and no significant architecture changes are expected in future versions of the Unicode Standard.

Our plan is to enhance AL32UTF8 and AL16UTF16 as necessary to support future versions of the Unicode standard.

***Are supplementary characters common for day-to-day use?***

The majority of the 45-thousand+ Supplementary characters are the rarely used CJK (Chinese, Japanese and Korean) characters; the remaining characters are Gothic, Old Italic, Musical and Mathematical symbols. Depending on the languages that you need to support and the nature of your application, these characters may not be very common for general day-to-day use. However it may be difficult to prevent users from inserting these characters once they become commonly available on their operating systems.

***What happens to the supplementary characters that get inserted into a UTF8 database?***

They will be treated as two separate undefined UTF8 characters occupying 6 bytes of storage. Although supplementary characters can be stored and retrieved in a UTF8 database, manipulation of these characters may yield unexpected results. It can also cause problems when communicating with other technology stacks such as Java, HTML etc, since 6 byte UTF8 characters are not recognized as an official UTF-8 encoding. (You may be able to work around this by setting the client NLS\_LANG character set to AL32UTF8, so that the 6 byte format will get converted to the 4 byte format.)

This supplementary character transformation format is documented in Unicode Technical Report #26 - Compatibility Encoding Scheme for UTF-16: 8-Bit (CESU-8). A list of current Unicode Technical Reports is found on <http://www.unicode.org/unicode/reports/>

Oracle recommends that customers switch to AL32UTF8 or AL16UTF16 for full supplementary character support.

### ***How I can upgrade from UTF8 to AL32UTF8?***

Since AL32UTF8 is a strict superset of UTF8, you can do this by issuing the ALTER DATABASE CHARACTER SET command. If you have 6 byte UTF8 supplementary characters in your UTF8 database, you will need to do an export and import of those data cells to convert the 6 byte supplementary characters into the 4 byte format.

As of today, most of the known operating systems do not yet provide support for the input and output of supplementary characters. It is therefore unlikely that any 6 bytes representation of supplementary characters will be present in current databases. This may change once support for the supplementary characters becomes widely available.

To assist you in determining whether you have any 6 byte supplementary characters in your UTF8 database, your data can be verified by using the character set scanner. Please refer to, *Character Set Scanner Utilities* in the Oracle Database 10g Globalization Support Guide.

### ***Why was AL24UTFFSS deprecated in Oracle9i?***

AL24UTFFSS was introduced with Oracle7 as the Unicode character set supporting UTF-8 encoding scheme based on the Unicode standard 1.1, which is now obsolete.

In Oracle9i, the supported Unicode database Character sets are AL32UTF8 and UTF8, which support the Unicode standard 3.0. The migration path for existing AL24UTFFSS database is to upgrade to UTF8 prior to upgrading to Oracle9i.

### ***Can I use the ALTER DATABASE CHARACTER SET command to migrate from AL24UTFFSS to UTF8 in Oracle8?***

No. Unicode standard 2.0 or greater is not backward compatible with Unicode standard 1.1 because the code values for the Korean Hangul characters defined in Unicode standard 1.1 were relocated in later versions. A new Unicode database character set (UTF8) was created in Oracle8 to support this change as defined in Unicode standard 2.1.

The export and import utilities will handle the data conversions between the different Unicode standard versions supported in AL24UTFFSS and UTF8. To assist you in determining whether you have any Hangul characters in your AL24UTFFSS database, Oracle recommends verifying your data by using the Character Set Scanner. Please refer to, *Character Set Scanner Utilities* in the Oracle Database 10g Globalization Support Guide.

***What about the migration from AL24UTFFSS to AL32UTF8 in Oracle9i?***

Since AL24UTFFSS is not a valid database character set in Oracle9i, the migration path to AL32UTF8 is to upgrade to UTF8 prior to upgrading to 9i.

Alternatively this can be achieved by importing a AL24UTFFSS .dmp file from your earlier releases into a AL32UTF8 database.

***Can my pre-Oracle9i database client software communicate with the 2 new Unicode character sets AL32UTF8 & AL16UTF16 in Oracle9i?***

There are interoperability issues that can affect Oracle8/8i clients connecting to an Oracle9i instance using these new character sets. As these character sets do not exist in Oracle8/8i, applications need client side patches in order to make sense of data stored using AL32UTF8 (CHAR, VARCHAR2, CLOB, LONG columns) and AL16UTF16 (NCHAR, NVARCHAR2, NCLOB) in Oracle9i.

For 8.0.x to 8.1.5 based Oracle clients to work with an AL32UTF8 database, a patch based on bug 897420 must be applied. This patch is not required for Oracle clients running 8.1.6 or above.

For 8.0.x to 8.1.x Oracle clients that must work with AL16UTF16 NCHAR columns, a patch for base bug 1634613 must be applied. This is only relevant to applications, which make use of SQL NCHAR datatypes (NCHAR NVARCHAR2 or NCLOB).

Please contact Oracle Support Services to find out if a patch is available for any particular platform and version.

**SUMMARY**

Unicode support has been available to customers using Oracle databases for a long time. It was first introduced as a database character set in Oracle RDBMS 7.2, supporting Unicode standard version 1. Starting in Oracle9i, Oracle adds to the already extensive Unicode features by providing support for the 2 most popular encoding forms of Unicode UTF-8 and UTF-16. Two new Unicode character sets were added to provide support for the Supplementary characters introduced in the Unicode standard version 3.1. Unicode standard version 3.2 is now supported in Oracle Database 10g

As the Unicode standard has evolved, Oracle has consistently enhanced our products to comply with the latest requirements. With each release of the database, new Unicode features are introduced to provide comprehensive coverage for meeting our customers globalization needs.



Oracle Unicode database support  
October 2001  
Author: Simon Law  
Contributing Authors:

Oracle Corporation  
World Headquarters  
500 Oracle Parkway  
Redwood Shores, CA 94065  
U.S.A.

Worldwide Inquiries:  
Phone: +1.650.506.7000  
Fax: +1.650.506.7200  
[www.oracle.com](http://www.oracle.com)

Oracle Corporation provides the software  
that powers the internet.

Oracle is a registered trademark of Oracle Corporation. Various  
product and service names referenced herein may be trademarks  
of Oracle Corporation. All other product and service names  
mentioned may be trademarks of their respective owners.

Copyright © 2000 Oracle Corporation  
All rights reserved.