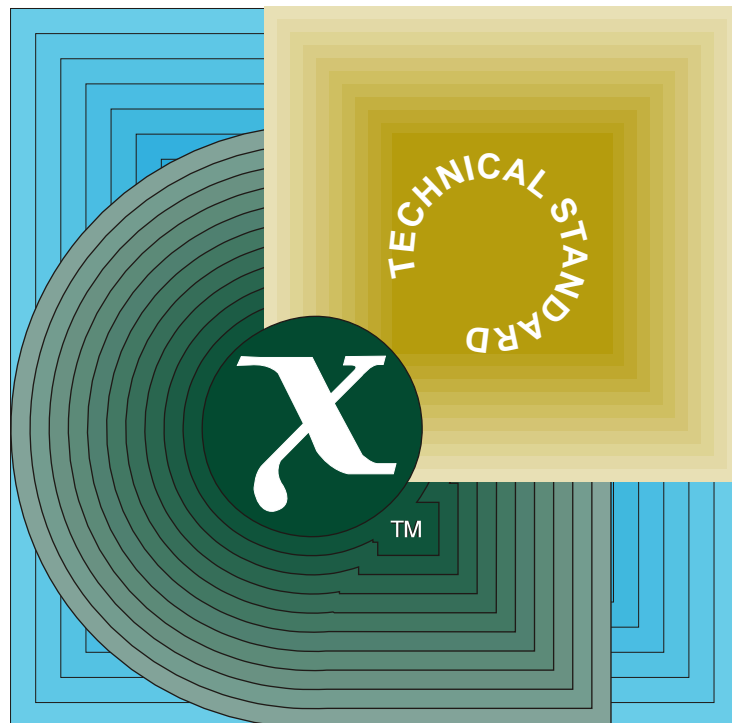


Technical Standard

File System Safe UCS Transformation Format (UTF-8)



THE *Open* GROUP

[This page intentionally left blank]

X/Open CAE Specification

File System Safe UCS Transformation Format (UTF-8)

X/Open Company Ltd.



© *March 1995, X/Open Company Limited*

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of the copyright owners.

X/Open CAE Specification

File System Safe UCS Transformation Format (UTF-8)

ISBN: 1-85912-082-2

X/Open Document Number: C501

Published by X/Open Company Ltd., U.K.

Any comments relating to the material contained in this document may be submitted to X/Open at:

X/Open Company Limited

Apex Plaza

Forbury Road

Reading

Berkshire, RG1 1AX

United Kingdom

or by Electronic Mail to:

XoSpecs@xopen.co.uk

/ Contents

Chapter 1	Introduction.....	1
1.1	Background.....	1
1.2	Scope.....	1
Chapter 2	Transformation Format.....	3
2.1	Criteria	3
2.2	Specification	4
2.3	Example	6
	Glossary	9
	Index.....	11
List of Figures		
2-1	Transformation Format.....	4

Preface

X/Open

X/Open is an independent, worldwide, open systems organisation supported by most of the world's largest information systems suppliers, user organisations and software companies. Its mission is to bring to users greater value from computing, through the practical implementation of open systems.

X/Open's strategy for achieving this goal is to combine existing and emerging standards into a comprehensive, integrated, high-value and usable open system environment, called the Common Applications Environment (CAE). This environment covers the standards, above the hardware level, that are needed to support open systems. It provides for portability and interoperability of applications, and so protects investment in existing software while enabling additions and enhancements. It also allows users to move between systems with a minimum of retraining.

X/Open defines this CAE in a set of specifications which include an evolving portfolio of application programming interfaces (APIs) which significantly enhance portability of application programs at the source code level, along with definitions of and references to protocols and protocol profiles which significantly enhance the interoperability of applications and systems.

The X/Open CAE is implemented in real products and recognised by a distinctive trade mark — the X/Open brand — that is licensed by X/Open and may be used on products which have demonstrated their conformance.

X/Open Technical Publications

X/Open publishes a wide range of technical literature, the main part of which is focussed on specification development, but which also includes Guides, Snapshots, Technical Studies, Branding/Testing documents, industry surveys, and business titles.

There are two types of X/Open specification:

- *CAE Specifications*

CAE (Common Applications Environment) specifications are the stable specifications that form the basis for X/Open-branded products. These specifications are intended to be used widely within the industry for product development and procurement purposes.

Anyone developing products that implement an X/Open CAE specification can enjoy the benefits of a single, widely supported standard. In addition, they can demonstrate compliance with the majority of X/Open CAE specifications once these specifications are referenced in an X/Open component or profile definition and included in the X/Open branding programme.

CAE specifications are published as soon as they are developed, not published to coincide with the launch of a particular X/Open brand. By making its specifications available in this way, X/Open makes it possible for conformant products to be developed as soon as is practicable, so enhancing the value of the X/Open brand as a procurement aid to users.

- *Preliminary Specifications*

These specifications, which often address an emerging area of technology and consequently are not yet supported by multiple sources of stable conformant implementations, are released in a controlled manner for the purpose of validation through implementation of products. A Preliminary specification is not a draft specification. In fact, it is as stable as X/Open can make it, and on publication has gone through the same rigorous X/Open development and review procedures as a CAE specification.

Preliminary specifications are analogous to the *trial-use* standards issued by formal standards organisations, and product development teams are encouraged to develop products on the basis of them. However, because of the nature of the technology that a Preliminary specification is addressing, it may be untried in multiple independent implementations, and may therefore change before being published as a CAE specification. There is always the intent to progress to a corresponding CAE specification, but the ability to do so depends on consensus among X/Open members. In all cases, any resulting CAE specification is made as upwards-compatible as possible. However, complete upwards-compatibility from the Preliminary to the CAE specification cannot be guaranteed.

In addition, X/Open publishes:

- *Guides*

These provide information that X/Open believes is useful in the evaluation, procurement, development or management of open systems, particularly those that are X/Open-compliant. X/Open Guides are advisory, not normative, and should not be referenced for purposes of specifying or claiming X/Open conformance.

- *Technical Studies*

X/Open Technical Studies present results of analyses performed by X/Open on subjects of interest in areas relevant to X/Open's Technical Programme. They are intended to communicate the findings to the outside world and, where appropriate, stimulate discussion and actions by other bodies and the industry in general.

- *Snapshots*

These provide a mechanism for X/Open to disseminate information on its current direction and thinking, in advance of possible development of a Specification, Guide or Technical Study. The intention is to stimulate industry debate and prototyping, and solicit feedback. A Snapshot represents the interim results of an X/Open technical activity. Although at the time of its publication, there may be an intention to progress the activity towards publication of a Specification, Guide or Technical Study, X/Open is a consensus organisation, and makes no commitment regarding future development and further publication. Similarly, a Snapshot does not represent any commitment by X/Open members to develop any specific products.

Versions and Issues of Specifications

As with all *live* documents, CAE Specifications require revision, in this case as the subject technology develops and to align with emerging associated international standards. X/Open makes a distinction between revised specifications which are fully backward compatible and those which are not:

- a new *Version* indicates that this publication includes all the same (unchanged) definitive information from the previous publication of that title, but also includes extensions or additional information. As such, it *replaces* the previous publication.

- a new *Issue* does include changes to the definitive information contained in the previous publication of that title (and may also include extensions or additional information). As such, X/Open maintains *both* the previous and new issue as current publications.

Corrigenda

Most X/Open publications deal with technology at the leading edge of open systems development. Feedback from implementation experience gained from using these publications occasionally uncovers errors or inconsistencies. Significant errors or recommended solutions to reported problems are communicated by means of Corrigenda.

The reader of this document is advised to check periodically if any Corrigenda apply to this publication. This may be done either by email to the X/Open info-server or by checking the Corrigenda list in the latest X/Open Publications Price List.

To request Corrigenda information by email, send a message to `info-server@xopen.co.uk` with the following in the Subject line:

```
request corrigenda; topic index
```

This will return the index of publications for which Corrigenda exist.

This Document

This document is a CAE Specification (see above). It is intended for systems programmers with experience in the C programming language.

Typographical Conventions

The following typographical conventions are used throughout this document:

- **Bold** font is used in text for filenames, keywords, type names, data structures and their members.
- *Italic* strings are used for emphasis or to identify the first instance of a word requiring definition. Italics in text also denote functions, which are shown as follows: *name()*.
- Code examples are shown in `fixed width` font.
- Ranges of values are indicated with parentheses or brackets as follows:
 - (a,b) means the range of all values from a to b, including neither a nor b
 - [a,b] means the range of all values from a to b, including a and b
 - [a,b) means the range of all values from a to b, including a, but not b
 - (a,b] means the range of all values from a to b, including b, but not a.

Trade Marks

UNIX[®] is a registered trade mark in the United States and other countries, licensed exclusively through X/Open Company Limited.

X/Open[®] is a registered trade mark, and the “X” device is a trade mark, of the X/Open Company Ltd.

Referenced Documents

The following standards are referenced in this specification:

Internationalisation Guide

X/Open Guide, July 1993, Internationalisation Guide, Version 2 (ISBN: 1-859120-02-4, G304).

ISO 8859

ISO 8859-*: 1987 Information Processing — 8-bit Single-byte Coded Graphic Character Sets — Parts 1 to 10 inclusive.

ISO C

ISO/IEC 9899:1990, Programming Languages — C (technically identical to ANSI standard X3.159-1989).

ISO/IEC 10646

ISO/IEC 10646-1:1993, Information Technology — Universal Multiple-Octet Coded Character Set (UCS) — Part 1: Architecture and Basic Multilingual Plane.

ISO POSIX-2

ISO/IEC 9945-2:1993, Information Technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities (identical to IEEE Std 1003.2-1992).

This chapter explains the need for a Universal Character Set Transformation Format.

1.1 Background

In the past, most information processing systems were based on the 7-bit ASCII coded character set (codeset). Most languages require characters additional to those in the ASCII codeset. The ISO 8859 standard defines codesets, each of which caters for a group of languages (the section on Character Sets in the **Internationalisation Guide** gives details). All the ISO 8859 standard codesets contain ASCII as a subset.

The referenced ISO/IEC 10646-1 standard offers a Universal Character Set (UCS) that provides the capability to encode multi-lingual text within a single codeset, by using either two octets (*UCS-2*) or four octets (*UCS-4*). ASCII-based operating systems need to cope with the representation and handling of the large number of characters possible with this new standard. The encoding of the ISO/IEC 10646-1 standard does not protect the null byte (byte with all bits zero) nor the ASCII slash character (/). An annex to the ISO/IEC 10646-1 standard originally specified a UCS Transformation Format called UTF-1 which did not protect the slash character (/). The ISO/IEC 10646-1 standard subsequently deprecated the use of UTF-1 and added a normative annex for a different transformation format called UTF-8. UTF-8 specified in the ISO/IEC 10646-1 standard annex is identical to the UTF-8 specified in this document, and was originally defined by the X/Open-UniForum Joint Internationalisation Group (JIG) under the designation FSS-UTF.

1.2 Scope

There is considerable work required to adapt existing programming languages, operating systems and utilities to the requirements of the ISO/IEC 10646-1 standard; that is outside the scope of this specification.

This document addresses the problem of the handling of the UCS by historical operating systems and utilities. In particular, this specification defines how data must be represented within the file system by specifying a transformation format that is compatible with UNIX systems, supporting multi-lingual text in a single encoding.

This specification is based on the assumption that there is a requirement to maintain the existing operating system software investment, while at the same time taking advantage of the use of the large number of characters provided by the UCS. The objective is to define a transformation format that is usable on historical operating systems' file systems in a non-disruptive manner.

Transformation Format

This chapter describes the criteria met by UTF-8 and gives the UTF-8 specification.

2.1 Criteria

UTF-8 meets the following criteria:

- It is compatible with historical file systems (which disallow the null byte and the ASCII slash character as part of the filename).
- It is compatible with existing ASCII-based programs. Converting existing ASCII-based programs will be much easier if ASCII characters do not occur in a multi-byte encoding. A UTF-8 representation of a non-ASCII character contains no ASCII code values. If the UCS value is in the range [0x00,0x7F], the transformation is also in this range; otherwise, the transformed byte sequence does not contain any bytes in the range [0x00,0x7F].
- It is easy to convert from and to UCS.
- The first byte indicates the number of bytes to follow in a multi-byte sequence.
- UTF-8 is not extravagant in terms of the number of bytes used for encoding.
- It is possible to find the start of a character efficiently starting from an arbitrary location in a byte stream.
- UTF-8 is a multi-byte encoding as defined by Section 5.2.1.2 of the ISO C standard.

2.2 Specification

UTF-8 encodes UCS values in the range [0,0x7FFFFFFF] using multi-byte characters of lengths 1, 2, 3, 4, 5 and 6 bytes.

For all encodings of more than one byte, the initial byte specifies the number of bytes used by setting 1 in the equivalent number of high-order bits. The next most significant bit is always 0. For example, a 2-byte sequence starts with 110 and a 6-byte sequence starts with 111110.

The following table shows the format of the first byte of a character; the free bits available for coding the character are indicated by *x*.

Single byte	0XXXXXXXX	seven free bits
First of two bytes	110XXXXXX	five free bits
First of three bytes	1110XXXXX	four free bits
First of four bytes	11110XXX	three free bits
First of five bytes	111110XX	two free bits
First of six bytes	1111110X	one free bit

All subsequent bytes in multi-byte characters have the following format:

10XXXXXX six free bits

For all subsequent bytes the two most significant bits are set to 10. Therefore, every byte that does not start with 10 is the start of a UCS character sequence.

Figure 2-1 illustrates UTF-8:

No. of bytes in char.	No. of bits available for coding	Byte Sequence in Binary
1	7	0VVVVVVV
2	11	110VVVVV 10VVVVVV
3	16	1110VVVV 10VVVVVV 10VVVVVV
4	21	11110VVV 10VVVVVV 10VVVVVV 10VVVVVV
5	26	111110VV 10VVVVVV 10VVVVVV 10VVVVVV 10VVVVVV
6	31	1111110V 10VVVVVV 10VVVVVV 10VVVVVV 10VVVVVV 10VVVVVV

Figure 2-1 Transformation Format

The UCS value is the concatenation of the *v* bits in the multi-byte encoding. When there are multiple ways to encode a value (for example, UCS 0), only the shortest encoding is legal. The range of values possible with each type of byte sequence is shown in the following table.

No. of bytes in sequence	Hexadecimal Value	
	Minimum	Maximum
1	00000000	0000007F
2	00000080	000007FF
3	00000800	0000FFFF
4	00010000	001FFFFFF
5	00200000	03FFFFFF
6	04000000	7FFFFFFF

The following table defines the transformation from UCS-4 to UTF-8 (the expression syntax used in the table is the same as the C language):

$0 \leq x \leq 0x7F$	1st byte: x
$0x80 \leq x \leq 0x7FF$	1st byte: $0xC0 \mid (x \gg 6)$ 2nd byte: $0x80 \mid (x \& 0x3F)$
$0x800 \leq x \leq 0xFFFF$	1st byte: $0xE0 \mid (x \gg 12)$ 2nd byte: $0x80 \mid ((x \gg 6) \& 0x3F)$ 3rd byte: $0x80 \mid (x \& 0x3F)$
$0x10000 \leq x \leq 0x1FFFFF$	1st byte: $0xF0 \mid (x \gg 18)$ 2nd byte: $0x80 \mid ((x \gg 12) \& 0x3F)$ 3rd byte: $0x80 \mid ((x \gg 6) \& 0x3F)$ 4th byte: $0x80 \mid (x \& 0x3F)$
$0x200000 \leq x \leq 0x3FFFFFFF$	1st byte: $0xF8 \mid (x \gg 24)$ 2nd byte: $0x80 \mid ((x \gg 18) \& 0x3F)$ 3rd byte: $0x80 \mid ((x \gg 12) \& 0x3F)$ 4th byte: $0x80 \mid ((x \gg 6) \& 0x3F)$ 5th byte: $0x80 \mid (x \& 0x3F)$
$0x4000000 \leq x \leq 0x7FFFFFFF$	1st byte: $0xFC \mid (x \gg 30)$ 2nd byte: $0x80 \mid ((x \gg 24) \& 0x3F)$ 3rd byte: $0x80 \mid ((x \gg 18) \& 0x3F)$ 4th byte: $0x80 \mid ((x \gg 12) \& 0x3F)$ 5th byte: $0x80 \mid ((x \gg 6) \& 0x3F)$ 6th byte: $0x80 \mid (x \& 0x3F)$

2.3 Example

The example below implements the ISO C standard `wctomb()` and `mbtowc()` functions, to demonstrate the algorithms for converting from UCS to UTF-8 and converting from UTF-8 to UCS. The example includes error checks, some of which may not be necessary for conformance:

```
typedef
struct
{
    int    cmask;
    int    cval;
    int    shift;
    long   lmask;
    long   lval;
} Tab;

static
Tab    tab[] =
{
    0x80,  0x00,  0*6,  0x7F,          0,          /* 1 byte sequence */
    0xE0,  0xC0,  1*6,  0x7FF,        0x80,        /* 2 byte sequence */
    0xF0,  0xE0,  2*6,  0xFFFF,       0x800,       /* 3 byte sequence */
    0xF8,  0xF0,  3*6,  0x1FFFFFF,    0x10000,     /* 4 byte sequence */
    0xFC,  0xF8,  4*6,  0x3FFFFFFF,   0x200000,    /* 5 byte sequence */
    0xFE,  0xFC,  5*6,  0x7FFFFFFF,   0x4000000,   /* 6 byte sequence */
    0,                                           /* end of table */
};

int
mbtowc(wchar_t *p, const char *s, size_t n)
{
    long l;
    int c0, c, nc;
    Tab *t;

    if(s == 0)
        return 0;

    if(*s == 0) {
        if (p != NULL)
            *p = 0;
        return 0;
    }

    nc = 0;
    if(n == 0)
        return -1;
    c0 = *s & 0xff;
    l = c0;
    for(t=tab; t->cmask; t++) {
        nc++;
        if((c0 & t->cmask) == t->cval) {
            l &= t->lmask;

```

```

        if(l < t->lval)
            return -1;
        if (p != NULL)
            *p = l;
            return nc;
    }
    if(n <= nc)
        return -1;
    s++;
    c = (*s ^ 0x80) & 0xFF;
    if(c & 0xC0)
        return -1;
    l = (l<<6) | c;
}
return -1;
}

int
wctomb(char *s, wchar_t wc)
{
    long l;
    int c, nc;
    Tab *t;

    if(s == 0)
        return 0;

    l = wc;
    nc = 0;
    for(t=tab; t->cmask; t++) {
        nc++;
        if(l <= t->lmask) {
            c = t->shift;
            *s = t->cval | (l>>c);
            while(c > 0) {
                c -= 6;
                s++;
                *s = 0x80 | ((l>>c) & 0x3F);
            }
            return nc;
        }
    }
    return -1;
}

```


/ Glossary

byte

An individually addressable unit of data storage that is equal to or larger than an octet, used to store a character or a portion of a character; see **character**. A byte is composed of a contiguous sequence of bits, the number of which is implementation-dependent. The least significant bit is called the *low-order* bit; the most significant is called the *high-order* bit. Note that this definition of *byte* deviates intentionally from the usage of *byte* in some international standards, where it is used as a synonym for *octet* (always eight bits). On a system based on the ISO POSIX-2, a byte may be larger than eight bits so that it can be an integral portion of larger data objects that are not evenly divisible by eight bits (such as a 36-bit word that contains four 9-bit bytes).

character

A sequence of one or more bytes representing a single graphic symbol or control code. This term corresponds to the ISO C standard term *multibyte character* (multi-byte character), where a single-byte character is a special case of a multi-byte character. Unlike the usage in the ISO C standard, *character* here has no necessary relationship with storage space, and *byte* is used when storage space is discussed.

character set

A finite set of different characters used for the representation, organisation or control of data.

coded character set (codeset)

A set of unambiguous rules that establishes a character set and the one-to-one relationship between each character of the set and its bit representation.

FSS-UTF

The File System Safe UCS Transformation Format originally developed by the X/Open- UniForum Joint Internationalisation Group (JIG), subsequently adopted by ISO as UTF-8 in a normative annex to the ISO/IEC 10646-1 standard.

null byte

A byte with all bits set to zero.

UCS

Universal Multiple-Octet Coded Character Set.

UCS-2

The ISO/IEC 10646-1 standard allows two basic forms for characters: Universal Coded Character Set 2 (UCS-2) and 4 (UCS-4). UCS-2 is also known as the Basic Multilingual Plane (BMP). Characters are encoded in the lower two octets (row and cell).

UCS-4

The ISO/IEC 10646-1 standard allows two basic forms for characters: Universal Coded Character Set 2 (UCS-2) and 4 (UCS-4). UCS-4 characters are encoded in the full four octets.

Index

ASCII	1
code values	3
slash character	3
byte	9
character	9
character set	9
coded character set (codeset)	9
codeset	1
conversion	
from UCS to UTF-8	3, 6
from UTF-8 to UCS	3, 6
file system	
historical	3
FSS-UTF	9
null byte	3, 9
operating system	1
transformation format	1, 3
criteria	3
UCS	1, 9
value	4
UCS-2	9
UCS-4	9
universal character set	1
UTF-1	1
UTF-8	1, 3
criteria	3
specification	4

