Unicode: What is it and how do I use it?

Abstract: *The rationale for Unicode and its design goals and detailed design principles are presented. The correspondence between Unicode and ISO/IEC 10646 is discussed, the scripts included or planned for inclusion in the two character set standards are listed. Some products that support Unicode and some applications that require Unicode are listed, then examples of how to specify Unicode characters in a variety of applications are given. Use of Unicode in SGML and XML applications is discussed, and the paper concludes with descriptions of the character encodings used with Unicode and ISO/IEC 10646 plus sources of further information are listed.*

Tony Graham
Senior Consultant
Mulberry Technologies, Inc.
17 West Jefferson Street, Suite 207
Rockville, MD 20850 U.S.A.
phone: 301/315-9631
email: info@mulberrytech.com
http://www.mulberrytech.com

Keywords: Unicode; ISO/IEC 10646; UTF-8; UTF-16; UCS.

1. What is Unicode?

Unicode is a character encoding standard published by, of all people, the Unicode Consortium. Unicode is designed to include all of the major scripts of the world in a simple and consistent manner. The Unicode standard also defines properties of the characters and algorithms for use in Unicode implementations.

Existing character encodings are a mixture of one-byte and two-byte encodings—Chinese, Japanese, and Korean encodings need two bytes per character, and most other coded character sets use only one byte per character. Most coded character sets are more or less compatible with ASCII for the first 127 character numbers, but above that it's anybody's game. Changes between character encodings are signalled by escape sequences embedded in the text stream, so to determine the encoding, and in some cases to determine how many bytes to read as a single character, you need to process the entire text stream up to that point. The net result is that one transmission error or one misstep in applying encoding-translation software can garble an entire document.

Unicode was born from frustration by software manufacturers with the fragmented, complicated, and contradictory character encodings in use around the world. The technical difficulties of dealing with different coded character sets meant that software had to be extensively localized before being released into different markets, which meant that the "other language" versions of software could be significantly delayed and also be signficantly different internally because of the character handling requirements. Not surprisingly, the same character handling changes had to be regrafted onto each new version of the base software before it could be released into other markets. Of course the character encoding is not the only aspect to be changed when localizing software, but it is a large component, and it is the aspect most likely to have a custom solution for each new language.

The Unicode work began in the late 1980s, and the Unicode Consortium was incorporated as Unicode, Inc. in 1991. Today the Consortium's members include many major software and hardware manufacturers plus. The current version of the Unicode Standard is version 2.1. Version 2.1 is issued as a Technical Report that lists additions and corrections from version 2.0 of the Standard. That document is available as a 950-page book that defines Unicode and lists every character with an example glyph. As an example of the problems of handling the world's scripts, in order to display every glyph, the book required specialized formatting software from five different companies.

The Unicode Standard defines a fixed-width 16-bit, uniform encoding scheme for written characters and text. The standard defines 38,887 distinct coded characters that includes characters for the major scripts of the world, as well as technical symbols in common use. The Unicode Standard is also code-for-code identical with ISO/IEC 10646-1:1993, although it does define more semantics for characters than does ISO/IEC 10646. The Unicode Technical Committee has a liaison membership with the ISO/IEC Working Group responsible for computer character sets.

2. Unicode Design Goals

The original design goals of the Unicode Standard were:

Universal	*The repertoire had to be large enough to encompass all characters likely to be used in general text interchange.*
Efficient	*Plain text, composed of a sequence of fixed width characters, is simple to parse, and software does not need to maintain state, look for special escape sequences, or search forward or backward through text to identify characters.*
Uniform	*A fixed-length character code allows efficient sorting, searching, display, and editing of text.*
Unambiguous	*Any given 16-bit value always represents the same character.*

3. Unicode Design Principles

Building on the design goals, the design of the Unicode Standard reflects the following ten design principles.

Sixteen-Bit Characters

All Unicode character codes are a uniform 16 bits. For compatibility with existing computer systems that can't handle 16-bit characters, the Unicode Standard defines the UTF-8 and UTF-7 formats for lossless transformation between Unicode characters and 8-bit and 7-bit character sequences, respectively.

Full Encoding

Using 16-bit characters allows over 65,000 code positions that can be assigned to characters. This far exceeds the expected requirements for all modern and most archaic languages. As of Unicode 2.1, there are still over 18,000 unassigned code positions.

Figure 1. Will the real "a" please step forward?

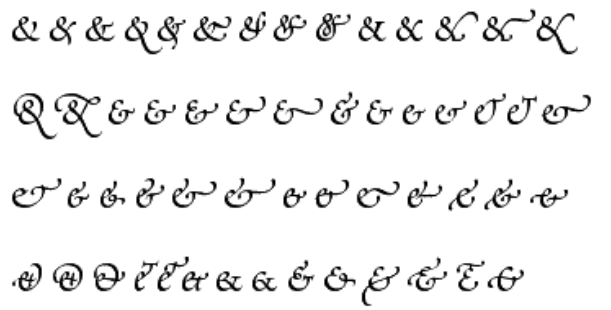

Figure 2. Ampersands

Characters, Not Glyphs

Unicode is concerned with characters, not glyphs. Characters are the smallest component of a written language, and glyphs are the representation of the written or displayed character. There is often a one-to-many or many-to-one relationship between characters and glyphs. Even in English, there are many different glyphs that can represent a single character (Figure 1), fonts that contain many glyphs for the same character (Figure 2), and in high-quality typesetting, single-glyph ligatures used for sequences of, for example, "f" and "i" or "f" and "l" characters (Figure 3).

Semantics

The Standard provides well-defined semantics for each character, including numeric, spacing, combination, and directionality properties.

The specification of these semantics is not included in ISO/IEC 10646, and the writers of the Unicode Standard see this as a major feature of Unicode over ISO/IEC 10646.

Plain Text

Plain Unicode is a sequence of character codes. Unicode does not currently define any codes for specifying language, font, etc., although there are character codes that provide hints about directionality of the text.

In Unicode terms, SGML and XML are "fancy text" or "higher-level protocols" since their markup

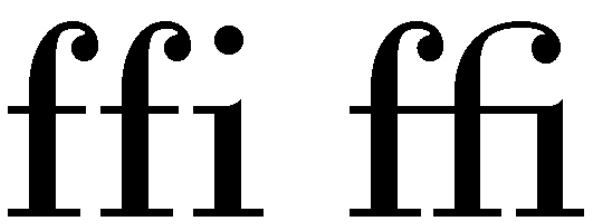

Figure 3. "ffi" and "ffi"ligature

represents additional data structures interspersed in the stream of plain Unicode characters.

The "plain text" nature of Unicode is set to change since a recent Unicode Technical Report **[TR7 98]** proposes the addition of language tags using characters on Plane 14 of ISO/IEC 10646. Since the language tags cannot be confused for textual content, parsing for them is, in theory, easier, than, say, using SGML or XML, and they are seen as a lighter-weight mechanism for specifying language.

Logical Order

Characters are stored in their logical order: in the sequence in which they are read, which is not always the sequence in which they are displayed. For example, Arabic and Hebrew are read right-to-left. Since the logical start of the right-to-left text is the character closest to the right margin, that character is the first character in the Unicode character stream, as shown in Figure 4.

The characters' directionality properties, and use of character codes specifying changes in direction when mixing characters of different dominant direction, provide sufficient information for correct rendering of the text.

In addition, the Unicode Standard specifies that combining marks always follow their base character. In contrast, existing encodings are not standardized, and some require the combining characters before the base character and some after.

Unification

Unicode unifies characters within scripts across languages so that characters with equivalent form are given a single code. For example, common letters, punctuation marks, symbols, and diacritics were each given one code. In addition, over 120,000 ideographs used in Chinese, Japanese, and Korean were unified to 20,902 character codes.

The exceptions are "compatibility characters" that could have been unified but remained as separate code positions, often in support of round-trip mapping between Unicode and an existing code set. For example, ISO 8859-1 includes the character "Ö", and Unicode includes "O" and " ¨" plus "Ö" as a compatibility character even though "Ö" could be

Display order:

Example of "tfel-ot-thgir" text

Logical order:

Example of "right-to-left" text

Figure 4. Bidirectional Ordering

Figure 5. Sample text with Unicode and Simplified Chinese fonts

remapped to the two-character sequence without loss of information.

Han Unification: The Downside

The downside of the unification process is that a "Unicode" font would use the same glyph for the same character number, not matter what glyph variations existed for the "un-unified" forms of the character. At best, this leads to homogenized fonts that use generic representations of the characters and please no-one, and at worst this leads to the occasional "wrong" character when the font uses the wrong glyph variant for a user' locale.

Figure 5 shows a small portion of a Simplified Chinese web page using three different fonts. Note how the Unicode font and the two Simplified Chinese fonts have differences that are visible even to the non-reader of Chinese.

Extra information can be added to the document, using the proposed language tags or a "higher-level protocol" such as XML markup and the `xml:lang` attribute, to specify the correct language and/or locale for the text, and this can be used to key the selection of fonts for representing the characters, but this leads to more complex software and detracts from the "Characters, not Glyphs" design principle.

Dynamic Composition

We have already seen "O" and " ¨" as an example of a base plus a combining character. Instead of allowing just "well-known" accented characters such "Ö", Unicode allows dynamic composition of accented forms where any base character plus any combining character (or sequence of combining characters) can make an accented form.

Equivalent Sequence

Since some character may be represented in precomposed or dynamically composed forms, the Unicode Standard defines equivalent sequences for each precomposed form. Since sequences of combining characters can follow a base character, the Standard

also defines a canonical ordering for combining characters.

Note that the Unicode Standard does not prescribe one particular internal representation of composed characters or one particular sequence of combining characters. Systems may choose to normalize Unicode text to one particular representation, and the W3C work on "Requirements for String Identity Matching and String Indexing" may standardize that for the entire Web, but in Unicode all sequences of characters are permitted.

Convertibility

Round-trip conversion between Unicode and many pre-existing standards is possible since each character has a unique correspondence with a sequence of one or more Unicode characters. When a base standard includes multiple variant forms of a single character, the variants are not unified so there will always be a mapping between Unicode and the base standard.

While accurate convertibility is guaranteed, many conversions require a mapping table since the corresponding Unicode characters may not be in the same sequence as in the base standard or a base standard character may map to a sequence of Unicode characters.

Guaranteeing convertibility has required compromises such as the inclusion of many compatibility characters but this does mean that Unicode can become a replacement or alternative for these pre-existing standards. It also means that an application can read or write using a pre-existing coded character set but be "Unicode inside".

4. Unicode and ISO/IEC 10646

The Unicode Standard is code-for-code compatible with ISO/IEC 10646-1:1993, Information Technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane.

ISO/IEC 10646 is a four-octect (32-bit) coded character set, although it currently only specifies characters in the Basic Multilingual Plane (BMP), and these can be represented with 16-bit characters. The BMP includes characters in general use in alphabetic, syllabic, and ideographic scripts together with various symbols and digits. ISO/IEC 10646 specifies the names and coded representation of these graphic characters. In addition, two coded representations are specified: UCS-4, the four-octet (32-bit) form of the UCS, and UCS-2, the two-octet (16-bit) BMP form of the UCS.

In ISO/IEC 10646 terms, the BMP is "Plane 00 of Group 00". Characters outside the BMP will require more than 16 bits for their coded representations. Although there are proposals for assignments in Plane 01, the ISO/IEC Working Group has not begun the formal process of creating Plane 01.

Timeline

The Unicode Standard and ISO/IEC 10646 began as independent efforts. They merged in 1992, and today the Unicode Technical Committee and the ISO/IEC Working Group operate in parallel.

1989	DP 10646 published, independent of Unicode
1990	Unicode 1.0 published
1990	DIS-1 10646, independent of Unicode
1991	Unicode and ISO 10646 agree to merge
1992	Unicode 1.0.1, modified for merger
1992	DIS-2 10646, merged with Unicode
1993	IS 10646-1:1993, merged standard
1993	Unicode 1.1, revised to match 10646-1:1993
1995	10646 amendments
1996	Unicode 2.0, revised to cover amendments
1998	Unicode 2.1, corrected errors, added euro

Unicode's Extra Semantics

To ISO/IEC 10646, the Unicode Standard is a "profile" that implements certain portions of ISO/IEC 10646. In particular, Unicode supports only the BMP, uses the UTF-16 coded representation (or UCS-2 if no surrogates[1] are used), and counts as "implementation level 3" since it includes both combining marks and precomposed characters.

To Unicode, ISO/IEC 10646 is a superset of the Unicode Standard since it can include more characters than can Unicode. However, the Unicode Standard defines additional character semantics beyond those defined in ISO/IEC 10646, which the Unicode Standard stresses as the major difference between the two standards.

In ISO/IEC 10646, the character name is the main resource for character semantics and cross-mapping between standards. The character names are identical in the Unicode Standard, but that Standard adds or includes alias names, usage annotations, character properties, conformance specifications, and tables of explicit mappings between the Unicode Standard and pre-existing standards.

1. Four-byte sequences used to represent characters outside the BMP.

5. What in the World is Included?

Fulfilling the goal of including the major scripts of the world, the Unicode Standard and ISO/IEC 10646 currently support the following:

Primary Scripts

- Arabic
- Armenian
- Bengali
- Bopomofo
- Cyrillic
- Devanagari
- Georgian
- Greek
- Gujarati
- Gurmkhi
- Han
- Hangul
- Hebrew
- Hiragana
- Kannada
- Katakana
- Latin
- Lao
- Malayalam
- Oriya
- Phonetic
- Tamil
- Telugu
- Thai
- Tibetan

Secondary Scripts

- Numbers
- General Diacritics
- General Punctuation
- General Symbols
- Mathematical Symbols
- Technical Symbols
- Dingbats
- Arrows, Blocks, Box Drawing Forms, and Geometric Shapes
- Miscellaneous Symbols
- Presentation Forms

What's in the Works?

Planned or proposed scripts from around the world (and out of this world) include:

- Ethiopic
- Cherokee
- Canadian Syllabics
- Runic
- Ogham
- Braille Pattern Symbols
- Yi
- Sinhala
- Thaana
- Khmer
- Burmese
- Syriac
- Western Musical Symbols
- Byzantine Musical Symbols
- Deseret Alphabet
- Shavian
- Etruscan
- Gothic
- Linear B
- Cypriot Syllabary
- Mongolian
- Cham
- Tai (Dai) scripts
- Glagolitic
- Coptic
- Buginese
- Old Hungarian Runic
- Phoenician
- Avestan
- Phillipine Scripts
- Basic Egyptian Hieroglyphics
- Meriotic
- Old Persian Cuneiform
- Ugaritic Cuneiform
- Tifinagh
- Tengwar
- Cirth
- tlhingan Hol (Klingon)
- Brahmi
- Javanese
- Old Permic
- Sinaitic
- South Arabian
- Pollard

6. Is it a Universal Character Set?

As we have seen, the Unicode Standard and ISO/IEC 10646 do support a multitude of scripts, but their estimated success or failure in representing the major scripts of the world varies depending on whose opinion you ask. To the majority of Japanese, the answer is a resounding "No", largely because of dissatisfaction with the Han unification process that merged Japanese ideographs with those of Chinese and Korean.

Table 1 How do I specify Ж

Standard or Application	Representation	Comment
Unicode	U+0416	An individual Unicode value is expressed as *U+nnnn*, where *nnnn* is the character's number expressed in hexadecimal notation.
	CYRILLIC CAPITAL LETTER ZHE	All Unicode characters have unique names, which are the same as those in the English language version of ISO/IEC 10646. Names contain only the uppercase letters A to Z, space, hyphen-minus, and occasionally digits. As this table shows, this makes it easy to generate computer-sensible identifiers for individual characters.
ISO/IEC 10646	0000 0416	UCS-4
	0416	UCS-2
	CYRILLIC CAPITAL LETTER ZHE	ISO/IEC 10646 assigns a unique name to each character.
Java	\u0416	Java defines a special Unicode escape sequence for representing the hexadecimal value of a Unicode character.
Pre-TC SGML	Ж	Prior to the WebSGML Adaptations TC, SGML allowed only decimal numeric character references.
XML and WebSGML Adaptations SGML	Ж	XML, and SGML after the WebSGML Adaptations TC, allows both decimal and hexadecimal numeric character references. The numeric references do not contain a "u" or otherwise signal that "this is Unicode" since they are references to character numbers in the document character set and, in principle, any single-byte or multi-byte coded character set could be used in an SGML application.
	Ж	
SGML Declaration	1046	Decimal character numbers only may be used.
	Ж	Decimal numeric character references may be used in minimum literals
DSSSL	#\cyrillic-capital-letter-zhe;	Characters may be referenced by an identifier derived from the character name.
	#\U-0416	Jade also supports of the form "U-" plus the hexadecimal character number.

Unicode supports many base standards as they existed in 1993, but the field of language is not static, and new characters are continually being created. According to **[Huang and Huang 89]**, the Chinese are inventing many new characters each year, and it remains to be seen how well new characters are added to the repertoire.

In addition, Unicode does not yet support some modern scripts, although some unsupported scripts are for languages that are commonly written with other, supported scripts. While Unicode also doesn't support many archaic and obsolete scripts, several of these are proposed for future inclusion.

7. Who Uses Unicode or ISO/IEC 10646?

Both the list of software that supports Unicode and the list of applications that require Unicode support are continually growing. On the software side, this includes:

- Java
- MacOS
- Windows NT
- Windows 95 (partial)
- AIX
- Plan 9
- NetWare 4.0
- QuickDraw GX
- Jade
- nsgmls and some other SGML parsers
- XML applications

Applications that require Unicode support include:

- XML
- HTML 4.0
- Java

8. How do I specify "Ж"?

The character set may be standardized, but a character's representation in programs and data still varies with the application. Table 1 shows some of the ways that the "Ж" character is represented.

9. How do I use Unicode with SGML?

After the fundamental step of selecting SGML software that supports Unicode, you can then reference ISO/IEC 10646 as the BASESET in the CHARSET declaration in your SGML Declaration. The following CHARSET declaration is taken from the SGML Declaration for XML:

```
CHARSET
   BASESET
  "ISO Registration Number 176//CHARSET ISO/IEC
   10646-1:1993 UCS-2 with implementation level
   3//ESC 2/5 2/15 4/5"
DESCSET      0        9 UNUSED
             9        2 9
             11       2 UNUSED
             13       1 13
             14      18 UNUSED
             32      95 32
             127      1 UNUSED
             128     32 UNUSED
             160 65376 160
```

10. How do I use Unicode with XML?

All XML processors are required to support both the UTF-8 and UTF-16 encodings of Unicode characters. XML processors may also support other encodings, and the XML Recommendation lists some of them, but an XML document or parsed entity that is not in either UTF-8 or UTF-16 must begin with an XML Declaration or text declaration that specifies the encoding, for example, `<?xml encoding="EUC-JP"?>`.

To distinguish between the two encodings that do not need an encoding declaration, parsed entities in the UTF-16 encoding must begin with the Byte-Order Mark (BOM). The BOM is used in Unicode systems as a signature for detecting the sequence of bytes in each 16-bit character. When the first two bytes in the parsed entity are FE_{16} and FF_{16}, then the bytes are in the correct order for the processor, and when the first two bytes are FF_{16} and FE_{16}, the byte pairs need to be transposed before interpretation as Unicode characters. The BOM is unnecessary for UTF-8, the 8-bit encoding of Unicode characters, so its presence is used in XML systems to indicate the UTF-16 encoding.

XML parsed entities may use any legal graphic character specified in the Unicode Standard and ISO/IEC 10646. Any Unicode character that is not allowed in the current encoding may be included by a numeric reference to that character's number in the ISO/IEC 10646 repertoire. Once characters outside the Basic Multilingual Plane become part of ISO/IEC 10646, Unicode systems will be able to reference the characters using a two-character sequence called a "Surrogate Pair". The code positions for the blocks of Surrogate Pair characters are not valid XML character numbers, so references to characters outside the BMP will be made by numeric references to their ISO/IEC 10646 character number, not by pairs of numeric references to the corresponding Unicode Surrogate Pair character codes.

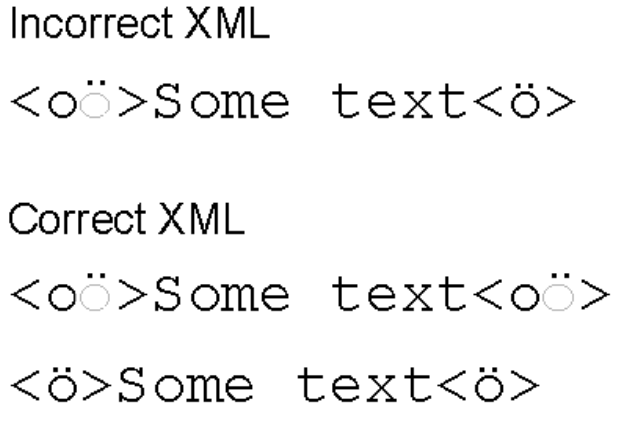

Figure 6. Canonical equivalents are not equivalent to XML

As stated previously, one of Unicode's design principles is "Equivalent Sequence" between precomposed characters and their canonical or compatibility decompositions. By default, however, XML does not consider that precomposed characters and their compatibility decompositions are equal in string comparisons. The XML recommendation does state that "at user option, processors may normalize such characters to some canonical form", but a quick survey doesn't show any XML parsers implementing this user option.

For example, according to the Unicode standard, LATIN SMALL LETTER O + COMBINING DIAERESIS (o + ¨) and LATIN SMALL LETTER O WITH DIAERISIS (ö) are canonical equivalents, but if they were used as the name in an element's start-tag and end-tag, respectively, this would cause an error because they are not equivalent according to the XML recommendation (Figure 6).

This is more likely to happen, however, if the two occurrences of the name or string were in two different entities, such as a document and its DTD, created using different editors that output different, but canonically-equivalent under Unicode, character sequences for the same text.

It remains to be seen whether the forthcoming W3C work on string identity matching obviates this difference from Unicode before it becomes a problem for XML.

11. Can I use ISO Entities with Unicode?

Annex D of ISO 8879 declared several sets of general entities, and these have been in wide use over the last twelve years. ISO TR 9573 superseded Annex D and declared more sets of general entities.

Despite this longevity and comparatively widespread use, Unicode and ISO/IEC 10646 do not include characters for all of the ISO entities. If you want to stay with the official character repertoire, XML-compatible versions prepared by Rick Jelliffe of several ISO entity sets are available at http://www.altheim.com/specs/charents.html.

The W3C Mathematical Markup Language (MathML) effort needed all of the ISO TR 9573 entities and more besides, so the MathML recommendation **[MathML 98]** defines characters for the additional entities and, for now, standardizes character numbers for these characters in the user-defined area of Unicode and ISO/IEC 10646. The recommendation states that the "STIX project of the STIPUB group of scientific and technical publishers" is working both on making available a complete set of fonts for Unicode characters for science and technology and on proposing inclusion of these characters in Unicode and ISO/IEC 10646. The STIPUB Consortium, working through the American Mathematical Society, submitted a proposal for additional math symbols and technical symbols to the ISO/IEC working group in March 1998 **[WGReg 98]**, but additional math symbols and technical symbols are not currently listed among Unicode's proposed scripts **[Prop 98]**.

Details of the characters, including lists organized by entity name and by Unicode character number, are available at **[MathML 98]**. The entities are not declared in the MathML DTD **[MathML DTD]**, but SGML-compatible versions of the entity sets are available at **[Carlisle 98]**.

Since the character numbers for these additional characters are in the user-defined area, it is best to take the advice of the MathML recommendation and refer to these characters by entity name only, as the character numbers will change if the characters are officially included in Unicode and ISO/IEC 10646.

12. What are the Unicode and ISO/IEC 10646 Coded Representations?

The Unicode Standard and ISO/IEC 10646 share a bewildering variety of coded representations for the same characters. The following sections will help in deciphering the acronyms.

UCS-4 – Four-Octet (32-bit) Canonical Form of the UCS

UCS-4 is the four-byte representation of characters in ISO/IEC 10646. It is not used in the Unicode Standard, and it is unnecessary in that Standard, since the Unicode Standard specifies a uniform, fixed-width 16-bit coded character set.

UCS-2 – Two-Octet (16-bit) BMP Form of the UCS

UCS-2 is the two-byte representation of characters defined in ISO/IEC 10646, but Unicode is compatible with UCS-2.

UTF-7 – UCS Transformation Format, 7-Bit Form

UTF-7 is defined in IETF RFC 1642, and is summarized in an Appendix to the Unicode Standard 2.0, but it is not part of ISO/IEC 10646. UTF-7 is a 7-bit form that is safe for use with email, Simple Mail Transport Protocol (SMTP), and Multimedia Internet Mail Extensions (MIME). Each 16-bit Unicode character is encoded as between 1 and $2\frac{2}{3}$ 7-bit characters.

UTF-8 – UCS Transformation Format, 8-Bit Form

UTF-8 is common to both the Unicode Standard and ISO/IEC 10646, and all XML processors are required to support this format. Originally developed by X/Open to enable use of Unicode character data in 8-bit Unix environments, this transformation format was known as File System Safe UTF (FSS-UTF) and as UTF-2 before being renamed UTF-8 and included as a normative addendum to ISO/IEC 10646.

UTF-8 encodes each Unicode character as one, two, or three bytes (and UCS-4 characters as up to six bytes). All of the ASCII code values are represented as a single byte, most non-ideographic characters are represented as two-byte sequences, and the remaining Unicode characters are represented as three-byte sequences. Once characters outside the BMP are defined, the Surrogate Pair codes that Unicode uses to represent these characters will be represented as four-byte sequences

UTF-8 is a simple and efficient conversion and a reasonably compact encoding, and it is compatible with ASCII since the ASCII characters map to the same coded representation in UTF-8. However, many Japanese, for example, do not like UTF-8 since their files become bigger when the current double-byte characters are converted to three bytes of UTF-8.

UTF-16 – UCS Transformation Format for Planes of Group 00

UTF-16 is an ISO/IEC 10646 encoding that is equivalent to the Unicode Standard with the use of

surrogates. The only difference between UCS-2 and UTF-16 is the use of surrogates with UTF-16. However, while XML specifies support for UTF-16, and UTF-16 includes the code points for surrogate pair characters, the character numbers for the code points in the surrogate blocks are not legal XML character numbers. Surrogate pairs in a UTF-16 document will be refer to a non-UCS-2 character, but numeric character references to a character number in the surrogate blocks are not legal XML.

Use of code points in the surrogate blocks, or the S (Special) Zone of the BMP as it is called in ISO/IEC 10646, enables addressing of characters in planes $1...16_{10}$ of group 0 of UCS-4; that is, it allows addressing of over one million additional UCS-4 code values in the range $00010000...0010FFFF_{16}$.

Which is best for me?

There is no single, simple answer to this question. The choice of encoding will depend in part upon your language and in part upon the tools that you are using. For example, if you are working in English, it is simplest to use UTF-8 since UTF-8 is a superset of ASCII, but if you are working in Japanese, it would be preferable to use UTF-16, since using UTF-8 would result in larger files. However, if you working with Perl, the latest version of which handles Unicode characters as UTF-8 internally, it might be simplest to only use UTF-8 for input and output.

13. Further Information

Definitive information on the Unicode Standard is available from the Unicode Consortium at http://www.unicode.org, and the Unicode Concept Dictionary, a useful glossary of Unicode terms, is available at http://cns-web.bu.edu/pub/djohnson/web_files/i18n/unicode.html.

Information on ISO/IEC 10646 is available at http://www.dkuug.dk/jtc1/sc2/wg2/, and the ISO/IEC 10646 Concept Dictionary is available at http://cns-web.bu.edu/pub/djohnson/web_files/i18n/ISO-10646.html.

Bibliography

[Carlisle 98] Carlisle, David, *MathML Files: DSSSL style sheet for MathML*, http://www.nag.co.uk/projects/OpenMath/mml-files/

[Huang and Huang 89] Huang, Jack K. T., and Timothy D. Huang. 1989. *An Introduction to Chinese, Japanese, and Korean Computing*. Singapore: World Scientific

[MathML 98] Mathematical Markup Language, Chapter 6, *Entities, Characters, and Fonts*, http://www.w3.org/TR/REC-MathML/chapter6.html

[MathML DTD] Mathematical Markup Language, Appendix A, *The MathML DTD*, http://www.w3.org/TR/REC-MathML/appendixA.html

[Prop 98] Unicode Consortium, *Proposed New Scripts*, http://www.unicode.org/pending/pending.html

[TR7 98] Unicode Consortium, Technical Report #7, *Plane 14 Characters for Language Tags*, http://www.unicode.org/unicode/reports/tr7.html

[WGReg 98] ISO/IEC JTC 1/SC 2/WG 2, N1750, *Partial Document Register (N1600 - N1810) + standing documents*, http://wwwold.dkuug.dk/JTC1/SC2/WG2/docs/documents

Biography

Tony Graham has been working with SGML for over seven years. He has worked as an Editor and a Document Analyst with Uniscope, Inc. in Tokyo, Japan for four years, and as an SGML Consultant with ATLIS Consulting Group, and he is currently a Consultant with Mulberry Technologies, Inc., a Consultancy specializing in SGML and XML training and design. Tony has designed, built, and tested DTDs and SGML applications for clients in the academic publishing, aerospace, automotive, database publishing, electronic component, photocopier, and software industries, and the languages used in these systems have been English, Japanese, Chinese, and Korean. In addition, his contributions have been incorporated into the DocBook, J2008, and Pinnacles SGML application standards.

Tony is also a qualified Electrical Engineer, and he has written programs in everything from FORTRAN on mainframes to programs in custom languages on embedded microcontrollers. Within this range is included SGML processing programs in DSSSL, XSL, Perl, Tcl, C, and Scheme.