Preservation Handbook
Plain text

Author	Martin Wynne and Stuart Yeates
Version	2
Date	17.08.05
Change History	Written by MW 2004. Revised by SY May 2005. Revised by MW August 2005.

Definition

Plain text is the lowest common denominator encoding for textual information, using an encoding standard such as ASCII or Unicode, and omitting all metadata, markup and formatting.

'Plain text' is used in contrast with 'binary' in describing the encoding of textual data. In fact, plain text is actually a type of binary encoding, since any computer file is fundamentally a sequence of bits, but plain text is usually seen as a special case and differentiated from other binary encodings, and this is the usage which is preferred here. A 'plain text' file is a special case of a binary file which contains only text represented by alphabetic, numeric and punctuation characters.

'Plain text' is also often used to refer to electronic text files containing markup which only uses standard alphabetic numeric and punctuation characters, in formats such as SGML, XML or HTML . These types of text documents with markup are not however included in this definition of 'plain text' and are dealt with in the Preservation Handbook for Marked-up Textual Data..

For the purposes of this document, the final definition we arrive at then is:

Plain text data is electronic data which contains only text represented by alphabetic, numeric and punctuation characters and does not contain markup, annotations or metadata.

Description

ASCII

Most texts described as plain text in the UK (and other predominantly Anglophone environments) use US-ASCII. While this is usually an adequate format for sharing documents between computers and applications when the text contains only simple, modern English prose, there are many cases where characters will not display properly when transferred between computers and applications. US-ASCII cannot represent a number of characters, including the pound, euro and trademark symbols, em- and en-dashes and characters such as thorn from early English prose. Accents and non-English characters are entirely lacking. Unix, Windows and Mac operating systems use different line break characters. This shows that even pieces of software that nominally use the same character sets cannot reliably interchange documents. These problems multiply when trying to cope with more than one character set.

Code pages and ISO 8859

A number of regional and country-specific variants of ASCII exist to handle accents and characters used in European languages. These are called code pages, and have been standardised as ISO 8859. These are not suitable preservation formats because there is no reliable way to determine which code page a document is encoded in, and no standard way of including it within the file itself. Code pages also restrict the ability of documents and programs to contain and process text with a mixture of languages. The preferred preservation strategy is to convert from code pages and ISO 8859 to Unicode. Such conversion can damage files if the conversion tool translates from the wrong code page, so it is recommended that, unless the text can reliably be validated after conversion, the original also be preserved.

EBCDIC

EBCDIC is an alternative single-byte encoded character set used only on IBM mainframe computers, and a contemporary of ASCII. It was never widely used outside IBM machines and was never widely internationalised. Many reliable conversion tools exist. The recommendations for preservation are the same as for ASCII.

Unicode

"Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language." (http://www.unicode.org/)

The aim of Unicode (standardised as the ISO 10646 family) is for all of the characters in all of the world's languages, including some languages of the past, to be mapped unambiguously onto a single numerical code. Unicode is certain to become the standard for character encoding in the future, and is already supported by current versions of Mac, Windows and Linux operating systems, along with Java, XML and MS Word. It already offers mappings for all widely spoken or well studied languages, coverage of less well known languages is growing fast and coverage of non-traditional languages (such as sign languages) is planned. US-ASCII and ISO-8859-1 begin with the same sequence as the default Unicode encoding (called UTF-8), which means that a US-ASCII file is also a UTF-8 file. There are other encodings of Unicode (called UTF-16 and UTF-32), but these are mainly used within applications, or where large volumes of text in languages with large character sets are being stored. Fortunately most applications which support Unicode (and all XML compliant applications) read all of these varieties of Unicode.

The AHDS does not insist on the deposit of textual resources encoded with Unicode, but expects to recommend its use under most circumstances in the near future. Resources using Unicode are preferred now. It is also preferable in the case of resources which have been encoded in a non-Unicode format for a copy converted to Unicode to be deposited as the preservation copy.

If the depositor has been working in a non-Unicode format, and has converted the text to Unicode in order to create a version for deposit, then it is important that the non-Unicode version be deposited as well. This will help to identify and fix problems which may have silently occurred in the conversion process. It may also be a useful format for a dissemination version of the resource, as it is not yet the case that all platforms and applications are Unicode-aware.

Unicode Fonts

Because the Unicode character set is very large and new languages are being added in an on-going fashion, many systems do not include a full set of Unicode fonts. Fonts covering portions of the Unicode character set are often included in language packs for various systems. Fonts for some languages may need to be installed manually by the user, especially for less widely used languages. Unlike other areas of Unicode support, this is likely to remain an issue, because the process of including minority, historic and artificial languages in Unicode is on-going.

Custom Extensions

All of the character sets above have been extended by various applications and users beyond their original use. While such extensions are not necessarily bad, their use makes the files unsuitable for preservation, as it is uncertain whether these extensions will be understood in the future. Unicode has a special method for handling such extensions, called the private use range. If the private use range is used, the meanings of each character within the range needs to be documented.

In addition to this, it should be stressed that the successful preservation of the intellectual content of a resource is about more than these purely technical questions. The text needs to be represented in an unambiguous and consistent way. Above all, clear, concise and complete documentation is necessary.

Additional Information

Creating and Documenting Electronic Texts: a guide to good practice
<http://ahds.ac.uk/creating/guides/electronic-texts/ > Last checked 17/08.05

Binary and text files (Wikipedia entry)

<http://en.wikipedia.org/wiki/Binary_and_text_files> Last checked 17/08.05

Technical Environment

Common Formats

Note there are many different encodings for the many languages of the world. It is hoped that Unicode will replace all of these encodings, but at the moment there are different de facto standards in different regions and for different languages. It is not possible to cover all of this variation in this document.

Format	Notes
ASCII (7 and 8-bit encodings)	Suitability for preservation depends on method of encoding the text and adequacy of documentation.
Other 7 and 8 bit encodings for different languages	Conversion to Unicode preferable to create preservation copy. Retention of original may be necessary unless the Unicode version can be validated reliably.
Unicode	Suitable for preservation

Additional Information

Text Encoding Initiative: languages and character sets
< http://www.tei-c.org/P4X/CH.html > Last checked 01/10/2004

Unicode home page
< http://www.unicode.org > Last checked 01/10/2004

A brief introduction to code pages and Unicode
< http://www-106.ibm.com/developerworks/library/codepages.html?dwzone=unicode > Last checked 01/10/2004

The ISO 8859 Alphabet Soup
< http://wwwbs.cs.tu-berlin.de/user/czyborra/charsets/ > Last checked 01/01/2004

A tutorial on character code issues
< http://www.cs.tut.fi/~jkorpela/chars.html > Last checked 24/02/2005

Alan Wood's Unicode resources,
< http://www.alanwood.net/unicode/ > Last checked 24/02/2005

Unicode Code Charts
< http://www.unicode.org/charts/ > Last checked 24/02/2005

Unifier Converter (Windows)
< http://www.melody-soft.com/ > Last checked 24/02/2005

Sean Redmond's Greek - Unicode converter multi-platform CGI)
< http://www.jiffycomp.com/smr/unicode/ > Last checked 24/02/2005

Ingest Checklist

Only certain formats are suitable for reliable interchange and migration and thus for preservation. For deposit of textual resources which are composed of text files with no or minimal markup, it should first be ascertained whether plain text is the appropriate format for the data. If it is important for the transmission of the intellectual content of the resource that any of the following be encoded, then it is not appropriate for them to be encoded in a plain text file:

- formatting information

- the visual appearance of the document

- the logical structure of the document

- annotations.

It may be possible and appropriate for such information to be held in a separate file, and for a plain text data to be deposited as one component of a resource.

The following guidelines should be followed for the ingest of textual data in plain text format:

Level 1 (Essential)

- a manifest of all the components of the resource and its description

- documentation to be separate from the textual resource

- explicit documentation of the character set

- explicit documentation of the markup (if any)

- explicit differentiation of the markup (if any) from the text

- explicit documentation of the use of the Unicode private use range (if any)

- verification of the character set

Level 2 (Preferred)

- a standard character set

- a standard markup formalism for text structure, annotation and markup.

Level 3 (Best Practice)

- use of Unicode

- if a non-Unicode character set is used, provision of an additional Unicode version for preservation purposes, in addition to the non-Unicode version

Inform Depositor

- Where it is necessary to convert to Unicode, the depositer should be asked to validate the migrated text.

Preservation

Significant Characteristics

The following are the significant characteristics of plain text data for preservation purposes:

- the stream of alphanumeric characters;

- the character encoding used (e.g. Unicode UTF-8; Windows Code page 1250, etc);

- documentation of the languages and scripts which occur in the data;

- a list of any non-standard characters used, or techniques used to represent characters outside of the range of the encoding employed.

It is necessary to assess whether plain text is an acceptable preservation format for the data in a particular case, and, if yes, then whether a relevant type of plain text encoding is employed. Plain text should be used in documents where access only to the text is required, and where it is not necessary to represent the structure, format of the document, and where there are no annotations. Furthermore, it is necessary to ascertain whether a relevant encoding is correctly applied and documented. If all of these conditions are met, then plain text is an acceptable preservation format.

The AHDS aims to preserve the sequence of characters and words in a form such that the character set and language can be sufficiently well identified to allow the text to be rendered in a way that is meaningful to humans and to machines.

Technique

Ensure that information is available on the type of character encoding employed in the data, and that the information is accurate.

It is preferable for a preservation copy to be made using Unicode and for the depositors to make such a copy during the construction phase of the resource-building project. If this is not done, long-term preservation of the resource cannot be guaranteed, but will be attempted on a best efforts basis.

Most of the above discussion and recommendations apply to the contents of the textual resources themselves. However, it is essential that documentation of these resources be provided, usually in a separate file. It is recommended that documentation files adhere to the same recommendations that apply to textual resources, where this is relevant and possible. This means *inter alia* that XML, XHTML, SGML and 'plain text' following the above guidelines are preferred to binary and proprietary formats. A description of the file and text type should be provided on the AHDS Deposit form. Any further information about how the documentation can be read or interpreted should also be entered on the deposit form, including information about the language in which it is written if it is not English.

Validation of Exported Data

It is necessary to verify that the files received by the AHDS conform to the stated text formatting standard. Three techniques are useful for validating different aspects of the data:

- display a text file on screen. If necessary get an expert in the language to view it, or send an image or PDF file to the depositor for verification;

- run a script to check for characters outside of the expected range for the documented language or character set;

- use a program to automatically analyse and recognize language and encoding.

Problems and Issues

Resource creators using Unicode now should beware that some applications, such as word processors, editors and email programs, may take Unicode documents and translate them into some other, probably 8-bit, character set. It may not be possible to translate back, so particular care should

be taken to process Unicode documents only with software which has native support for Unicode, or which can preserve the encoding format.

There are code pages for some languages which may be currently in use, but which are at a high risk of not being supported in the future. This is unlikely to be a problem for the major languages of Western Europe and North America which use Latin-based scripts, where compatibility with Unicode is relatively straightforward. However, for many other languages for which conversion to Unicode may be essential for preservation purposes. Data which makes use of these languages must be addressed on a case by case basis.

Another potential problem relates to file suffixes. It may be convenient for the creator, user or for an application to have a file name that is meaningful, and this is permitted. It is however essential that information about file and text types should not be available only in file suffixes.

Sometimes file suffixes are used to indicate the content of a computer file, or the program it should be associated with. People often speak of a "doc file", meaning a MS Word text file, or a "dot txt file" to refer to some sort of plain text. While this may sometimes be a useful shorthand in the spoken language, it is not a suitable way to identify file types. There are several problems.

- there are many different types of encoding encompassed by each of the suffixes;
- not all computer operating systems and applications treat file suffixes as special encodings;
- some computer systems and software will automatically and silently change or hide file suffixes;
- in some cases certain file suffixes will stop files working with certain applications;
- the meanings associated with particular suffixes are arbitrary and not subject to open or international standards, and so they mean different things to different people, operating systems and applications, and may vary in different countries and over time

Filenames should be ASCII (or from the lower range of Unicode, which is identical to ASCII), since not all current operating systems yet fully support Unicode.