Mac OS 8 Text Model Whitepaper

WorldScript, Unicode and TextObjects

John I. McConnell
International Technology Evangelist
jmcc@apple.com

Version 1.1, 7 May 1996

1 Introduction

Mac OS 8 will provide three ways to represent text: using existing Mac OS character set encodings, using Unicode, and using a higher-level mechanism called TextObjects. This whitepaper briefly describes each, makes some recommendations concerning how each should be used, and sketches likely future directions.

2 Requirements

Since the introduction of the Mac OS in 1984 and even since the introduction of WorldScript with System 7.1, important industry trends mandate enhancements to the existing Mac OS text capabilities.

- The wildfire-like spread of the Internet necessitates better interoperability. Data on the internet is in a mix of character set encodings including MacRoman, ISO Latin/1, Unicode, ShiftJIS, EUC-based encodings and dozens of others. The Mac OS must provide applications with better support for working with such data.
- Unicode has gained general developer acceptance. Because it differs significantly from existing character set encodings on the Mac, Apple must provide new facilities and tools for Unicode-based applications.
- The phenomenal growth of computer markets in Asia and other emerging markets places demands on the system to better handle the needs of their writing systems and culture. For example, systems in Japan have not usually supported sensible sorting of text.
- The emergence of speech and handwriting technologies places new requirements on the system.

Any such enhancements should be consistent with Apple's internationalization vision for the Mac OS, namely

- that any user anywhere can run any software with any mix of fonts, input methods, etc.
- that any programmer can write such software.

3 The Text Model Triad

Mac OS 8 will provide three ways to represent text: WorldScript, Unicode, and TextObjects. There are reasons for providing each model.

3.1 WorldScript

Apple uses the term WorldScript in several senses. In the narrowest sense, it is the extensions that patch Quickdraw and a few other parts of the system to add support for complex and large writing systems. In the broadest sense, it is the approach taken to internationalization on the Mac OS. In this latter sense, WorldScript also includes the Script Manager, the Text Services Manager (TSM), the specific character set encodings used by the Mac OS, and even the programming guidelines recommended in Technical Report OV20. In the rest of this paper, I will use WorldScript in this broadest sense.

WorldScript remains one of the best approaches to internationalization available on any platform. It has allowed Apple and third-party developers to release software in dozens of countries within a few weeks of the release in the US. The cost to localize such world-savvy products for any market is typically a fraction of the cost to localize a product that has not adopted WorldScript.

WorldScript does have some deficiencies:

- Unicode does not fit cleanly into WorldScript. In particular, the existing Toolbox interfaces are hard to adapt to Unicode.
- The WorldScript design has some awkward limitations concerning the maximum number of scripts and the supported character set encodings.
- WorldScript requires special expertise and creating a world-ready application is not easy.
- WorldScript provides no support for new media types, such as handwriting, nor for features important in new markets such as phonetic sorting of Japanese.
- WorldScript is intimately bound to the Mac OS which makes it difficult to write world-savvy, cross-platform applications.

With few exceptions, Mac OS 8 will support existing WorldScript applications. In particular, Mac OS 8 will provide all of the documented ScriptManager, QuickDraw, and TSM interfaces. The major enhancement for Mac OS 8 will be to re-implement the relevant managers for PowerPC and to integrate any patches into the core system. Existing WorldScript applications should thus run faster and with better stability than today.

On the other hand, the limitations of WorldScript make it unlikely that Apple will continue to enhance it beyond Mac OS 8. There is no plan to remove or even deprecate WorldScript, but Apple will invest little additional engineering effort to enhance it. As the Mac OS modern application model evolves through Gershwin and beyond, it is possible that some ScriptManager and Quickdraw interfaces will change but Apple has no plans to abandon WorldScript. In short, WorldScript will survive but developers looking at new applications should plan on using one of the other text models.

3.2 Unicode

Unicode is the emerging industry standard encoding that includes all of the characters in use in the world today. Because every character is a 16-bit integer, the programming model is simple and many developers have chosen to use Unicode as the heart of their internationalization strategy for new applications.

Mac OS 8 will be the first release of the Mac OS with significant Unicode support. Significant here means that it should be possible to write a non-trivial application which represents all text as Unicode. There are three aspects to this Unicode support on Mac OS 8. First, most parts of the system will be character set encoding neutral. Secondly, a few key parts of the system related to text processing will be Unicode-aware, that is recognize Unicode explicitly. Finally, a sophisticated character set encoding converter will make it easy to call those parts of the system that are neither encoding neutral nor Unicode-aware.

The Unicode-aware parts of the system are just display, printing, sorting and a very limited library for ScriptManager-like processing. In addition, all fonts that ship with the system will contain Unicode cmaps. Unfortunately, several parts of the system will be neither encoding-neutral nor Unicode-aware. The most problematic amongst these is TextEdit. Although a Unicode-based application can use the encoding converter to convert from Unicode to an encoding the manager understands, there will be data loss if the original text contains characters not present in the target encoding. For resources, Mac OS 8 will support application string resources in Unicode but string resources provided by Apple will remain in traditional encodings.

Beyond Mac OS 8, we can expect the rest of the system to move to become either encoding-neutral or Unicode-aware. In particular, we expect to provide more Unicode-aware text facilities, including a text engine and a richer ScriptManager-like facility. By the Gershwin release, all Apple-supplied text resources should be in Unicode and all interfaces should be either Unicode-aware or encoding neutral.

Although Unicode is a big advance over existing character set encodings, it is not a panacea for internationalization by itself. In particular it still requires too much expertise to create world-ready applications and it provides little support for new media types and emerging market needs. For this reason, the preferred representation for text on Mac OS 8 is a richer entity, called a TextObject.

3.3 TextObjects

A TextObject is an opaque data type intended to replace Str255 and ease the migration from traditional character set encodings to Unicode. The text within a TextObject may be in any character set, including Unicode. In addition, a TextObject contains a little information about the text, including the type of character set encoding, a locale (language and region) and optionally some annotation of the base text. This extra information allows the system to do a better job for certain operations such as sorting.

As the replacement for Str255, TextObjects are use extensively throughout Mac OS 8. All of the new Toolbox functions use TextObjects. They are ideal for short strings such as identifiers and menu items. Although there is no practical limit on size, they are not intended to contain documents, nor are they appropriate for text-intensive applications such as word-processors.

TextObjects provide a smooth transition strategy to Unicode. An application that uses TextObjects need not be recompiled as the Mac OS evolves towards Unicode. Those parts of the system that are encoding neutral accept or return TextObjects to the application without conversion. Those parts of the system that are Unicode-aware use the encoding converter to obtain a Unicode string. Eventually, the entire system will be either encoding-neutral or Unicode-aware and TextObject-based applications will have no need to convert between encodings.

Another major benefit of TextObjects is the annotation capability. In Mac OS 8, the only direct support for this in the system will be for yomi sorting of Japanese. Applications are free to use the annotation capability for their own purposes, but cross-application support of annotation will require application coordination, registration, or both. Annotation support for handwriting and speech will likely follow the lead set by developers.

Beyond Mac OS 8, we can expect the ScriptManager-like interfaces for TextObjects to become more complete. In addition, we expect the content of TextObjects to become increasingly Unicode. In particular, all Apple-supplied text resources will be packaged as TextObjects and contain Unicode. Finally, we may see more parts of the system support annotation. Other extensions, such as for rendering, are unlikely but beyond this it is hard to predict the future direction of TextObjects.

3.4 Summary

An application program written to run on Mac OS 8 has a choice of four ways to represent text:

- Using traditional character set encodings and WorldScript
- Using Unicode
- Using TextObjects with traditional character set encodings
- Using TextObjects with Unicode

Each way is best under some circumstances. The next section describes several different types of applications and makes some recommendations on which model they should use.

4 Application Models

There are three categories of Mac OS 8 applications that we can consider:

- Mac OS 8 Compatiable Applications
- Mac OS 8 Transitional Applications
- Mac OS 8 Only Applications

These categories are further defined in the whitepapers with the same name on this CD-ROM.

4.1 Mac OS 8 Compatible Applications

WorldScript is the only text model present on both System 7 and Mac OS 8. Most applications that run on both versions will use this model. Because Apple will also provide the character set encoding converter for System 7, some applications will also be able to make limited use of Unicode. An important subcategory here is client/server and internet applications. Such applications can use the converter as a gateway between the client application running on the Mac OS and a server running on another platform. The encoding converter eases problems of dealing with Unicode and non-Mac encodings.

An important exception involves application frameworks. Some framework providers plan to use the Mac OS encoding converter extensively within their libraries. Applications built using such frameworks could be entirely Unicode-based. Of course, there is a performance penalty for such conversion but it offers a degree of cross-platform compatibility.

4.2 Mac OS 8 Transitional Applications

Text is such a fundamental capability that it does not make much sense to provide a transitional strategy here. Instead, the transitional strategy for other Mac OS 8 services will determine our offerings in this category.

4.3 Mac OS 8 Only Applications

There are two classes of applications in this category: those that use Unicode and those that use TextObjects. Unicode is the recommendation for text-intensive applications, that is handle large amounts of text or perform character-by-character processing of text. TextObjects are the recommendation for applications that have minimal text requirements, such as utilities and perhaps even spreadsheets.