Technical Note TE535
WorldScript Q&As

CONTENTS

This Technical Note contains a collection of archived Q&As relating to a specific topic - questions sent the Developer Support Center (DSC) along with answers from the DSC engineers. Current Q&A's can be found on the Macintosh Technical Q&A's web site.

[Sep 01 1993]

WorldScript compatibility advantages and guidelines

What kind of resources are available to incorporate WorldScript capability to our programs?

___

The first thing to do to support WorldScript is to make sure that your product takes full advantage of the Script Manager. The most up-to-date and comprehensive reference for learning about it is *Inside Macintosh: Text.* Be sure to check the QuickDraw Text and Text Utilities chapters, because much of the old Script Manager functionality has been moved into those areas. Other references are Guide to Macintosh Software Localization and Localization for Japan. These are all available through APDA. (If compatibility with system software versions before System 7.1 is important to your application, see the earlier documentation: the Script Manager and Worldwide Software Overview chapters of *Inside Macintosh* Volumes V and VI, and *Macintosh Worldwide Development: Guide to System Software.)* See also the Text Services Manager chapter of *Inside Macintosh: Text.*

In general, all the rules for being Script Manager-compatible apply, but with a few new twists. WorldScript is still a very young technology and there isn't a lot of detailed information yet. Some human interface issues have not matured to the point of being ready to standardize. This means that you might find yourself breaking new ground as you add WorldScript support to your application.

The Japanese Language Kit is the first new product to take advantage of WorldScript. All the non-Roman system software uses WorldScript but doesn't really take full advantage of the possibilities.

The main advantage of WorldScript is that it supports multiple scripts in a system. The old model was that there could be a Roman script plus one other optional script, and Roman was always the secondary script, unless it was the only script. Now, any script can be the primary script, and there can be multiple secondary scripts. (Roman is still required to be available.) This makes it possible to have Japanese (and in the future, other scripts) installed as secondary scripts in any system.

Because many of the Macintosh Toolbox and operating system routines don't pass script or font information with text, it's difficult to display multiscript text properly. Menus, dialogs, window titles, and filenames are problem areas. A short-term solution is the Language Kit Extension, which was introduced in the Japanese Language Kit. It dynamically changes the system fonts according to the region code in each application's `'vers'` resource. This allows localized applications to display menus, dialogs, window titles, and filenames correctly. Multiple applications running concurrently can have independent system fonts.

The Language Kit Extension also extends the Views control panel to control the current script and font for the entire file system. This allows the user to display, create, and edit filenames for any script installed. Note that this solution supports only one script at a time, and fundamental changes will have to made to the Toolbox to support more than one script concurrently.

A technical overview of potential internationalization problem areas is given in the "Internationalization Checklist" on this issue's CD. Internationalization tips are also provided in "Writing Localizable Applications" in Issue 14 of *develop.* Here are some areas requiring special attention:

- Font names should be displayed in menus in their appropriate script and font.
- The system (primary) script may not be the same as the application script.
- Script system resources (date/time/currency) should be referenced explicitly.
- Text should be tagged with script and font information so that it can be displayed in its appropriate script and font.

Styled text and QuickDraw GX are multiscript-capable and should be used as much as possible. Unstyled text documents limit text to one script (character set), which is undesirable.

Ideally, the localized version of a product should be independent of the scripts and languages it supports. With Macintosh multiscript support and today's cultural diversity within nations, users want "foreign language-capable" applications that let them mix writing systems and use familiar regional formatting conventions.

Here are a few tests to see how well you're doing:

- Does the base version of your application work on all localized versions of system software (such as KanjiTalk, Arabic, German)?
- Install the Japanese Language Kit and see whether your application supports multiscript text. Try using the two-byte Japanese script with a relatively simple script such as German, and then with a bidirectional script such as Arabic.

Back to top

Pascal String length byte and two-byte scripts

Date Written: 11/16/92

Last reviewed: 3/1/93

How long can a Macintosh filename be on an international system? Our program currently assumes a maximum file length of 31. What does the length byte of a Pascal string signify on an international system--the actual length in bytes of the string, the logical length (the number of international characters), or neither?

___

The Macintosh file systems--the flat (MFS) and the hierarchical (HFS)--have (reasonable) limitations on the length of filenames. The limits on the lengths refer to the number of bytes required for the strings. Usually, filenames are represented as Pascal strings. The first byte of a Pascal string indicates the number of bytes occupied by the string. In the worst case in a two-byte script, only 15 two-byte characters fit into a Str31. Compared to one-byte scripts, this isn't so bad, however: two-byte characters tend to carry much more meaning than two of our familiar one-byte characters!

Back to top

Inserting one-byte characters in strings with two-byte text

Date Written: 11/16/92

Last reviewed: 3/1/93

In the two-byte script version of our application, we need to insert certain characters such as "-" and "%" into some of our strings. How can we do this, since these are obviously only one character long in C?

___

All 7-bit ASCII characters (codes less than 127) are maintained as such in all two-byte scripts. If your routines just concatenate existing (localized) strings and characters, you don't have to worry about anything. Otherwise, you'll need to call CharByte (*Inside Macintosh* Volume V, page 306, and Volume VI, pages 14-45, 14-114, and 14-124) while walking the bytes in the string.

In the Macintosh Technical Note "Fond of FONDs," the part about OutlineMetrics was updated recently to be "two-byte aware." The code fragment there should help you with strings containing text for two-byte scripts.

Back to top

Downloadables

	Acrobat version of this Note (K).	Download

Technical Notes by Date | Number | Technology | Title
Developer Documentation | Technical Q&As | Development Kits | Sample Code