New Technical Notes

Macintosh

Developer Support

WorldScript Q&As

Text

Revised by: Developer Support Center June 1993
Written by: Developer Support Center May 1993

This Technical Note contains a collection of Q&As relating to a specific topic—questions you've sent the Developer Support Center (DSC) along with answers from the DSC engineers. While DSC engineers have checked the Q&A content for accuracy, the Q&A Technical Notes don't have the editing and organization of other Technical Notes. The Q&A function is to get new technical information and updates to you quickly, saving the polish for when the information migrates into reference manuals.

Q&As are now included with Technical Notes to make access to technical updates easier for you. If you have comments or suggestions about Q&A content or distribution, please let us know by sending an AppleLink to DEVFEEDBACK. Apple Partners may send technical questions about Q&A content to DEVSUPPORT for resolution.

New Q&As this month:
WorldScript compatibility advantages and guidelines

WorldScript compatibility advantages and guidelines
Date written: 1/12/93
Last reviewed: 4/1/93

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

—

The first thing to do to support WorldScript is to become familiar with the Script Manager and make sure that your product complies. You can learn more about the Script Manager routines from *Inside Macintosh* Volumes V and VI and also *Macintosh Worldwide Development: Guide to System Software, Guide to Macintosh Localization, Localizing for Japan.* These are available from Addison-Wesley. You can find electronic copies on recent Developer CDs. *Inside Macintosh: Text* will be the most up-to-date and comprehensive reference when it is available later this year.

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. Work is in progress on a Developer University self-paced course on WorldScript, but it isn't available yet. Some human interface issues have not matured to the point of being ready to standardize. This means that applications that stretch the envelope will be breaking new ground.

The Japanese Language Module was announced at MacWorld and is scheduled to ship in the Spring of 1993. It's the first *new* product to take advantage of WorldScript. All the non-Roman system software uses WorldScript, but they don't really take advantage of any of the new possibilities.

The main new advantage is that there is support for 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, others) installed as secondary scripts in a U.S. (or other) system.

Because many of the Toolbox and operating system routines don't pass font or script information with text, this makes it difficult to display multiscript text properly. Menus, dialogs, window titles, and filenames are problem areas.

A short-term solution for this is the Language Manager, which is being 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, dialog boxes, and window titles correctly. Multiple applications running at the same time can have independent system fonts.

The Language Manager also extends the Views control panel to control the current font/script for the entire File System. This allows the user to display, create, and edit file names 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.

Here are some areas to pay special attention to:

• 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's (secondary) System fonts.
• Script System resources (date/time/currency) should be referenced explicitly.
• Text should be displayed in its appropriate script/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.

The relationship between localized application resources and localized application features is relaxing. Ideally, the language version of the product should be independent of which scripts and languages it supports.

Here are ways to check to see how well you're doing:

• Does the base (English) version work on all localized versions of system software (that is, KanjiTalk, Arabic, German)?
• Install one of the script bundles in the WorldScript folders on the Nov/Dec Developer CD and see whether you can use those script systems. Check both a WorldScript I and WorldScript II script.
• When available, install the Japanese Language Kit and see if your application supports multiscript text. Try both the U.S. and other localized Systems.

This isn't a complete list of tests, but it should be enough to get you started.

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!

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.