NOTE: This Technical Note has been underlined retired. Please see the Technical Notes page for current documentation.

Technical Note OV510
Worldwide Overview 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&As can be found on the Macintosh Technical Q&As web site.

[Oct 01 1990]

Changing DITL resource to adapt to language

Date written: 11/18/92

Last reviewed: 10/8/93

We'd like to maintain only one version of our globally distributed application, which would adapt to the language in use by changing DITL resource text items and menu titles and items. Does the Macintosh Operating System support this?

Currently the Macintosh Operating System doesn't inherently support localized resources for several languages, or choose the right language according to the localized version of the system. However, your approach of including all localized text items in the same application is absolutely feasible. Just include an option to let the user select the language - somewhere in Preferences, if not in a dedicated "Languages" menu - and design a numbering scheme for the resource IDs such that the resources to be loaded can be determined from the language code.

It's better to let the user choose the language, rather than derive it from the system. This provides for a choice in case the user lives in a multilingual region, or in case your application doesn't include translations for the language of the user's system.

Because menus, windows, and dialogs are displayed with the system font, this approach works only for languages supported by the application script. Usually, the application script is the same as the system script, so you're limited to languages supported by the system script. If the user has installed a language kit however, he/she can register your application as belonging to a different script, and the system font of that script is used to display your user interface items (see the technical note OV-20 "Internationalization Checklist", section "Working With Language Kits", for more information about language kits and application scripts). In that case, you should offer the languages of the selected script for which you have resources. Currently there is no way for an application to set its application script, nor is there a way to find out which language the user has selected for your application in the language register.

Back to top

Maintaining one application version with localized text

Date written: 9/11/92

Last reviewed: 10/8/93

We need to localize our application for several international markets. Do you have any special tools or recommendations for us?

You can use a localization tool called AppleGlot (on the *Developer CD Series* disc) to localize text in your application. Once a file has been localized the first time, the tool can compare versions and copy over everything that has stayed the same (usually 99%) so that you can focus on the text that's different. It also creates a nice audit trail and is pretty easy to use. It should save you a lot of time.

To take full advantage of this tool, you need common code for all localized versions, which is what you're planning to do to avoid the mess of having multiple sources. Occasionally, your application might have features that make sense only on a particular script system; in that case, you can check for that configuration and enable those routines when appropriate. Once you have common source and tools that help localize your application, you can add auxiliary resources for various languages.

If you have only a small amount of text in your application, it makes sense to bundle everything together in one worldwide product. Apple's TrueType fonts, for example, have internal name tables with names and information such as copyright strings in about a dozen languages. Each string is tagged with a platform, script, and language. But if you have a fair amount of textual resources, it might make more sense to have optional files and resources that can be installed as needed.

Unless you intend to support every script and language, you'll probably want to have a set of resources for unavailable languages. You can pick whatever language you want for this other set (English is popular), but the trick is to use only 7-bit ASCII characters. All script systems use the same character codes for the range $00-$7F, which match ASCII. It's the 8-bit characters that differ radically. This means that text that includes characters like ..., (TM), (c), and * will not display properly on non-Roman script systems. Just substitute text such as . . ., tm, (c), and * for them. You can decide what's appropriate and necessary.

Another thing to consider is checking for and supporting secondary script systems in your application. The Macintosh Toolbox doesn't fully support secondary scripts such as Japanese menus on an English system, but your application can support secondary script data even with the current Toolbox limitations, by using styled text commands.

Back to top

Macintosh 'vers' resource language field

Date written: 9/11/92

Last reviewed: 10/8/93

How is the language portion of the `'vers'` resource used by the system software, if at all? I'd like to use the `'vers'` resource with my multilanguage application.

The language field in the `'vers'` resource is part of the version ID to make it unique. The Language Kit Extension, part of the Japanese and Chinese language kits, is the first piece of Macintosh system software to reference the language of the `'vers'` resource. It uses this information to switch system fonts, making it possible to display localized (such as Japanese) application resources. A language registration utility can be used by the user to override the `'vers'` language.

See the Japanese or Chinese Language Kit and the technical note OV-20 "Internationalization Checklist", section "Working With Language Kits", for more information.

Back to top

Disabling multiscript keyboard menu items

Date written: 8/11/92

Last reviewed: 10/8/93

When a user launches our application, which doesn't support multiple scripts, on System 7.1 with multiple scripts installed, the user is led to believe that multiple scripts work with our application because the Process Manager enables all items in the keyboard menu. How can we manually disable items in the keyboard menu so users won't be confused?

In System 7.0 and 7.1 an application can call `KeyScript(smKeyDisableKybds)` to disable keyboards which are not in the system script or Roman (see *Inside Macintosh* Volume VI, pages 14-43 and 14-44). As of System 7.1, this keyboard disable state is part of an application's context and gets updated when applications are activated and deactivated, so calling this from one application will not affect others. This is not a complete solution, as it does not allow you to enable or disable selected keyboards, e.g., disable 2-byte scripts for password entry. At this point, there is no documented interface for disabling or enabling selected keyboards.

Back to top

Human interface for wildcards and Boolean operators

Date written: 6/11/91

Last reviewed: 10/8/93

Is there a universally recognized wildcard character for the Macintosh, like the "*" in the MS-DOS world? Furthermore, for Boolean logic, should my application accept Pascal syntax (such as .NOT., .AND., .OR.), C syntax (such as !, &&, ||), or still another convention? My users aren't programming gurus.

First, see if there's a friendlier way to implement the wildcard's function. Take a look at System 7 Finder's Find command, for example. If you find wildcard use is necessary, "*" is common, though for file searching any character other than ":" can be used in an HFS filename.

As for Boolean operators, nonprogrammers prefer a syntax that matches natural language as closely as possible, so AND, OR, and NOT are better than their C counterparts. However, user testing indicates that the most intuitive, user-friendly way to put Boolean search criteria on a command line is to bring up a dialog with pop-up menus used to form a natural-language sentence describing the search (like System 7's Find). If you can make something like this work for your application, your nontechnical users will love you.

Back to top

International documents reference list

Date Written: 3/1/91

Last reviewed: 10/8/93

Overview Books

- Guide to Macintosh Software Localization (Addison-Wesley 1992)

- Localization for Japan

- Macintosh Human Interface Guidelines (Addison-Wesley 1993)

Inside Macintosh & Co.

- Inside Macintosh: Text (Addison-Wesley 1993)

- Inside Macintosh, volume I (Addison-Wesley 1985)

- Inside Macintosh, volume V (Addison-Wesley 1987)

- Inside Macintosh, volume VI (Addison-Wesley 1991)

- Macintosh Worldwide Development: Guide to System Software, ß draft

Technical Notes

- OV 20 - Internationalization Checklist

- OV 510 - Worldwide Overview Q&As

- TE 2 - Font Family Numbers

- TE 4 - Font Names

- TE 5 -- Fonts and the Script Manager

- TE 7 - Modifying the Standard String Comparison

- TE 9 - Script Manager 2.0 Date & Time Problems

- TE 10 - Script Manager Print Action Routine

- TE 21 - Fond of FONDs

[Back to top](#)

International VBL timing on the Macintosh

Date Written: 3/9/90

Last reviewed: 6/14/93

Do VBL tasks execute at 50 Hz in Europe?

Televisions use AC line frequency to determine the vertical retrace interval. The Macintosh does not. All Macintosh systems have their own internal 60.15 Hz clock for timing VBL tasks, regardless of whether that clock is actually connected to video circuitry. Tasks installed by _VInstall are always timed by this 60.15 Hz clock. Slotted Macintosh systems may have additional frequencies determined by the installed video card. Tasks installed by _SlotVInstall are timed by these card-specific clocks. In any case, VBL task frequency is not determined by the location of your power source.

[Back to top](#)

[Back to top](#)

Downloadables

Acrobat version of this Note (56K).   [Download](#)