#227: Toolbox Karma

Written by: Ed Tecot February 1989

This Technical Note discusses Macintosh Toolbox compatibility and what you can do to help the Macintosh continue evolving in the future.

It is getting increasingly difficult to make additions to the Macintosh Toolbox. The single greatest obstacle today is compatibility. Often, engineering is prevented from doing something in an elegant manner because it would break some applications. This usually leaves three choices:

1. Break the application. Engineering does not normally choose this course of action.
2. Don't support the feature. This is rarely a good choice. It is bad for the user, and it limits developers.
3. Implement the feature in a less-than-optimal way. This is the choice most often taken. Examples are the auxiliary window list, faking desk accessories in MultiFinder to force clipboard conversion, and the ever unpopular menu bar definition procedure (MBDF).

Engineering doesn't like making additions in this way, since it clutters the architecture and makes Macintosh programming even more difficult.

Rules, Rules, Rules

You're probably thinking, "But I followed the rules." You're right. You've followed the stated guidelines in *Inside Macintosh* and the Macintosh Technical Notes. You've done nothing explicitly wrong.

However, you can do **more** than just follow the rules. Consider what effect your design decisions have on the Macintosh community. Understand that by taking advantage of a documented feature, you may be preventing the Macintosh from growing in the future. If you follow some of the following guidelines, you can give Apple some flexibility in changing rules that are no longer appropriate. These guidelines are just a sample, and hopefully you can extrapolate more from this list.

Traps Are Here to Stay

The trap interface is the ultimate Macintosh standard. Even when data structures change, the traps always work. Use them to their fullest. Don't directly manipulate data structures when a trap call will do, don't use `_HandToHand` to duplicate a handle if there is an explicit trap call available (e.g., `_TENew`), and don't patch traps. If a trap does not work the way you want, implement your own code instead of trying to patch the required functionality into the trap. If you absolutely must patch a trap, don't make assumptions about registers (e.g., `A5`) or modify the stack. See Technical Note #212, The Joy of Being 32-Bit Clean, for more information on the evils of patching traps.

Data Structures Are Subject to Change

Engineering won't haphazardly change them, but by using the traps, you give them the flexibility to make these changes. If everyone had been using `_SetWRefCon` and `_GetWRefCon`, the auxiliary window list might not have been necessary. Of course, if everyone agrees to use these traps **and** leave the auxiliary window list alone, maybe they can fix this one in the future.

Write Robust Definition Procedures

All of the definition functions, WDEF, CDEF, MDEF, etc. have room for growth. Do not stunt this growth by making unnecessary assumptions. If you do not understand the message, don't do anything. If a parameter is documented as unused, don't use it; it may be used in the future. These same rules apply to anything which might be called from ROM, such as drivers, user procedures, and filter procedures. Treat the MBDF as undocumented. It has changed considerably in the past and will continue to do so.

Use Globals With Caution

Globals often have their meaning changed or their format altered. Use the trap interfaces when available (e.g., `_TickCount` instead of `Ticks`). Try to avoid using them at all if possible.

Your Future is Apple's Future

As a developer you play a key role in shaping the future of the Macintosh. By going beyond the guidelines in *Inside Macintosh* and the Macintosh Technical Notes and considering the effects of your design decisions on the whole Macintosh community, you allow the Macintosh to grow and change while still maintaining compatibility. We won't break your applications, we can fully support features you desire, and we can implement these features in the best possible way for us, for you, and for the users. By going that extra step, you help us make programming the Macintosh simpler and ensure the best possible future for your products as well as ours.

Further Reference:

- *Inside Macintosh*, Volume V, Compatibility Guidelines
- Technical Note #2, Compatibility Guidelines
- Technical Note #117, Compatibility: Why & How
- Technical Note #212, The Joy of Being 32-Bit Clean