Application Compatibility and Chicago

This note provides guidelines for writing applications for Windows 3.0/3.1 so that they will continue to work without compatibility problems in Chicago. Generally speaking, as long as you followed the Windows SDK documentation, you will be ok.

Guidelines for Windows Application Developers:

- Don't depending on internal data structures. The window (hwnd), menu (hmenu), device context (hdc), region (hrgn), bitmap (hbitmap), and task (htask) data structures have already changed significantly for the Chicago release. Other structures may change before release.

- Don't assume objects are allocated in USER's or GDI's data segment. We are attempting to remove system resource limitations which requires that many objects not be allocated in the small 64K data segments. For example, assuming an hwnd is an offset in USER's data segment would be incorrect in Chicago.

- Don't replace system dlls like ToolHelp.DLL, Shell.DLL, CommDlgs.DLL unless you use the version apis (VER.DLL). All these DLLs are changing for Chicago and the user's system will malfunction if applications replace these DLLs with older 3.0 or 3.1 versions. Don't assume you did this correctly, most applications didn't.

- Test the Windows version number properly. Code such as

```
winVer = LOWORD(GetVersion());
If (HIGHBYTE(winVer) >= 10 && LOWBYTE(winVer) >= 3)   // BAD CODE HERE
     run;                                             // BAD CODE HERE
else                                                  // BAD CODE HERE
     exit;                                            // BAD CODE HERE
```

  Will break under Chicago which will have a version number of 4.00 since the first test of the minor version will fail. Don't laugh, many of your colleagues have already made this mistake.

  Use the following code instead:

```
winVer = LOWORD(GetVersion());
winVer = (((WORD)(LOBYTE(winVer))) << 8) | ((WORD)HIBYTE(winVer));
if (winVer >= 0x030A)          /* NOTE: Always use a HEX value here!!! */
```

- Windows 2.x apps will not be supported under future versions of Windows. Make sure your applications have been tested and built using any of the Windows 3.x SDKs so that they are marked as Windows 3.0 or above applications and can run in protected mode.

- Don't simply copy Progman grp files onto a user's disk. Use Progman's DDE functionality to add your application to Progman. Additional functionality is offered to users who use our new group file format. Using DDE to create your items will cause the creation of a Chicago style group.

- Don't assume minimized application windows have icon title windows. If you walk windows lists and "know" windows with class name "0x8004" or "#32772" are icon titles, you will break under Chicago. Write your code so that things will continue to work even if you don't find these windows.

- Don't hard-code the sizes of menus, scroll bars, sizes of captions, etc. Use GetSystemMetrics() to get these sizes. They currently change depending on display drivers and will be user adjustable in the future. Also watch the WM_WININICHANGED message and refetch these values accordingly.

- Don't hard-code button colors to be the 'standard' three shades of grey. Use the GetSystemColors api for these. Also watch the WM_WININICHANGED message and refetch these colors accordingly.

- Debugger writers should use Toolhelp.DLL if you aren't already doing so. The older windebug.dll will no longer

work under Chicago.

- Don't assume that GlobalWire() gives you DOS addressable memory. Use GlobalDOSAlloc() for that purpose.

- Don't assume that GlobalAlloc(GMEM_FIXED) gives you DOS addressable memory. Use GlobalDOSAlloc() for that purpose.

- Printer softfont info is currently stored in win.ini and is associated with a particular port (LPT1 for example). This format will be changed so that it will be associated with a printer and is independent of port.

- Don't assume the contents of WinOldAp (DOS App manager) data structures in its heap. Variables such as the window handle for the VM have changed significantly for example.

- Do not overtune your applications STACKSIZE HEAPSIZEsettings in the applications .DEF file. Several applications have been encountered who have tuned these settings, STACKSIZE in particular, to an exact situation on windows 3.00 or 3.10. These applications sometimes have problems even on windows 3.00 and 3.10 because different windows display drivers have different "stack depth" characteristics. On CHICAGO the windows core components, KERNEL USER GDI DISPLAY, have different stack depth characteristics than 3.10. In general, at least a +2k "fudge factor" is recommended for these two settings.

Guidelines for Display Driver Developers:

- For enhanced mode grabbers, the meaning of the WindHand field in the EXTPAINTSTRUC has changed. WindHand is the HWND of the grabber child window inside the WinOldAp window. All grabber painting should be restricted to this window. Grabbers weren't supposed to use WindHand for anything beyond GetClientRect(), GetDC(), and similar stuff.

- Grabbers shouldn't use EPStatusFlags bits other than fFocus, fVValid, fSelect, and fGrbProb. Some bits which are private to WinOldApp were accidently included in the DDK header files although not used in any MS distributed grabber source.

Guidelines for DOS Application Developers:

- Make sure you work in a Windows 3.1 DOS box. Especially make sure your setup program will function in a Windows DOS Box. Writing over progman group files or altering win.ini or system.ini would be bad things to do while Windows is running for example. Even though you have a DOS application, consider also writing a Window's based setup program if you need to do these sorts of things. There are many commercially available and source to one is included in the Window's SDK..

- Don't assume the location of the SFT or DOS buffers. These will be moving into high memory to provide extra conventional memory.