

Designed for Windows Mobile™ Software Application Handbook for Pocket PCs

May 2004



CONTENTS

May 2004

WHAT DOES THE LOGO MEAN?	3
SUMMARY OF UPDATES SINCE AUGUST 2003 VERSION.....	4
IMPORTANT NOTES ON APPLICATION TYPES.....	4
GENERAL REQUIREMENTS FOR WINDOWS MOBILE-BASED POCKET PC	
APPLICATIONS.....	5
INSTALLATION/UNINSTALLATION REQUIREMENTS.....	5
REGISTRY REQUIREMENTS.....	6
UI/SHELL SUPPORT REQUIREMENTS	6
FUNCTIONALITY REQUIREMENTS.....	10
DISPLAY OPTIONS	12
SPECIAL CIRCUMSTANCES AND ADDITIONAL REQUIREMENTS.....	13
SDI/FILE-BASED APPLICATIONS	13
INPUT METHODS (IMs).....	13
UTILITIES.....	14
ADD-ONS.....	14
DEVICE DRIVERS	14
HARDWARE	14
EXEMPTIONS FOR APPLICATIONS BUILT USING THE .NET COMPACT FRAMEWORK.....	15
OTHER MICROSOFT LOGO PROGRAMS	16

WHAT DOES THE LOGO MEAN?

The Designed for Windows Mobile™ logo program was developed by Microsoft to help end-users easily identify software products that are compatible with Windows Mobile - based Pocket PCs. Users of Pocket PC devices are assured that software products with the Designed for Windows Mobile logo are designed specifically for Pocket PC, and incorporate new functionality featured in Windows Mobile where applicable. For independent software vendors (ISV), licensing the logo opens up a number of marketing opportunities.

- **Microsoft Mobile2Market:** Once your applications have been logo-certified, you can submit them to Mobile2Market, a program designed to help ISVs generate new revenue opportunities by connecting them with mobile operators and retailers. Visit <http://www.microsoft.com/windowsmobile/mobile2market/>
- **Market Differentiation:** Once your applications have been logo-certified, you can use the logo on product and marketing materials to show that your product is compatible with and designed for Windows Mobile-based Pocket PCs.
- **Microsoft Certified Partner Program:** ISVs with one or more products that pass testing for this logo program are eligible to join the Microsoft Certified Partner Program at the Member level. For more information, visit <http://www.microsoft.com/partner/partnering/certified/>

Software applications are tested by NSTL (http://www.nstl.com/logoprogram/win_ce_logoprograms.html), QualityLogic (http://www.qualitylogic.com/certification_programs.html), and VeriTest (<http://www.veritest.com/certification/ms/ppc/default.asp?visitor>) For complete information about the logo testing process, including instructions on getting the necessary materials to get started as well as pricing, please visit their Web sites:

The Designed for Windows Mobile logo indicates that a software product provides all the features outlined in these guidelines. It is not a “quality assurance” seal. Neither Microsoft nor the independent test labs listed above will test the quality of your product or ensure that it is “bug-free” as part of the certification program — certification testing is solely to make sure that your product has full functionality, that the logo requirements are met, and that your product is not generating frequent faults or system crashes. The independent testing houses can provide full testing of your application separate and apart from the certification program.

Please note that the logo license agreement states: “You may only use the logo as a symbol that your product has passed the applicable Designed for Windows Mobile compatibility testing. You may not explicitly state or imply that Microsoft or the testing organization in any way endorses your product. Also, the logo program is not intended to be a “certification” program, that is, the logo does not represent that Microsoft or the testing organization certifies or warrants your product(s) in any way.”

Summary of Updates Since August 2003 Version

Added:

- Display Options Guidelines
 - Landscape / Square Screen
 - High DPI
 - GAPI
- Keyboard-based Pocket PC drivers
- Today Screen applet navigation
- Installation messages for prior Windows Mobile version applications

Important Notes on Application Types

- Applications may be file-based or non-file based. See below for additional requirements for file-based applications.
- Special cases: Development tools, as well as browser plug-ins, and add-in products such as graphics, filters, custom dictionaries, or other non-executables, which target Microsoft Windows Mobile-based Pocket PC, may earn the logo. Specific notes and requirements for some of these products are included in this document; additional standards will be published as other categories of products emerge. Products which fall into these categories will be handled on a case-by-case basis.

GENERAL REQUIREMENTS FOR WINDOWS MOBILE-BASED POCKET PC APPLICATIONS

The following guidelines for Windows Mobile-based Pocket PC applications which bear the Designed for Windows Mobile logo were fashioned with the end-user in mind. The goal behind these requirements and recommendations is to foster ease of use by providing a consistent user interface and consistent operation. Like the other Microsoft logo programs, the Designed for Windows Mobile logo is intended to inform consumers that the product bearing the logo complies with a set of criteria that ensures a convenient and predictable user experience.

Installation/Uninstallation Requirements

Applications licensed or created by OEMs for distribution exclusively in ROM are exempt from meeting the installation/uninstallation requirements, as well as the cross-platform requirements. However, if the application is distributed by means other than in ROM and requires a setup program, the application must meet all requirements as outlined here.

Required: Use Windows Mobile for Pocket PC Application Installation Mechanism

Applications bearing the Designed for Windows Mobile logo must be installed to the Windows Mobile-based Pocket PC platform using the Pocket PC Application installation mechanism. The Cabwiz SDK setup tool must be used to create Pocket PC-specific CAB files. If the application supports installation via a partnered PC, the setup process must use the CEAppMgr desktop application manager program (to register the application for Pocket PC application management). We are no longer supporting the single utility on the desktop computer, PPCLoad, but a standard installation method.

===Related Information===

CAB Files: Microsoft's compression file format often used for installation of applications.

Wceload: Resides on the Pocket PC platform. A dialog box with status bar should be displayed, indicating that Wceload is extracting components from CAB files and installing them on to the device.

Add/Remove Programs: Part of ActiveSync. Can be initiated from Tools menu within ActiveSync. Allows the end user to install and uninstall applications residing on the Pocket PC device from the desktop. Also, subsequent installation of applications can be performed without initiating setup.

*Remote API developers should refer to the following document on initiating RAPI:
<http://support.microsoft.com/default.aspx?id=831883>*

Required: Shortcuts in Programs Folder Created on Install

Developers are required to create shortcuts for their applications within the My Pocket PC\Windows\Start Menu\Programs folder on the Windows Mobile-based Pocket PC device. Alternatively, when appropriate, shortcuts can be created in the My Pocket PC\Windows\Start Menu\Programs\Games folder. This ensures that your application is accessible via Start/Programs, or Start/Programs/Games, respectively.

Setup should only create a single shortcut for each application on the device. Setup should not place the icon in the Start Menu automatically. However, it is permissible to offer the user the choice to do so.

Installation Support

Windows Mobile 2003 second edition automatically presents a message when an application compiled with an earlier operating system is installed. The message is just a warning that the previous application may not be fully functional in new screen orientation modes. This message can be avoided by updating the inf file as follows:

BuildMax field (in the [CEDEVICE] section of the .inf file used to create the cab):

- Passing a value of 0xA0000000 will disable the warning on a small screen device.
- Passing a value of 0xC0000000 will disable the warning on a device that supports rotation.
- Passing a value of 0xE0000000 will disable the warning on all devices

Registry Requirements

Required: Persistence of User Configurations in Registry

When the user uninstalls, the application should clean up any data from files or the registry, and leave as little behind as possible. The application should use the "Uninstall_Init()" and "Uninstall_Exit()" functionality to clean up data files and databases.

Required: Store DLLs Only in Windows or Application Directory

Any .DLL files used in the installation of the application on the Windows Mobile-based Pocket PC platform should only store "shared" files/DLLs in the Windows directory, and every other file in the application's own directory (which may be modified by the end user). Since the user may change the destination directory on the device during installation, applications can determine their current destination directory on the device during runtime by the registry entry:

```
HKLM\Software\Apps\<Provider> <AppName>\InstallDir
```

where <Provider> and <AppName> are the required strings specified in the Setup INF file.

Additionally, if the application installs GX.dll (GAPI), it must install that DLL in the application directory, not the Windows directory. Applications that use GAPI must install this DLL.

UI/Shell Support Requirements

Consistency of the Menu Bar and NavBar is very important to Pocket PC applications. The basic order, left to right, should be consistent with Pocket Word.

Required: Application Title Displayed on NavBar

The topmost application must continuously display its title on the NavBar, regardless of which view the application is in. Other titles (e.g. the names of the current view, document, card, dialog box) must be within the client area if it is displayed. The name displayed on the NavBar should be fully visible and may not be truncated. The only time the name may be truncated is when a notification is displayed.

Required: Applications May Not Use NavBar for Anything Application Specific

Other than the title, as required above, and the alert and notifications systems supplied through the API, the applications may not use the NavBar area for any additional functionality, such as subtitles of the current view, controls, tool bars, status icons, and so on.

Exception: The title, as required above, and the alert and notifications systems supplied through the API are exempt from this requirement.

Required: Use System Support if Implementing Fullscreen Mode

Applications may only hide the NavBar with the shell API SHFullScreen(hwnd, SHFS_HIDETASKBAR). This API places the NavBar directly behind the calling application in the Z-order, so that should the application go away for any reason, the NavBar will be immediately displayed. In general, it is not recommended to hide the NavBar because the Start Menu is the primary navigation mechanism on the device.

Required: Display New and SIP Button Controls Only in the Standard Locations

It is not required that an application always display the New and SIP buttons. However, whenever these controls are displayed, they must be displayed in their standard locations in the bottom left and bottom right, respectively. It is strongly recommended that any application that has the ability to create new data objects use a New button in the same location and support the Pocket PC global New functionality for users who have this feature turned on.

The SIP control should be displayed continuously in most applications, and only be hidden in the rare case where no input of any kind is possible (e.g. full screen mode of a game). Hiding/showing the SIP button should not be overdone to avoid confusion of when it will be available—it is far more confusing to have it flash than to have the SIP control visible in some views where no input is possible.

Recommended: Applications that draw to the screen should use GAPI. To provide a consistent user interface and maximize support across all Pocket PC devices, any application that draws directly to the screen should use GAPI.

Required: Application Windows without Full Screen Parent Window Must Be Activated When Switching to Today Screen

Since there is not a list of running tasks, application windows that do not have a full screen parent, must use some way for the user to return back to the parent screen. Some recommended methods include Start Menu, Programs shortcut, notification or Today screen tray icon, which must be activated when device is switched to the Today screen. This means that when the user returns to the Today screen, the application window will be visible to the user. If an application does not do this, there will be no way of getting back to it because there is no taskbar button to retrieve the running application.

Required: Today Screen Components Must Handle the WM_ERASEBKGND Message

Today screen components now have a transparent background. This allows the Today theme graphic to show through your component. For your component to appear transparent, it must handle the WM_ERASEBKGND message.

Required: Today Applets Navigation

Plug-ins on the today screen are now hardware navigable. Applications that leverage this capability need to set a generic registry structure that will alert shell if a plug-in “understands” the new selection mechanism as well as if it wants to be selectable. Also, focus must not be trapped indefinitely within the applet plug-in.

```
[HKLM\Software\Microsoft\Today\Items\<PlugInName>]  
"Selectability"=dword
```

Required: Order of Menus and Buttons

Any menus appearing in the Menu Bar must appear in the leftmost portion of the Menu Bar, except for when dynamically providing space for the global New button. Tool buttons, both standard and application-specific, must appear to the right of the menus in the Command Bar.

Required: Common Menu Items Must Appear in Order

If you provide any of the following menus on the Menu Bar, they must appear in this order, left to right: Edit, View, Insert, Format, Tools. In general, the File menu is not recommended—a list of documents/items in a main application view is preferred to simplify data manipulation within each application.

Required: Common Buttons Must Appear in Order

If you provide any of the following command buttons on the Menu Bar, they must appear in this order, left to right: New, Open, Save, or Print. It is strongly recommended that you follow the UI guidelines for button size and style.

Required: Common Functions Must Appear in Order

If you provide any controls on the Menu Bar (such as buttons or drop-down boxes) for manipulating Style, Font, Font Size, Bold, Italic, or Underline, they must appear in this order, left to right.

Required: ToolTips on All Command Bar Controls

If the application exposes a command bar, it is required to support ToolTips for the command bar controls. The ToolTip is displayed with a tap-and-hold action and must describe the function of the button. ToolTips must not include any shortcut sequence (i.e. no CTRL + key). For example, the Bold button's tool tip would simply read "Bold". First-time users of an application can thus easily determine the function of an unfamiliar button. See also Exemptions for Applications built using the Microsoft Windows .NET Compact Framework.

Required: 16x16 and 32x32 Pixel Icons for Application and File Types

Applications are required to register 16x16 and 32x32 pixel icons for their main executable and saved file types.

Required: Notification System Not Duplicated

Applications that display time-or-event based notifications must use the provided notification system for displaying those notifications; applications may not create their own notification system.

Required: No Duplication of Functionality Provided by Microsoft Pocket Outlook Object Model

Applications that use PIM-type data (appointments, contacts, tasks) must use the Microsoft Pocket Outlook Object Model. This is required to maximize available user storage space by preventing the storage of superfluous PIM data items. Additionally, use of the Microsoft Pocket Outlook Object Model provides a consistent user interface and simplifies communications conducted with a Pocket PC.

Required: Microsoft MAPI for the Pocket PC Functionality Not Duplicated

Likewise, applications are required to use the universal Inbox provided by MAPI (Message API) where appropriate. For example, some communication applications (such as two-way paging applications) may require greater functionality than MAPI provides, and are exempt from this requirement. All other applications, however, are expected to use the Universal Inbox provided with Pocket PC.

Required: Support for Standard Wait Cursor

Applications must display the system wait cursor (color wheel) when asked to execute a command that renders the current window, or the system as a whole, unresponsive to user input for more than 0.5 seconds. When the system wait cursor is displayed only in the current window, the user may continue to interact with other windows, including the NavBar.

Required: Applications Must Handle the Input Panel Appearing/Disappearing

From the UI perspective, the application must be designed to account for the fact that an 80-pixel tall Soft Input Panel (SIP) may be docked in the area immediately above the command bar at any time, requiring the application to be resized out of the way. The specific requirements are:

- All text entry fields must be accessible with the SIP up. This can be accomplished either by placing all edit fields high enough on the screen so that they will not be obscured by an 80 pixel tall SIP, or by having a scroll bar when the SIP is displayed (or both).
- Applications must resize in response to docked input methods. This includes resizing and/or shifting things like dialog tabs, status bars, and other elements as needed, and/or resizing or drawing scroll bars to ensure all UI is accessible while an 80-pixel tall SIP is up. Note that a docked input method may be as tall as 140 pixels, but applications are not expected to be optimized for a SIP height greater than 80 pixels.

Recommended: Avoid controls on bottom 80 pixels on Pocket PCs.

In some cases where landscape may be used, it is recommended that applications avoid putting controls on the bottom 80 pixels on Pocket PC devices. Dynamic scrollbars in dialogs and prop sheets will be added if the controls fall below the 240x240 screen area. Scroll bars will not be added to windows.

Required: Help Systems Integrated with System TOC

Applications that provide help files on the device are required to register their help system with the Pocket PC's Help System Table of Contents, so that users have one entry point for help on all installed applications. This requires that all help content is authored in HTML format.

Note: Applications with small amounts of Help information presented in one or two dialog boxes are not subject to this requirement. The intent of this requirement is not that every word of help is listed in the main help TOC, but that the main help system provides easy access to any substantial help information available on the device. See also Exemptions for Applications built using the Microsoft® .NET Compact Framework.

Required: No Help Tool Button or Help Item on Menu Bar

Because general application and context-sensitive help is accessed via the Start Menu on a Pocket PC (for the current window), applications must not provide another access point, in order to ensure consistency across the platform. See also Exemptions for Applications built using the .NET Compact Framework.

Required: Applications Must Respect User Settings

The application must keep user's settings including regional settings, theme colors, and so on.

Functionality Requirements

Required: Full Functionality and Stability

Windows Mobile applications must be fully functional and stable on the designated test platforms. All functionality must be in place when the application is submitted for testing. While logo testing is not QA testing, the goal is to confirm that the application being tested does not appear to adversely affect the overall stability or performance of the device environment. The application must recover from standby/suspend situations on leading equipment. The application must also be well behaved with Shell and system features, such as not compromising hardware button functionality or overriding other Shell or system features, unless it is required for the proper functioning of the application.

Test platforms for the test will be determined by the operating system requirements specified for the application:

Required: Help Must Be Functional

Any online help included with Windows Mobile applications must be fully functional at the time of testing. At minimum, the help system for the Pocket PC must be activated via Start/Help from anywhere within the application. **Note:** It is not required that help content be final at the time of logo testing.

Required: Must Not Assume External Storage

Although many Pocket PCs have them, external storage cards are not required on Pocket PCs. Applications must not crash, hang, or exhibit adverse behavior on devices without external storage cards.

Required: Graceful Application Shutdown

Because applications will be shut down by the Shell without the user taking any explicit action, applications must shut down gracefully. This means not displaying any dialog boxes or other UI, not consuming excessive CPU time, and not consuming significant additional memory while closing.

Required: Restoration of State When Starting

Because the Pocket PC user will not have any concept of which applications are and are not running, applications must restore critical state when launched. Note that this requirement does not mean that complete state must be restored, although this would be ideal. The goal is for the user to feel comfortable returning to the application, whether or not it was shut down since last used. For example, applications should restore the user to the previously selected view and scroll state, if applicable.

Required: Non-full Screen Dialog Boxes Displayed Out of the Way of the Input Panel

Any non-full screen dialog box should be vertically centered in the area above the standard, 80-pixel high Input Panel when it is displayed. It must be placed in this space even if the Input Panel is not currently displayed, since if the user displays the Input Panel, no part of the dialog box will be hidden by the Input Panel. If the dialog box is taller than the space available when the Input Panel is displayed, the dialog box must be butted against the top of the screen. Message box text should be informative but brief to avoid part of the message box potentially being hidden behind the SIP.

Required: No Underlined Accelerators in Menus or Dialogs Boxes

Because there is no physical keyboard on the Pocket PC, accelerators do not make sense. It is required that there be no underlined accelerators in Windows Mobile-based Pocket PC applications.

Required: No Shortcut Sequences Listed in Drop-down Menus, ToolTips, or PopUp Help

Because there is no physical keyboard on the Pocket PC, shortcuts such as Ctrl+N for New must not appear on toolbars, ToolTips, PopUp help, or other controls such as menus.

Note: Support for the Ctrl+Q Power User shortcut, which exits the current application, is recommended, but this shortcut item may not be visible on screen.

Required: Applications Must Use the Connection Manager to Configure All Connectivity Options

Connection Manager provides users with a single user interface from which they can configure all their connectivity options—wired and wireless network cards, modems, cellular, VPN, and so on. It also exposes a simple API, allowing the developer to essentially ask Connection Manager to provide an Internet or “work network” connection, and leaves the system to sort out the details.

Required: Long Filename Support

The application must:

- Support long filenames as described below.
- Use long filenames for displaying all document and data filenames in the Shell, in dialog boxes and controls, and with icons.
- (strongly recommended) Hide the .XXX extensions in the application itself.
- Use the common dialog boxes to save and load files (recommended).

Pocket PC applications that save files that are exposed to the user must support long filenames (LFNs) with all the following required features:

- Users must be able to enter filenames of 128 characters, including all uppercase and lowercase standard characters, embedded spaces, embedded periods, and so on.
- Leading and trailing spaces must be stripped by the Save As command.
- It is not necessary to allow leading and trailing periods. These may be stripped by your application if you wish.
- Question marks anywhere in the filename must prevent the file from being saved. No error message needs to be displayed.
- The plus-sign (+), comma (,), semicolon (;), equals-sign (=), left-square bracket ([), and right-square bracket (]) must be supported anywhere in the filename, including leading and trailing. Embedded periods must also be supported. These should not cause any error conditions.

Display Options

Landscape / Square

Dynamic scrollbars in dialogs and prop sheets will be added if controls fall below the 240x240 screen area. Scroll bars will not be added to windows.

GAPI Coordinate Rotation: GWES rotates screen tap coordinates when in landscape mode. Though GAPI can draw to the screen as normal, it cannot understand these rotated coordinates. The solution is for GAPI to force the device back to portrait whenever it gains focus, and to restore the original state upon termination.

High DPI Implementations

The shell will stretch application icons: If an application does not provide a correctly-sized icon the Shell will automatically stretch the provided application to the proper size. This could result in aesthetic problems. Developers are encouraged to redesign high dpi icons for their applications

EDG (EAPG) and MDD will think pixel-specific APIs and messages: Under this model, legacy application calls to pixel specific APIs will be automatically doubled. New applications can either change their CEOS version stamp to v4.3 or attach a manifest stating that the application is high DPI-aware, to disable thinking. The two groups will automatically scale pixel coordinates in function calls made by legacy applications. High DPI-aware applications will attach a manifest to disable this thinking, gaining access to

true coordinates. EDG APIs are: CreateWindowEx, SetPixel, SetWindowPos, GetWindowRect, GetClientRect, GetStockObject, BeginPaint, EndPaint and DeleteDC if necessary, CreateFontIndirect, DrawText, and ExTextOut. MDD APIs will be identified by the CSG

OEMs may implement a display driver ExtEscape: To enable high-DPI access to the screen, MDD will define the GetFrameBuffer ExtEscape for OEMs to implement in their display driver. This ExtEscape will provide game developers access to the full, QVGA framebuffer, enabling them to design applications that take advantage of the high-DPI screen and that do not suffer the performance degradation of GXDMA

SPECIAL CIRCUMSTANCES AND ADDITIONAL REQUIREMENTS

SDI/File-Based Applications

These are applications with the primary purpose of opening, creating, and editing documents. Word processors, spreadsheets, and so on, are all considered file-based applications. (Examples of non-file-based applications include PIMs and games.) File-based applications are subject to the following additional requirements.

Required: Support for Common Dialog Boxes

File-based and/or Single Document Interface applications must support common dialog boxes on the Windows Mobile-based Pocket PC, such as dialog boxes for naming and saving files, and dialog boxes for not using proprietary interfaces for these functions.

Required: Support Only One Instance of Each Application or Control Panel

Because there is no taskbar and memory is managed by the Shell, users switch between applications that are still running using the Start Menu or Programs folder. As a result, only one instance of each application or applet must be allowed to run. Any application that supports multiple open documents or data types must support that functionality from within the application, not through multiple instances.

Input Methods (IMs)

Any application or component whose primary purpose is to input text into the device is defined as an Input Method.

Note that an IM must be docked. Docked IMs act like a docking toolbar, taking ownership of the area immediately above the command bar. When an IM is implemented as docked, applications will resize upwards and out of the way of the IM.

Required: 80-pixel Docked IMs

It is required that Input Methods (IMs) be designed at 80-pixel high max and docked, because all applications bearing the logo will have been designed with this assumption in mind.

Required: Input Methods Use Input Panel Architecture

All Input Methods must plug into the input panel architecture. This includes using the SIP button on the command bar to activate/hide the Input Method, using the Input Method arrow button on the taskbar to select the Input Method, and having options (if any) in the

Input control panel applet. In addition, the Input Method must unload when not the currently selected Input Method.

Note: Since an Input Method is not an application, IMs are not subject to all application requirements. For example, unlike an application, an IM is allowed to have its own help button, since the system level method of Start>Help will not give context help for an IM. Other requirements will be reviewed on a case-by-case basis considering the details of the IM in question.

Required: Keyboard-based Pocket PC devices

Applications that install and uninstall a keyboard driver are required to set/unset the proper registry key located under
HKEY_CURRENT_USER \SOFTWARE\MICROSOFT\SHELL\HasKeyboard={0=false;1=true}

Utilities

Utility products are non-file-based applications or application components designed to optimize the Pocket PC environment. Some requirements may not apply to certain utilities, and will be considered on a case-by-case basis.

===Related Information===

An example of a utility is bBackup by BSQUARE. It backs up user's data.

Add-ons

Add-on products are generally non-executables such as data libraries, clip art collections, templates, and macros. Add-ons must operate with an application that bears the logo and must fulfill all relevant logo requirements for installation, and so on.

Device Drivers

Device Drivers must fulfill all relevant requirements for UI and Windows CE .NET Test Kit (CETK) where applicable.

Hardware

Compatible hardware peripherals may qualify for a *Designed for Windows Mobile* logo and participate in the Mobile2Market program if they work as advertised for specific Pocket PC models and can be classified within the following groups:

- Hardware accessory
 - CF/SD Memory, stylus, case, etc.
- Hardware – proprietary connection without software
 - Head set, Car Kit, charger, cable, etc.
- Hardware with device driver
 - GPS, keyboard, network, etc.
- Hardware with driver and UI
 - Keyboard, network card, GPS, barcode scanner, camera, etc.

The certification process requires the hardware and device driver, if needed, be submitted to NSTL for testing and Microsoft Windows CE.Net Test Kit (CETK) review. Specific details may be obtained from NSTL

(http://www.nstl.com/logoprogram/win_ce_logoprograms.html).

Exemptions for Applications Built Using the .NET Compact Framework

The following requirements are currently exempted for applications built using the .NET Compact Framework.

Exempt: No Help Tool Button or Help Item on Menu Bar

Because general application and context-sensitive help is accessed via the Start Menu on a Pocket PC (for the current window), and in order to ensure consistency across the platform, applications must not provide another access point.

Exempt: Help Systems Integrated with System TOC

Applications that provide help files on the device are required to register their help system with the Pocket PC's Help System Table of Contents, so that users have one entry point for help on all installed applications. This requires that all help content is authored in HTML format.

Note: Applications with small amounts of Help information presented in one or two dialog boxes are not subject to this requirement. The intent of this requirement is not that every word of help is listed in the main help TOC, but that the main help system provides easy access to any substantial help information available on the device.

Exempt: ToolTips on All Command Bar Controls

If the application exposes a command bar, it is required to support ToolTips for the command bar controls. The ToolTip is displayed with a tap-and-hold action and must describe the function of the button. ToolTips must not include any shortcut sequence (i.e. no CTRL + key). For example, the Bold button's ToolTip would simply read "Bold". First-time users of an application can thus easily determine the function of an unfamiliar button.

OTHER MICROSOFT LOGO PROGRAMS

Microsoft offers a number of logo programs, which all share a common goal: providing the end user with an easy way to recognize products that were designed to work well with Microsoft's industry-standard operating systems and applications. Customers of products that bear the logo can more easily find products compatible with their existing systems and take advantage of the functional skills and knowledge they already possess. It is possible for applications to bear more than one logo, and developers are encouraged to take advantage of the full array of marketing opportunities presented by these programs.

Microsoft Certification Programs:

- Certified for Windows 2000 (<http://msdn.microsoft.com/certification/>)
- Designed for Microsoft Windows XP (<http://www.microsoft.com/winlogo/default.asp>)
- Microsoft Commerce Server 2000 Integration Testing
(<http://www.microsoft.com/commerceserver/partners/default.asp>)
- Windows 2000 Independently Verified Compatible
(<http://www.microsoft.com/Partner/Partnering/certified/default.asp>)
- Featuring Microsoft Visual Basic Technology
(<http://msdn.microsoft.com/vba/license/process.asp>)
- Microsoft Windows Terminal Services Testing
(<http://www.microsoft.com/ntserver/ProductInfo/terminal/default.asp>)