

Designed for Windows Mobile™ 6 Standard Software Application Handbook

May 2007



Contents

May 2007

WHAT DOES THE LOGO MEAN?.....	3
SUMMARY OF UPDATES SINCE OCTOBER 2006 VERSION	4
IMPORTANT NOTES ON APPLICATION TYPES	4
GENERAL REQUIREMENTS FOR WINDOWS MOBILE 6 STANDARD-BASED APPLICATIONS	5
INSTALLATION/UN-INSTALLATION REQUIREMENTS	5
UI/SHELL SUPPORT REQUIREMENTS	8
FUNCTIONALITY REQUIREMENTS	10
DISPLAY OPTIONS	12
SPECIAL CIRCUMSTANCES AND ADDITIONAL REQUIREMENTS	14
SDI/FILE-BASED APPLICATIONS.....	14
UTILITIES	13
ADD-ONS	153
DEVICE DRIVERS.....	15
HARDWARE	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 6 Standard-based devices. Device users are assured that software products with the Designed for Windows Mobile Version 6 logo are designed specifically for their device, and incorporate new functionality featured in Windows Mobile 6 Standard-based devices where applicable. For Independent Software Vendors (ISVs), licensing the logo opens up a number of marketing opportunities.

- **Microsoft Mobile2Market:** Once your Windows Mobile 6 Standard-based applications have been logo-certified, you can submit them to Mobile2Market, a program designed to help ISVs create new revenue opportunities for their applications by connecting them with mobile operators and retailers. Visit msdn.microsoft.com/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 and designed for Windows Mobile 6 Standard-based devices.
- **Microsoft Certified Partner Program:** ISVs with one or more products that pass certification testing for this logo program are eligible to join the Microsoft Certified Partner Program at the Member level. For more information, visit 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/sphone/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 (NSTL, QualityLogic, and VeriTest) 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 application performs all claimed functionality, that the logo requirements are met, and that your product is not generating frequent faults or system crashes.

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 6 Standard 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 October 2006 Version

Added:

- Recommended: Running Application Verifier
- Ability for application to run on multiple devices
- Antivirus screening
- Hard coding colors of dialog boxes
- Ink Support
- Application must pass Hopper test

Important Notes on Application Types

- Applications may be file-based or non-file based. See below for additional requirements for file-based applications.
- Guidelines and requirements listed are for Windows Mobile 6 Standard-based devices.
- Special cases: Development tools, as well as browser plug-ins and add-in products such as graphics, filters, custom dictionaries, or other non-executables that target Windows Mobile 6 Standard-based devices, 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 that fall into these categories will be handled on a case-by-case basis.
- Hardware, which is bundled with your software, must be tested by the Windows Hardware Quality Lab. E-mail <mailto:wcelogo@microsoft.com> or visit <http://www.microsoft.com/hwdq/hwtest/>.

GENERAL REQUIREMENTS FOR WINDOWS MOBILE 6 STANDARD APPLICATIONS

The following guidelines for Windows Mobile Standard-based applications, which bear the Designed for Windows Mobile Version 6 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 Version 6 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/Un-installation 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 6 Standard Installation Mechanism

Applications must be installed to the Windows Mobile 6 Standard platform using the Application installation mechanism. The Cabwiz SDK setup tool must be used to create CAB files.

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 \Windows\Start Menu or \Windows\Start Menu\Games folder on the device. To ensure the shortcut is in the correct location regardless of device configuration, use the appropriate CE String in the CAB file.

Setup should only create a single shortcut for each application on the device.

Required: No warnings during installation

There should be no warning message displayed during installation that suggests that the application was designed for an earlier version of the operating system.

Windows Mobile 6 automatically presents a message when an application compiled for an earlier operating system is installed. The message is 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 0xE0000000 will disable the warning on all devices

Required: Packages Must Have Setup XML (WAP Provisioning)

The Setup XML is generated automatically by CABWizSP. If you wish to create your CAB manually, you will also have to create this Setup XML manually.

Required: Provisioning XML File Must Include Install CSP XML with NoUninstall Parm

The provisioning XML file should include an Install CSP section at the start of the document, just below the <wap-provisioningdoc> element. This Install CSP XML should also include the NoUninstall parm.

Required: CEsSetup DLL Must Expose Correct Four Interfaces

Windows Mobile 6 Standard-based devices support the optional use of CEsSetup DLLs. These .dll files are called at various points during an installation or subsequent uninstallation. A CEsSetup DLL, if included, must have four entry points: Install_Init, Install_Exit, Uninstall_Init, and Uninstall_Exit.

Required: Install CSP Section of Setup XML Must Contain Valid Values for Each of the Parameters

The following requirements must be met for the Install CSP section of the Setup XML file.

Note: Because CabWizSP will automatically generate this XML correctly, this is only a requirement for CABs that are generated by using any other method.

```
<characteristic type="Install">
  <parm name="InstallPhase" value="install" />
  <parm name="AppName" value="Microsoft Blackjack" />
  <parm name="NumDirs" value="1" />
  <parm name="NumFiles" value="1" />
  <parm name="NumRegKeys" value="0" />
  <parm name="NumRegVals" value="0" />
  <parm name="NumShortcuts" value="0" />
</characteristic>
```

Required: Application Name Must Be Less Than 70 Characters

The application name parameter "AppName" in the Install CSP XML must be less than 70 characters and include the company name followed by the application identifier; for example:

Microsoft Blackjack 1.0.

Required: Application Must Install/Uninstall Correctly

Applications must meet the following criteria to correctly install and uninstall:

- The installation or uninstallation operation must not crash, lock, or otherwise disable any of the functionality of the device.
- All expected confirmation and information dialogs should appear to the user, including installation confirmation, installation progress, removal confirmation, and any dialogs specified by the CEsSetup DLL.
- The installation and uninstallation logs must report zero errors.

Required: Applications Must Not Add Files to RAM File System During Installation

During installation, applications must not put any files in RAM file system. Files must be installed in persistent storage (local or removable, i.e., \StorageCard). To determine acceptable storage, applications can use the *GetStoreInfo* API.

Required: CEsSetup DLL Must Install to Local Storage Volume

If specified in the installation instructions, the CEsSetup DLL must correctly install (and run) regardless of its location on any local storage volume (e.g., IPSM, or MMC) if the InstallDir variable is specified and sufficient space exists.

Install the application from the MMC or IPSM to ensure the application can be installed from any location. This step is necessary to verify the application is not dependent on a specific installation path.

Required: Applications Cannot Be CPF Files

A .cpf file is a provisioning file that is wrapped in a .cab file and can be signed optionally with a certificate. Applications cannot be .cpf files because .cpf files are purely configuration data and do not include any applications.

Required: No CEsSetup DLL for HME or TSK Files

Home screen files (.hme or .tsk) should contain only home screen related files such as graphics, color schemes, and home screen plugins. CEsSetup DLLs are not allowed for .hme or .tsk files.

Required: Input Information

When submitting an application, a .cab file is required. Optionally, a Setup package that works on the desktop through Microsoft ActiveSync® may be submitted, in addition to the required cab.

Required: Clean Up Data from Files and Remove Registry Keys When Uninstalling

When the user uninstalls, the application should clean up any data from files (except for shared data files), and leave as little behind as possible. The application should use the "Uninstall_Init()" and "Uninstall_Exit()" functionality of a CEsSetupDLL to clean up data files and databases.

Additionally, any registry keys placed on the device during install or at a later time during execution must be removed (except for shared registry keys).

Required: Store DLLs Only in Windows or Application Directory

Any DLL files used in the installation of the application on the Windows Mobile 6 Standard platform should store only "shared" files or DLLs in the Windows directory, and store every other file in the application's own directory (which may be modified by the end-user).

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.

Recommendation: Do not write in keys other than HKCU

Applications writing to the registry must only use a subkey under HKEY_CURRENT_USER. The other keys (HKLM and HKCR) are meant for system services and drivers.

UI/Shell Support Requirements

Consistency of the Menu Bar and NavBar is very important to applications.

Required: Clean Up When Closing the Application

Applications must be written so that they recognize the request to close the application and perform cleanup. The specific messages that the application will receive include:

- WM_CLOSE (sent when the application is being asked by the device to shut down completely).

Recommended: WM_HIBERNATE (sent when memory is running low and the application should clean up any memory that it doesn't absolutely require).

For more information, see the Windows Mobile 6 Standard SDK.

Required: Menu on Right Soft Key

If the application uses a menu, the menu must be located on the right soft key. The left soft key must be for quick (common) actions, such as "New." If a second menu is used, it can be put on the left.

Required: Dialog Box Controls Must Be Stacked Vertically

If an application includes a dialog box, such as a list of options, all of the controls must be stacked vertically.

Required: No Ellipses for Menu Items

An ellipsis is sometimes used on desktop and other mobile device menus. Windows Mobile 6 Standard menu items cannot include an ellipsis (...).

Required: Use "Done" to Close a Screen

When a screen needs to be closed (for example, when an Option dialog box needs to be closed or when finished adding a contact), "Done" must appear on the left soft key.

Required: Use "Cancel" in Edit View

On a screen where edits can be made, "Cancel" must appear on the right soft key, or in the menu on the right soft key. The cancel functionality must return the device to the pre-edit stage, without making changes to the data on the current screen.

Required: Back Button Performs Backspace in an Edit Control

On a screen where text can be edited (for example, in a mail message), the back button must enable the user to backspace on the entire screen, but not exit.

Required: Spinner Controls

If an application requires radio-button or drop-down list behavior, spinner controls (left and right arrows) must be used.

Required: User Must Initiate Call by Choice or Acknowledgement

If an application initiates a voice call, the user must actively choose to make the call or acknowledge a prompt (if the application initiates the call in the background).

Required: User Must Initiate Data Download by Choice or Acknowledgement

If an application downloads data from an over-the-air connection, the user must choose to allow the download, or acknowledge a prompt (if the application initiates the call in the background). The choice may come in the form of a setting confirmed by the user at an earlier time.

Required: No Information Added to Title Bar

Applications cannot add additional icons to the title bar. It is permissible to replace the application title in the title bar with context-specific information. For example, the agenda view in calendar shows the date in the title bar instead of "Calendar".

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.

Recommended: If high-DPI mode is supported, also register 22x22 and 44x44 pixel icons.

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.

Required: Microsoft MAPI 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.

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. The application may present a progress bar instead of the Standard Wait Cursor, if it is loading, storing or rendering data.

Required: Applications Must Respect User Settings

The application must keep the user's settings, including regional settings, theme colors, and so on. For example, applications must use `GetSysColor()` to retrieve colors for the UI wherever possible.

Required: Hard coding colors of dialog boxes

The application shall use default system colors for all dialog boxes. Colors should not be hard coded into the application. For example, the application should use `GetSysColor()` or the `SystemColors` class. The application should display properly in any theme.

Required: Ink Support

If an application supports the Ink API, all of the Ink functionality should be tested if applicable. This includes:

- Drawing
- Lasso
- Highlighting
- Ink formatting
- Eraser
- Cut, Copy and Paste
- Drag and Drop
- Moving or Sizing
- Multilevel Undo/Redo
- Text Recognition

The application should also be tested on a device that does not support the Ink API. The application should function normally with the exception of the lack of Ink support. The application should not crash or have any other errors.

Recommended: Refrain from using GAPI

Developers writing graphics applications should use `DirectDraw` and `Direct3D Mobile` technologies instead of GAPI (the Game API). GAPI was deprecated in Windows Mobile 5.0 and will not be supported in future releases of Windows Mobile. GAPI based games may not be able to pass future "Designed for Windows Mobile" logo certification tests.

Functionality Requirements

Required: Full Functionality and Stability

Applications for Windows Mobile 6 Standard-based devices 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: Application must pass the Hopper test

The application shall pass 1 hour of Microsoft's Hopper test.

The Hopper utility is included in the Windows Mobile Professional and Standard SDKs available for download at:

<http://www.microsoft.com/downloads/details.aspx?FamilyId=06111A3A-A651-4745-88EF-3D48091A390B&displaylang=en>

Steps to run Hopper on an application:

Using rapistart from the Windows Mobile Power Toys you can run hopper on a particular application by doing:

```
rapistart.exe hopper /aGames\Solitaire.lnk
```

In this case, hopper will click the Start Menu, then Games and then click on Solitaire. The command line version does not support spaces.

The Windows Mobile Power Toys are available for download from:

<http://www.microsoft.com/downloads/details.aspx?FamilyID=74473fd6-1dcc-47aa-ab28-6a2b006edfe9&DisplayLang=en>

Once installed, a shortcut to the directory of files is placed in the Start Menu under All Programs. Its easiest to just start a command prompt and do:

```
cd "\Program Files\Windows Mobile Developer Power Toys\RAPI_Start"
```

and run the rapistart.exe command from there. Place the hopper.exe in the root of the file system on the device.

Recommended: Running Application Verifier

Microsoft's AppVerifier test will be used to evaluate the application's memory leak performance. This test comes bundled as part of the Microsoft CETK tests and verifies that there are no memory leaks in the application and can also detect some forms of heap corruption.

Exceptions:

- AppVerifier will not run on managed code (i.e. .NET) applications.
- Any errors associated with the Load/Free Library.

- Any leaks associated with the menu bar.
- Errors associated with pimstore.dll and mscoree2.dll.
- Error linked to DefWindowProc, the API used to invoke default window processing for any window messages not explicitly handled by the application.

Required: Ability for application to run on multiple devices

The application should have the ability to run on different types of Windows Mobile 6 Standard-based devices. There should be no evidence of black (or any other color mask) lines on the top, bottom or sides of the screen. The application needs to display and perform properly on landscape, portrait, and dynamically switching display devices.

Required: Antivirus screening

The application shall be free of viruses and malware.

Note: The ISV is not required to perform this testing but should be aware that the ITL (Independent Test Lab) will be screening the application for viruses and malware.

Required: Must Not Assume External Storage

Although many devices have them, external storage cards are not required on a Windows Mobile 6 Standard-based device. 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 while receiving a call or interruption

When receiving an incoming call or alert, the application must maintain functionality. Applications must not interfere with the user accepting a phone call or alert by incorrectly re-painting the screen.

Required: Restoration of State When Starting

Because the 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: No Underlined Accelerators in Menus or Dialogs Boxes without alternative selection methods

Devices may or may not have either a keyboard or a number pad for input. Therefore, while underlined or numeric accelerators are allowed, no functionality can depend solely on the use of accelerators. There must always be a way to access the functionality through a list selection or smartkey input.

Recommended: Avoid manually assigning mnemonics to menu items or dialog boxes.

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

Connection Manager provides the user 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. If an application must make a change to user settings or connection settings; the user must be notified.

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 dialogs and controls, and with icons.
- It is strongly recommended that the .XXX extensions are hidden in the application itself.

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.

Required: Shut Timers Off When Application Is Running in the Background

To optimize battery use, if an application uses a timer for visual elements, the timers must shut off when the application is no longer running in the foreground.

Required: Applications Must Not Interfere with Incoming Call Functionality

An application cannot interfere with the normal call user interface, and it cannot delay the user's ability to answer an incoming call. Applications that run in full screen mode or always

on top must move to the background to allow the incoming call user interface to appear on top.

Display Options

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

MFC Applications

Applications that use MFC classes must link statically to the MFC runtime. Distribution of dynamically linked MFC DLLs is not allowed.

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 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.

Utilities

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

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 device 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).

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>)

Mobile2Market Codesigning

The Mobile2Market codesigning program is designed to help ISVs deploy their applications to the greatest number of Windows Mobile devices with a single signature. For more information on Mobile2Market codesigning, visit: <http://msdn.microsoft.com/mobile2market>