

Windows Embedded POSReady 2009

Guidelines for products designed for the Windows Embedded POSReady operating system

Application Test Specification

A testing methodology to ensure that applications function smoothly on the Windows Embedded POSReady 2009 operating system



Disclaimer

Microsoft does not make any representation or warranty regarding specifications in this document or any product or item developed based on these specifications. Microsoft disclaims all express and implied warranties, including but not limited to the implied warranties or merchantability, fitness for a particular purpose and freedom from infringement. Without limiting the generality of the foregoing, Microsoft does not make any warranty of any kind that any item developed based on these specifications, or any portion of a specification, will not infringe any copyright, patent, trade secret or other intellectual property right of any person or entity in any country. It is your responsibility to seek licenses for such intellectual property rights where appropriate. Microsoft shall not be liable for any damages arising out of or in connection with the use of these specifications, including liability for lost profit, business interruption, or any other damages whatsoever. Some states do not allow the exclusion or limitation of liability or consequential or incidental damages; the above limitation may not apply to you. The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented. This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.

Microsoft, Windows, and the Windows Embedded logo are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

© 2009 Microsoft Corporation, All rights reserved.

Contents

Application Test Specification	1
Disclaimer	2
Contents	3
Welcome.....	4
Requirement Checklist.....	5
Requirements	6

Welcome

Welcome to Windows Embedded POSReady 2009 Test for ISV Solutions Application Test Specification, describing the technical requirements of the Platform Test for ISV Solutions. This test is intended to ensure that software from Independent Software Vendors (ISVs) install and function smoothly on the Windows Embedded POSReady 2009 operating system.

The Platform Test for ISV Solutions is required for software applications from Independent Software Vendors (ISVs) that meet the certification requirements for ISV's to be listed in the Windows Embedded POSReady 2009 Product Catalog.

Requirement Checklist

Applications must comply with all fundamental requirements to pass testing for this test component.

1. Executes on Windows Embedded POSReady 2009 and maintains stability while performing primary functionality.
2. Uses Windows Resources (heaps, page heaps, locks, and handles) appropriately.
3. Does not attempt to replace files protected by Windows File Protection

Requirements

1. Execute on Windows Embedded POSReady 2009 and maintain stability while performing primary functionality.

TEST OBJECTIVES

Applications must execute on Windows Embedded POSReady 2009 and perform their primary functions as expected without crashing or causing the user's computer to crash, fail, or function improperly.

A crash is any failure within a server component or service that either causes data loss or forces unscheduled downtime of the server or service. A crash within a client component or utility component is considered to be an application failure that prevents the user from continuing. A failure within a server component or service will not be considered a crash if it meets both of the following conditions:

- a) Does not cause loss of data,
- b) Does not force shutdown or unscheduled downtime for any server or service.

A failure within a client component or tool will not be considered a crash if it meets all three of the following conditions:

- a) Does not cause loss of data,
- b) Displays information that would allow a typical user to understand what went wrong and how to avoid the problem in the future
- c) Allows the user to continue running the application or close it.

2. Use Windows Resources (heaps, page heaps, locks and handles) appropriately.

TEST OBJECTIVES

The heap, critical sections, and handles can be misused, resulting in less reliable applications and failures with subtle circumstances that affect customers but may not be easily reproducible. You can easily test each of these items to ensure they are not misused. Applications must not misuse these resources in any way that could ever have potential negative consequences.

HEAP USE

Dynamic memory allocations come from the heap. Heap errors can result in security holes and can cause an application to fail. There are several invalid ways to use the heap, including:

- Allocating memory but writing beyond the end of the allocation (buffer overruns)
- Using allocated memory after it is freed
- Freeing an allocation twice
- Freeing unallocated memory
- Using wrong heap pointers

CRITICAL SECTION USE (LOCKS USAGE CHECKING)

Critical sections are user mode synchronization primitives that guarantee exclusive access to application data in a multithreaded environment. Invalid uses of critical sections include:

- Releasing a critical section that the current thread does not own

- Terminating threads while they own critical sections
- Using a critical section before being initialized
- Leaking critical sections (for example, did not call `DeleteCriticalSection`)
- Double initialized critical sections

HANDLE USE

Kernel handles — including handles to files, events, and so on — can also be misused in the following ways:

- Reusing a handle after being closed
- Using a handle for an operation that requires another handle type (you cannot read from an event)
- Using a random handle value
- Using a null handle or a pseudo-handle — for example, values returned by `GetCurrentProcess()` — when it is not permitted

To see why these kinds of errors can have bad consequences, consider the example of reusing a handle after it is closed. When a handle is closed, the system will reuse the value previously assigned. Assume that you have a file handle open and you close it, but you keep the value of the handle in some global variable. If some other part of the process opens a file handle for a totally different reason, perhaps even from external code, the new handle might contain the same value. If you still hold the old value in a variable and continue to use it, you may write into the wrong file.

Note: VeriTest uses Microsoft’s Application Verifier (AppVerifier) tool for this test. To read more about AppVerifier, please visit msdn.microsoft.com, and search for “AppVerifier”.

3. Do not attempt to replace files under Windows File Protection.

TEST OBJECTIVES

- Perform the initial application installation without attempting to replace any files protected by Windows File Protection (WFP).
- Perform any just-in-time installations without attempting to replace any files protected by Windows File Protection.

The application must not attempt to replace any files that are protected by Windows File Protection (WFP). To ensure that the application does not invoke WFP, it should call `SfclsFileProtected` when installing any file that it did not create. The Windows Installer service does this automatically.

Protected files include the following files that ship on the Windows Embedded POSReady 2009 product DVD:

- Most `.SYS`, `.DLL`, `.EXE`, and `.OCK` files.
- The following fonts: `Micros.ttf`, `Tahoma.ttf`, `Tahomabd.ttf`, `Dosapp.fon`, `Fixedsys.fon`, `Modern.fon`, `Script.fon`, and `Vgaoem.fon`.

Note: Some redistributable files, such as specific versions of Microsoft Foundation Classes (MFC) DLLs, are installed by Windows Embedded POSReady 2009 and are protected by WFP.

Protected files form the core of the operating system and it is essential for system stability that the proper versions are maintained. These files can only be updated through service packs, operating system upgrades, Quick Fix Engineering (QFE) hotfixes, and Windows Update. Applications cannot replace them, and attempting to replace these files by any means other than those listed above will result in the files being restored by the Windows File Protection feature (see the subsection About Windows File Protection, below).

If the application requires newer versions of these components, it must update these components by using a Microsoft Service Pack that installs the required versions.

EXAMPLE: When Microsoft publishes an update to DirectX, it will be provided in a package (either a Windows service pack or its own service pack). An application including the updated DirectX must use the package install and not attempt to directly install files from the package. Installing individual files is not allowed; in addition, Windows File Protection would prevent it and the user experience would be poor.

ABOUT WINDOWS FILE PROTECTION

Windows File Protection is a feature of Windows Embedded POSReady 2009 that prevents the unauthorized replacement of essential system files. WFP runs as a background process on Windows Embedded POSReady 2009 and monitors the files listed earlier in this section. When WFP detects that a protected file has been changed, it restores the original.

Do not prompt the user to update or delete any Windows File Protected components.

Note: Attempting to install components that are under Windows File Protection but have not yet been installed on the system will cause Windows File Protection to install the components. This is correct behavior.