

Windows Logo Program

Certification requirements for products that run on Microsoft® Windows® Vista operating systems

Requirements for the Windows Vista Logo Program for Software

This document outlines the requirements for the Certified for Windows Vista Logo Program. Please send any questions to swlogo@microsoft.com.

Microsoft

Version 1.1.0001
October 30, 2006

The information presented in these guidelines reflects Microsoft Corporation's views as of the date of publication. These views can and probably will change in response to changing market conditions. Microsoft makes no warranties or guarantees, implicit or explicit, in or by virtue of this document. Unless otherwise permitted by law, no part of this document may be copied without Microsoft's prior written permission.

© Microsoft Corporation 2006. All rights reserved.

Contents

Introduction	4
This Document	4
Program Overview	4
Security and Compatibility	5
Summary of Security and Compatibility Requirements.....	5
Windows Security and Compatibility Requirements.....	5
Installation	11
Summary of Installation Requirements	11
Installation Requirements	11
Reliability	20
Summary of Reliability Requirements.....	20
Summary of Changes	23
Resources	24
FAQ	25

Introduction

This Document

The purpose of this document is to outline the technical requirements an application must meet in order to become Certified for Windows Vista. For details on how to test an application, please refer to the Certified for Windows Vista Test Cases document.

Program Overview

The goal of the Vista Logo Program is to increase the quality of applications across the Windows ecosystem. The requirements in this document outline criteria that help make an application more compatible, reliable, and secure when running on the Windows Vista operating system. Applications that meet all of these requirements can carry the Windows Vista Logo artwork. This artwork is not yet available.

The Windows Vista Logo Program is one of certification. Applications must be independently tested by a Microsoft approved testing vendor before they are granted logo certification. As part of this process, a Logo Disclosure Document will be created for each application. That document will be created by the ISV applying for the logo and will be verified by the testing vendor. The document will contain information about the application's behavior and how the application passed certain requirements. When a Vista Logo Program requirement calls for behavior to be documented, the information must be included in the logo document. Logo documents will be made publicly available for review by customers.

Security and Compatibility

Summary of Security and Compatibility Requirements

Customer Benefits

The requirements that follow will improve the overall security of applications and help ensure that they will work with Windows Vista when it is run on different architectures, under different configurations, and in different modes. The requirements also help ensure that applications will continue to work properly as the operating system is serviced or upgraded.

List of Windows Security and Compatibility Requirements

- 1.1 Follow User Account Control Guidelines
- 1.2 Support x64 Versions of Windows Vista
- 1.3 Sign Files
- 1.4 Sign Drivers
- 1.6 Perform Version Checking Properly
- 1.8 Support Concurrent User Sessions
- 1.10 Avoid Loading Services and Drivers in Safe Mode
- 1.11 Follow Anti-Malware Policies

Windows Security and Compatibility Requirements

1.1 Follow User Account Control Guidelines

Criteria

To meet this requirement, every executable file (with a .EXE extension) included with an application must have an embedded manifest that defines its execution level:

```
<requestedExecutionLevel level="asInvoker|highestAvailable|requireAdministrator"  
uiAccess="true|false"/>
```

If only a small number of features in an application will require administrative privileges (For example, an application needs to configure a firewall), the main process of the application must still be run as a standard user. The administrative features must be moved into a separate process that runs with administrative privileges.

Applications that run their main process with elevated privileges (requireAdministrator or highestAvailable) must receive a waiver from Microsoft to obtain certification. Only system tools should apply

for this waiver. Games, office productivity applications, messaging clients, etc. will not be considered.

Applications that need to drive input to the UI of another window may set `uiAccess=true`, but will also need to apply for a waiver. Only accessibility applications will be considered for this waiver.

Rationale

A user's Windows experience is more secure when applications run only with the permissions needed.

Additional Information

Applications should follow the best practices for working with User Access Control as described on MSDN:

<http://msdn.microsoft.com/windowsvista/default.aspx?pull=/library/en-us/dnlong/html/AccProtVista.asp>

Applications that attempt to circumvent UAC by modifying the ACLs of the system will fail the tests for this requirement.

1.2 Support x64 Versions of Windows Vista

Criteria

To maintain compatibility with x64 versions of Windows, it is necessary that:

- Applications and their installers must not contain any 16-bit code or rely on any 16-bit component, since 16-bit code will not run on 64-bit versions of Windows Vista.
- If an application is dependent on kernel-mode drivers for operation, x64 versions of these drivers must be available. The application setup must detect and install the proper drivers and components for the 64-bit Windows OS.

WOW64 will allow 32-bit code to run on 64-bit versions of Windows, so it is not necessary that the application run natively on x64 versions of Windows.

Rationale

Many Windows Vista users will be running x64 versions of the operating system, so it is crucial for applications to be compatible with this OS.

Additional Information

It is acceptable to remove some features from an application when it installs on an x64 version of Windows, as long as these features are not part of the application's primary functionality.

1.3 Sign Files

Criteria

All executable files must be signed with an Authenticode certificate. This includes files with the following extensions: exe, dll, ocx, sys, cpl, drv, scr

Limited waivers may be available for 3rd party redistributables when signed versions are not available.

Rationale

Signing helps users decide if they want to trust an application, and assures users that files have not been tampered with. It also allows applications to run properly in locked-down environments.

Additional Information

Code Signing certificates are available from several vendors, as described at:

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsecure/html/rootcertprog.asp>

Once the certificate has been obtained, these files should be signed using the Sign Tool that comes with the Windows SDK:

<http://windowssdk.msdn.microsoft.com/library/>

1.4 Sign Drivers

Criteria

Any kernel-mode drivers that are installed by the application must have a Microsoft signature obtained through the WHQL or DRS program.

Rationale

Poorly written or malware drivers can severely affect the stability and security of a system. On 64-bit versions of Windows Vista, unsigned drivers will not be able to load.

Additional Information

More information about these driver signing programs can be found at:

<http://www.microsoft.com/whdc/winlogo/hwrequirements.mspx>

1.6 Perform Version Checking Properly

Criteria

An application must not fail to run if the OS version number increases, unless the End User License Agreement prohibits use on future operating systems. If the application is supposed to fail, it must do so gracefully by sending a message to the user and writing a message to the NT event log.

Applications must use the version-checking APIs (GetVersionEx) to check the OS version. They must not read the following the registry keys for the OS version:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows  
NT\CurrentVersion
```

Rationale

When Windows users upgrade their operating system, they shouldn't be hard blocked from using their current applications simply because the OS version has increased.

1.8 Support Concurrent User Sessions

Criteria

- If an application does not support running in multiple user logon sessions, it must alert the user of this fact before failing and write a message to the NT event log. This includes user sessions created using Remote Desktop Connection.
- Applications must allow multiple sessions unless prevented by architecture issues. If an application does not support this scenario, the user must be alerted upon failure. Additionally, applications that rely on 3D graphics are not required to work

over a Remote Desktop Connection, but the user must be alerted upon failure.

- Applications must correctly check for multiple application sessions. If multiple application sessions are unsupported, the application must check both user session and task information. This will avoid the blocking of application use in concurrent user sessions.
- Sound from an application in one user session must not be heard when another user is active in the current session.
- Applications must support fast user switching

Rationale

Windows users should be able to run concurrent sessions without conflict or disruption.

Additional Information

For more information about Fast User Switching:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/apcompat/apcompat/test_your_application_with_fast_user_switching.asp

1.10 Avoid Loading Services and Drivers in Safe Mode

Criteria

If any of the following subkeys are set, the drivers and services must run in safe mode without any errors:

HKLM/System/CurrentControlSet/Control/SafeBoot/Minimal
HKLM/System/CurrentControlSet/Control/SafeBoot/Network

If any of these keys are set, they must be listed in the logo documentation.

Rationale

Safe Mode is designed to allow users to diagnose and repair their Windows configurations. Drivers and services should not run in safe mode unless they are needed for basic operations of the system (for example, file system drivers) or for diagnostic and recovery purposes (for example, anti-virus scanners).

1.11 Follow Anti-Malware Policies

Criteria

All applications must meet the privacy guidelines put forth by the Anti-Spyware Coalition.

Applications that are considered malware or spyware are subject to losing their logo certification.

Rationale

Malware is a major concern for customers. When they buy software that carries the Vista logo, they should have some assurance that it is free of malware.

Additional Information

For anti-malware policies, see:

<http://www.antispywarecoalition.org/documents/index.htm>

Installation

Summary of Installation Requirements

Customer Benefits

Customers can be confident that applications will install on Windows without degrading the operating system or other applications.

Installation Requirements List

- 2.1 Use Windows Components for Installation
- 2.2 Support User Account Control for Installation
- 2.3 Install to Correct Folders
- 2.4 Sign ClickOnce Manifests
- 2.5 Keep ClickOnce Installations Isolated
- 2.7 2.7 Correctly Configure Package Identity
- 2.9 Install Windows Resources Properly
- 2.10 Follow Best Practices for Creating Custom Actions
- 2.12 Avoid Reboots During Installation
- 2.13 Support Command Line Installation
- 2.14 Follow Component Rules

Installation Requirements

2.1 Use Windows Components for Installation

Criteria

Applications must use the Windows Installer (MSI) or ClickOnce for installation.

Windows Installation packages must not receive any errors from the Internal Consistency Evaluators (ICEs) listed here:

1-24, 27-31, 33-36, 38, 40-57, 59, 61-63, 65, 67-72, 74-84, 86-87, 89-94, 96-99

Rationale

Using the built-in installation engines creates consistent, reversible, transacted installations. It is inherently safer to use operating system components to make changes to system configurations. There are fewer compatibility and application migration issues because these engines, unlike installation scripts, can evolve with the operating system. Additionally, enterprise customers find that these component-driven installations greatly reduce their deployment and repackaging costs.

Additional Information

A variety of third-party tools are available that can be used to create Windows Installation packages.

The ICES can be run from the Orca application, which ships with the Windows SDK:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/msi/setup/internal_consistency_evaluators_ices.asp

2.2 Support User Account Control for Installation

Criteria

An installation must not assume the installation is being performed in the user context of the user who will ultimately run the application. All applications that are installed system wide must be installed by a full administrator context (even when invoked by a protected administrator), thus each application must do first run tasks separately from install.

Applications that are using a non-self-containing bootstrapper/chainer must provide an embedded manifest for the executable that designates the desired execution level for the installer. For bootstrappers that explicitly do not require elevation, the `requestedExecutionLevel` should be set to `asInvoker` (`leastPrivilege` in Beta 1), and `uiAccess` should be set to `false`.

```
<requestedExecutionLevel  
  level="asInvoker"  
  uiAccess="false"/>
```

Other levels: `highestAvailable`, `requireAdministrator`

If an application is launched by the installer after installation is complete, the application must be launched in the context of the original user.

Rationale

One of the largest changes to the Windows operating system for the Vista release is the addition of User Account Control, which by default will run applications with reduced privileges. Installers need to manage their privilege levels accordingly.

2.3 Install to Correct Folders

Criteria

Applications should be installed to Program Files or the user's AppData folder by default.

For per-machine installations, user data must be written at first run and not during the install. Since the install potentially elevates to a different user account during the per-machine install, there is no correct user location to store data at install time.

Rationale

Users should have the flexibility to install applications where they need them and should have a consistent and secure experience with the default location of files.

2.4 Sign ClickOnce Manifests

Criteria

ClickOnce packaged applications must be signed with a valid Authenticode certificate.

Rationale

Because ClickOnce applications can be deployed from outside the local machine, it is particularly important that Windows users have assurance that they are running only the ClickOnce applications they are expecting.

Additional Information

This requirement applies only to applications that use ClickOnce for deployment.

2.5 Keep ClickOnce Installations Isolated

Criteria

ClickOnce applications must store data only in user folders. Any registry keys written by a ClickOnce application must be documented.

Rationale

ClickOnce applications are designed to have minimal impact on a user's system.

Additional Information

This requirement applies only to applications that use ClickOnce for deployment.

2.7 Correctly Configure Package Identity

Criteria

- Unless installed via bootstrapper or chainer and the ARPSYSTEMCOMPONENT property set as described below, the application must be correctly identified in Software Explorer (formerly known as Add/Remove Programs).

For correct Software Explorer identification, the application must supply all the information in the table below so that Software Explorer can obtain information about the application as needed. These values can be set using properties in the Windows Installer-based package.

If the application is installed via a bootstrapper or chainer, the ARPSYSTEMCOMPONENT property can be set in the Windows Installer packages. The legacy Uninstall keys should be written so that they point to the bootstrapper or chainer.

Property (property names are case sensitive)	Corresponding registry value	Contains
ProductName	DisplayName	Display name of application
Manufacturer	Publisher	Publisher/Developer of

		application
ProductVersion	VersionMajor	Major version number of application
ProductVersion	VersionMinor	Minor version of application

- To prepare for an application upgrade, the product must have [UpgradeCode](#), [ProductCode](#), [ProductVersion](#), and [ProductLanguage](#) properties for an upgrade to appropriately address the previous package.
- To prevent the product from being downgraded, the product must follow Platform SDK guidance for [Preventing an Old Package from Installing Over a Newer Version](#), since downgrading is not what users expect from a package install.

Rationale

When the installation package identifies itself correctly, Windows users can manage their applications more easily when servicing or upgrading the application or upgrading the operating system. Proper identification also provides a consistent experience for removing applications.

Additional Information

This requirement applies only to applications that use Windows Installer for installation.

2.9 Install Windows Resources Properly

Criteria

Applications must not attempt to install files or registry keys that are protected by Windows Resource Protection (WRP). If the application requires newer versions of system components, it must update these components by using a Microsoft Service Pack or a Microsoft-approved installation package containing the system component. System components must never be repackaged.

Rationale

Windows Resource Protection (WRP) is designed to ensure that protected system resources are updated only by using Microsoft-approved installation or update mechanisms. Following WRP guidelines will improve system reliability by ensuring that the results of an installation are predictable.

2.10 Follow Best Practices for Creating Custom Actions

Criteria

- The intended behavior of each custom action must be documented.
- Nested install custom actions ([type 7](#), [type 23](#), and [type 39](#)) have proven to be excessively problematic for application installs and must not be used.
- Custom columns must not be added to standard tables. Adding columns to standard tables is reserved for future versions of Windows Installer.
- Gacutil must not be called from a custom action. Gacutil is not designed to be used during installation.
- Return codes from custom actions are insufficient to record the success or failure of a custom action, so the outcome of custom actions must be recorded in the log using [MsiProcessMessage](#).
- Custom tables or properties must not be prefixed with 'msi' (case insensitive), as this prefix is reserved for future use in new standard tables and properties.
- Custom actions that change system state must be written as a deferred and rollback custom action pair.

Rationale

Custom Actions are a powerful tool to extend the capability of installations. Following the best practices outlined below will ensure that customers experience the same level of quality with custom actions as with standard actions. This does not apply to custom action basic types [19](#), [35](#), and [51](#).

Additional Information

This requirement applies only to applications that use Windows Installer for installation.

2.12 Avoid Reboots During Installation

Criteria

If the application install has ForceReboot actions, this must be documented with a justification.

Files-in-use dialogs authored into Windows Installer packages must contain an option to “Automatically close applications and attempt to restart them after setup is complete.” This additional functionality allows users to avoid reboots through the Restart Manager infrastructure.

Rationale

Rebooting the system after an install is an inconvenient disruption for users and is usually unnecessary. It can be particularly bothersome for machines that require frequent servicing and high availability.

Additional Information

This requirement applies only to applications that use MSI for installation.

For almost all application installs, the Windows Installer ForceReboot action should not be used.

For more information, visit this link:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/msi/setup/using_windows_installer_with_restart_manager.asp

2.13 Support Command Line Installation

Criteria

Applications should successfully install in quiet mode via a command line with /qn switch. Any necessary command line options need to be documented.

Applications should successfully install via command lines with FASTOEM=1. Any necessary command line options need to be

documented. Alternatively, an ISV may document other fast install capabilities.

Rationale

Requiring a graphical user interface for installation can block several installation scenarios. Applications should support command line installation to better enable deployment via SMS or other enterprise tools.

Additional Information

This requirement applies only to applications that use Windows Installer for installation.

2.14 Follow Component Rules

Criteria

- If the Component table's component field is left null, justification must be documented.
- There must not be more than one COM server included per component. If a component contains a COM server, this must be the KeyPath for the component.
- There must not be more than one file specified per component as a target for the Start menu or a Desktop shortcut.

Rationale

To ensure that the installation or removal of one application does not harm any other applications on the system and that the Windows Installer service correctly removes all resources connected with that program, an application must adhere to the component rules.

Though the Component table's ComponentId field is nullable, a component with a blank ComponentId is not uninstallable or manageable via the Windows Installer APIs.

Additional Information

This requirement applies only to applications that use Windows Installer for installation.



Reliability

Summary of Reliability Requirements

Customer Benefits

These requirements make an application more reliable by minimizing the number of crashes, hangs, and reboots experienced by users. The reliability requirements can help ensure that software is more predictable, maintainable, resilient, recoverable, and proven.

Reliability Requirements List

- 3.1 Maintainable Software: Eliminate Unnecessary Reboots
- 3.2 Resilient Software: Eliminate Application Failures

3.1 Maintainable Software: Eliminate Unnecessary Reboots

Criteria

- All application installers must take advantage of the Restart Manager APIs to avoid system reboots (see requirement 2.12). If an installer uses MSI without any custom actions, this functionality is provided.
- All applications must be “restart manager aware” by listening and responding to the following shutdown messages:
 1. WM_QUERYENDSESSION with LPARAM = ENDESESSION_CLOSEAPP(0x1): GUI applications must respond (TRUE) immediately in preparation for a restart.
 2. WM_ENDSESSION with LPARAM = ENDESESSION_CLOSEAPP(0x1): Applications must return a 0 value within 30 seconds and shutdown.
 3. CTRL C: Console applications that receive this message should shutdown immediately.

Rationale

System reboots are a major disruption and lead to a bad user experience. By registering for a restart and listening to shutdown messages, applications can avoid causing system reboots after installs and updates.

Additional Information

The Windows Installer should be used for application installs and patch installs. Reboots during installation should be avoided (see requirement 2.12).

For more information about Restart Manager, visit:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/rstmgr/rstmgr/about_restart_manager.asp

3.2 Resilient Software: Eliminate Application Failures

Criteria

- Any crashes or hangs that occur during the logo certification process must be fixed.
- Any errors found in the following Application Verifier tests must be fixed:
 - Basics: Exceptions, Handles, Heaps, Locks, Memory, TLS
 - Miscellaneous: DangerousAPIs, DirtyStacks
- Applications must handle only exceptions that are known and expected, and Windows Error Reporting must not be disabled. If a fault (such as an Access Violation) is injected into an application, the application must allow Windows Error Reporting to report this crash.
- ISVs must sign up to receive their crash data from Windows Error Reporting. This can be accomplished by signing up at: <https://winqual.microsoft.com>. ISVs must maintain map their certified applications to their company at this site and maintain those mappings while the product is supported. ISVs must target fixing 60% of their crash volume over the life of the product and maintain an average fix rate of 10 buckets per month or greater until this target is reached. (Buckets with fewer than 200 hits can be disregarded.) When fixes are available through patches or service packs, a response must be submitted through the Winqual site.

Rationale

Application failures such as crashes and hangs are top drivers of customer dissatisfaction. Eliminating such failures improves application stability/responsiveness and reduces loss of time, work, data, and control for the user.

Additional Information

The Application Verifier can be downloaded from:

<http://www.microsoft.com/downloads/>

In order to meet the ongoing reliability requirement above, developers must register with the Windows Developer Portal (<https://winqual.microsoft.com>) and get access to customer feedback data on application failures. After registration, developers must specify ownership of failure events so that reports can be created for failures in their applications. This can be done by mapping the binary name and version used to identify crash signatures with an organization. Once the failure events have been mapped to an organization, developers can view top failures for applications based on volume and growth. It is important to regularly review top failures reported through the Developer Portal, provide fixes for these problems, and notify users by creating responses from the portal.

Summary of Changes

Version 1.1

- Requirement 1.3 was modified to describe the waiver process for unsigned 3rd party files
- Provided clarification to requirement 2.2.
- ICE 39 was removed from requirement 2.1.

Resources

Windows Vista SDK

<http://windowssdk.msdn.microsoft.com/library/>

User Account Control Guidelines

<http://msdn.microsoft.com/windowsvista/default.aspx?pull=/library/en-us/dnlong/html/AccProtVista.asp>

ClickOnce Information

<http://msdn.microsoft.com/clickonce>

Windows Installer Information

<http://go.microsoft.com/fwlink/?LinkId=3985>

Also see:

[Sending Messages to Windows Installer Using MsiProcessMessage](#)

[Returning Error Messages from Custom Actions](#)

[Deferred Execution Custom Actions](#)

[Changing the System State Using a Custom Action](#)

[Organizing Applications into Components](#)

[Changing the Component Code](#)

[What happens if the component rules are broken?](#)

Privacy Information

<http://www.antispywarecoalition.org/documents/index.htm>

Developer Portal

<https://winqual.microsoft.com/member/wer/user/default.aspx>

FAQ

Why aren't the requirement numbers continuous?

Some requirements were cut from previous drafts. For consistency, these numbers are skipped in this draft.

Does every executable need to have a manifest that defines its execution level?

Yes.

Does the application need to be run natively in 64bit or is enough to run well under WOW64 to pass the certification?

It is acceptable to rely on WOW64.

Can I use an EXE to bootstrap my MSI install?

Yes.

Is it acceptable to use a third party tool (such as InstallShield) to create my installation package?

Yes. Your package should still be MSI-based and needs to meet all of the requirements.

If my application already has error reporting capabilities, is this acceptable for requirement 3.2?

No, applications must use Windows Error Reporting. This will enable more reporting scenarios and will maintain compatibility with Windows.

Where can I find more information on Windows Error Reporting and its APIs?

https://winqual.microsoft.com/help/wer_help/dev.aspx

When a requirement asks for documentation, where is this documented?

This should be documented in the logo document as described in the introduction.

My application does not meet all of the requirements, can I get an exception?

No, exceptions will not be granted. Limited waivers are available for specific requirements.