MSI Testing

BY BOB KELLY

As your organization's catalog of packages grows, the potential negative impact of a new package on your existing application base also increases. Thus, a common question arises: "What should I test?" Although the answer will differ depending upon the nature of your environment, there are a few ways to approach a solution:

- Test everything–Maintain a checklist or automation script that exercises the primary functions of each application with which your new package will exist. Typically, this checklist will grow over time as you continue to add items deemed necessary for testing.
- Test mission-critical applications–Focus testing on those applications that are most critical to your network.
- Test applications that share common files–After running conflict analysis, determine which applications share files; typically, these are DLLs and OCXs. Applications that use shared files are the most likely to be impacted. For resource-critical situations, testing these applications alone is often a sensible approach.
- Thoroughly test your package–Application conflicts aside, a thoroughly tested package will provide the highest level of confidence in the deployment of your new application.

This article discusses issues surrounding the testing of MSI packages and how you can take advantage of the Test Expert tool from Wise Package Studio to more thoroughly test your packages prior to deployment.

HOW TO THOROUGHLY TEST A PACKAGE

Before the advent of third-party testing tools, determining what to test and the best way to thoroughly test a package was a difficult task. Today's IT staff is aided with this task by products that not only break down a package into logical test elements but also provide a means of triggering and tracking the testing of each element. Third-party testing tools offer testing options for the following package elements:

- Package installation—Verification of the installation itself
- Standard tests—Verification of network and database connectivity, network location, and program execution
- Application verification—Verification of ClassIDs, file extensions, Help files, ProgIDs, and shortcuts
- Application execution—Verification of execution and identification of run-time loaded and/or created files and registry entries
- Uninstall tests—Verification that the uninstall removes residual files and registry entries

The following scenarios provide an overview of the effectiveness of using a third-party testing tool. In these examples, Test Expert is used to illustrate the testing process.

Package Installation Tests

Package installation tests verify that the installation runs on your local computer without errors. To perform this test, click Install from the toolbar to initiate the installation. To run as a user with different privileges, click Install As instead. When you are done, update the status of this test from Pending to Passed (or Failed).

If you run into a problem here, step back and examine the package contents in the Windows Installer Editor. You might also expose problems with your package using the Package Validation tool, which examines the makeup of the MSI database for any problems or rules that are not being followed.

! One tool that offers these testing capabilities is Wise Package Studio's Test Expert. This tool starts by displaying test plan details in the form of pie charts and tables. The graphics show the current progress of the testing process as well as a view of test results (successes vs. failures).

Windows Installer offers extensive logging options. You can specify what to log at a very granular level or you can log everything by specifying an asterisk (wildcard):

```
msiexec /i MyPackage.msi /l* C:\Packages\MyPackage\MyPackage.log
```

In addition to being able to specify logging options at the command line, you can also instruct your system to log the installation of all Windows Installer packages as a Windows Policy. To make this change directly in the registry, you simply make the following change:

Subkey: HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\Installer
Value: Logging (Type: REG_SZ)
Data: voicewarmup

Voicewarmup is the long way to specify all values; each letter represents a different logging option.

The log file can be a bit ugly to look at, but at times, it is the best place to find information beyond what is presented in any error dialog box you encounter. Microsoft provides a helpful viewer for these logs in the Windows Installer Software Development Kit (SDK—Wilogutl.exe).

! Some test cases might have to be run on multiple computers. For example, to test a launch condition that is specific to Windows XP, you must install not only on a computer running Windows XP but also on a non-Windows XP computer. To easily switch computers, while providing easy access to test case information, you can perform a Wise Package Studio client installation.

Standard Tests

If your application (or its package) needs to access the Internet, a database, or a local network share, or needs to execute a file, you can use standard tests to address any concerns. For this set of tests, you will need to add any items manually. Once an item has been added, it will appear within the check box list. You select the item by selecting its check box in the list, right-clicking the item, and clicking Execute. Once you complete this set up and run each test, the success or failure will be updated automatically upon each test's execution.

The only pre-populated test in this section is that of an Internet check (for the Wise home page). If your application does not need Internet connectivity, simply delete this test and proceed to the next section.

Application Verification

Among all the tests offered, the application verification section of tests is most valuable. Tracking each of these items manually is a daunting task that is made quite simple through the use of a third-party tool. The following list highlights available application verification tests:

- ClassIDs—This test checks whether COM objects declared by a ClassID can be created. This test might expose problems such as missing DLLs that are required by the ClassID to create the object. The ClassID test might be run automatically if the package being tested is currently installed.

- File extensions—This test creates a list of verbs associated with each extension in the package and prompts you to select a file type for each one. It then attempts to execute each verb so that you can visually determine whether the verb performs as expected.

- Help files—If the package being tested is installed, this test verifies that application Help files can be opened by allowing you to execute any CHM or HLP files located in your package.
- ODBC data sources—This test allows you to check whether an ODBC data source has been created correctly and whether the data source is accessible.
- ProgIDs—Similar to the ClassID test, this test allows you to check whether COM objects declared by a ProgID can be created.
- Search Locations—If search locations are detected in a package, this test verifies that the search locations exist and are accessible.
- Services—This test verifies that any included services are installed and correctly registered with the Service Control Manager.
- Shortcuts—This test verifies that shortcuts in the installation are attached to a valid target file and that the target opens when the shortcut is selected.

Application Execution

In addition to these tests, there are tests that allow you to execute your application and verify which files and registry entries are accessed by your package. These tests all run simultaneously to monitor the testing computer as you exercise the features of the application. Test Expert will identify files and registry keys that are accessed by the application but not installed by the package—the use of third party tools is not required.

For the item coverage tests, a tool will record which installed files are accessed during application execution so that you can identify files that might not be required by your package.

Uninstall Tests

The uninstall tests also run simultaneously to determine how the uninstall sequence in the package is performing. It detects created, destroyed, and residual items pertaining to the removal of the package. Created items include all files and registry items created by the application when it was opened as well as any files that were created while testing. Destroyed items identify items that existed on the computer before installation but were missing after uninstall. Finally, residual items refer to those elements of your package that were installed but not removed by the removal of your package. For the most part, these tests will not identify any serious problems with your package but are helpful to determine whether your package will do a good job of cleaning up after itself.

SUMMARY

The damage that can be done by a poor installation package to both your network and your package and deployment team's reputation are difficult to measure. Thus, putting your package through thorough testing as part of your quality assurance (QA) process is an essential step in ensuring success and will provide you with confidence in the performance of your MSI setup.

ABOUT THE AUTHOR

Bob Kelly is the founder of AppDeploy.com, a resource focused on desktop management products and practices. He is author of the Start to Finish Guide to Scripting with KiXtart and The Definitive Guide to Windows Desktop Administration. Bob is co-founder and president of iTripoli, Inc., a software company focused on producing helpful tools for administrators, including Admin Script Editor, a scripting editor for Windows administrators, and PackageCleaner, a tool that identifies MSI package contents that may pose problems if included in Windows Installer packages. Bob currently develops training videos and provides limited-engagement consulting services via AppDeploy.com.