



Software installations made easy™

Repackaging Basics

A Report by Wise Solutions, Inc.

Table of Contents

Abstract	1
The Importance of Repackaging	2
What is Repackaging?	2
Why Do Organizations Repackage Software?	2
Differences Between Software Repackaging and Software Development	3
Repackaging Best Practices	4
Capturing an Application	4
What Not to Repackage	4
Why is Documentation Important?	4
Creating the Proper Repackaging Environment	6
The Clean Machine	6
The Base Build Machine	6
Using an Exclusion List	7
Steps for Repackaging	8
Step 1: Review Packaging Requirements With the Project's Sponsor	8
Step 2: Analyze the Vendor Package	8
Step 3: Repackage the Application	9
Step 4: Customize the Package	10
Step 5: Test the Package	10
<i>Quality Assurance Testing</i>	11
<i>User-Acceptance Testing</i>	11
Step 6: Release the Package to End Users	12
Conclusion	13
Appendix A: Glossary of Common Repackaging Terms	15
Appendix B: About Wise Solutions	17

Abstract

This white paper provides an overview of a best practices approach to repackaging for Windows Installer in the corporate environment. The intended audience is system administrators and IT managers who are new to repackaging and want to implement a process-oriented approach to repackaging in their organization. The paper describes Wise Solutions' best practices methodology and recommendations for creating reliable packages that can be successfully distributed throughout an organization.

Best practices are a collection of commonly tried-and-true processes and procedures based on experience and knowledge in a variety of related situations. Wise Solutions has worked in the repackaging industry, and with repackaging experts, for a number of years. Throughout that time, Wise Solutions has gathered a wealth of knowledge in repackaging best practices that are embodied in its system administrator product, Wise Package Studio. While this paper highlights how to use Package Studio as part of this process, the general scope of the paper describes repackaging best practices in general terms that are not product specific.

The Importance of Repackaging

What is Repackaging?

Repackaging is the process of capturing the changes made by an installation program for the purpose of creating a new, customized installation. This new installation, which is called a “package”, is designed to support company standards and distribution methods. The repackaging process is critical to the success of software roll-outs. Repackaging saves time, effort, and money by allowing the system administrator to customize installations to meet the standards set for software distribution within their company. For instance, an administrator might repackage an application in order to prevent all dialog boxes from appearing during the installation. This prevents the end user from making an incorrect choice and installing the application in a manner that is inconsistent with the company’s standards. An incorrectly installed application can result in support calls to the system administrator or the help desk.

Why Do Organizations Repackage Software?

The main reason for repackaging an installation is to create a consistent, yet customized, installation. The main benefit of creating a consistent, customized installation is to reduce desktop support costs. As system administrators well know, the cost to support even one desktop computer is staggering. By standardizing the applications that are installed on company desktops, administrators reduce variability. Variability in end users’ desktops results in many unknown factors – if an end user installs a game will it conflict with the word processing software? If a user installs accounting client software into c:\fred rather than the appropriate application directory, it may take the system administrator some time to diagnose any problem that occurs.

Repackaging seeks to eliminate these types of issues through “customized consistency”. Customized consistency means customizing, or repackaging, an installation so that it behaves in a manner consistent with the company’s standards. Adherence to standards means that end user desktops are easier to support because the system administrator knows the software installed on them conforms to predefined standards that, in turn, reduce support costs.

Differences Between Software Repackaging and Software Development

Both system administrators and software developers create installations, however, the methods for how and why they create installations vary. Administrators repackage existing installations in order to customize the package to their company's standards for mass deployment. In contrast, software developers typically author new installations based on the requirements of the application they are installing. The following chart differentiates software developers and system administrators' requirements when creating installations.

Software Developers	System Administrators
<ul style="list-style-type: none"> - Method of installation creation: author new installations - Software developers typically author new installations from the ground up. 	<ul style="list-style-type: none"> - Method of installation creation: repackaging - Administrators customize an existing installation by repackaging it.
<ul style="list-style-type: none"> - Install on one PC - Target is individual PCs. 	<ul style="list-style-type: none"> - Install on many PCs. - An application can be distributed via a network, eliminating the need to walk to each PC and install it individually.
<ul style="list-style-type: none"> - Minimal integration with other applications - Little or no concern if the installed application breaks other applications. 	<ul style="list-style-type: none"> - Integration with other applications is critical - It is imperative that each installed application work with all other applications on the end user's PC.
<ul style="list-style-type: none"> - Support several platforms - Doesn't know specification of target platform, so application installation is designed for several platforms. 	<ul style="list-style-type: none"> - Support limited number of platforms - Knows target desktop specifications, so application installation only needs to run on specific platforms.
<ul style="list-style-type: none"> - Flexible installation - Installation can be installed where a user decides. 	<ul style="list-style-type: none"> - Inflexible installation - Requires the application to be installed in the same location every time.
<ul style="list-style-type: none"> - Media distribution - Typically distributes an application on a CD for each user. 	<ul style="list-style-type: none"> - Network distribution - Distributes an application through the network to multiple users.
<ul style="list-style-type: none"> - Extensive user input - Users interact with the installation and make their own choices about how the installation is installed. 	<ul style="list-style-type: none"> - No user input desired - The system administrator determines what features users will have and creates an automated installation that installs without user interaction.
<ul style="list-style-type: none"> - User-executed installation - User initiates the application installation. 	<ul style="list-style-type: none"> - Distribution system-executed - Distribution occurs through a network and can run silently on users' machines as they work, during the night, or when the user executes the application.
<ul style="list-style-type: none"> - Reports status to user - User receives constant messages about the application installation's progress. 	<ul style="list-style-type: none"> - Reports status to distribution system - User is not encumbered by application installation status reports.

As the chart shows, repackaging has very different requirements from authoring an installation from scratch. The next question is, what is the best way to fulfill these repackaging requirements? In Wise Solutions' experience, it involves implementing best practices that help ensure the creation of consistent, customized packages.

Repackaging Best Practices

Capturing an Application

A snapshot tool, such as Package Studio's SetupCapture, is often used to create packages. Snapshot technology refers to the process of performing a scan of a computer's hard drive before and after an installation is executed. The snapshot "captures" or records the changes made to the hard drive between the first and second scans. The difference between the scans is a "delta," or a package that contains all changes. This package takes the same actions as the original installation. The resulting package can be customized as required. It is an easy and efficient method for creating packages. However, using a snapshot tool, such as SetupCapture, is only one step in the repackaging process, as will be discussed later.

What Not to Repackage

Wise Solutions does not recommend repackaging certain types of applications. These include:

1. *.MSI files.*

Installations that are already in .MSI format should not be repackaged, but instead should be customized using a "transform." A transform customizes an .MSI installation without recreating it from scratch. Transforms are applied at install time and are used to change the default behavior of the installation. The original .MSI file is needed in order to create a transform. For more information about creating transforms, refer to Windows Installer SDK.

2. *Operating systems.*

Capturing is not appropriate for operating systems or service patches. Wise Solutions does not recommend repackaging these types of applications.

3. *Windows Media Player, Microsoft Internet Explorer, anti-virus software, and device drivers.*

All four types of applications listed make low-level changes to the operating system involving Windows File Protection. Wise does not recommend repackaging these applications.

Why is Documentation Important?

One of the keys to successful repackaging is documenting the steps in the process. This helps ensure that anyone involved in repackaging always adheres to the same process, which in turn, helps ensure consistency and reliability in packages. It is also an invaluable aid in training new packagers to follow the expected steps. Each of the items below involves documenting a specific step of the overall process. Each item is also discussed in more detail later in this paper.

Wise Solutions suggests documenting the following:

- *The Repackaging Environment*

One of the keys to a successful repackaging effort is having the correct development and testing environment for creating packages. Standardization of this environment is also critical; each repackaging team member should create and test packages on machines that have the appropriate software installed and identical hardware platforms. Consistency is important – each repackager and tester should use a machine with identical software and hardware. This helps isolate potential problems and aids in trouble-shooting. If conflicting results occur on machines that are set up differently, it will be difficult to trace the source of the problem. Documenting the repackaging environment's setup helps avoid creating and testing packages on machines with different configurations.

- *Company IT Standards for Repackaging*

Are there rules that must be followed in order to deploy a new application according to company standards? Wise Solutions recommends documenting these standards to ensure they are followed consistently with every package that is created.

Company standards may include the following:

1. Are shortcuts used for launching applications? If so, is there a naming convention for shortcuts?
2. Are there naming conventions for applications on the Start menu?
3. What security permissions are typically set on end users' PCs that could affect the manner in which packages install or how applications run?
4. How are packages distributed? Will this require additional customization of the package?
5. Do packages need to install silently (with no dialogs or user interaction) or quietly (no user interaction is desired but indication of the installation's progress is necessary)?

- *The General Repackaging Process*

It is important to have a standardized repackaging process to ensure reliability and consistency in the process and in the resulting packages. Without a documented process, repackaging team members could choose a variety of methods for repackaging without any guarantee that their results would be successful or reproducible. The repackaging process should be comprehensive and cover all steps in repackaging, including setting up the proper environment and the specific steps to take when packaging, testing, and troubleshooting packages.

One approach to documentation is to create a checklist of repackaging tasks that all repackagers are required to use. Another approach is to use a repackaging tool that tracks the steps in the process, such as Wise Package Studio. Package Studio contains a built-in checklist of processes and tasks for standardizing repackaging projects. As packaging team members complete a task, Package Studio automatically checks it off and moves to the next task on the list.

- *The Specific Repackaging Project*

This involves documenting the specific, unique steps that will be involved in a specific repackaging project. This is different than the general repackaging process discussed earlier. The general repackaging process should be the standard set of steps that are followed for every repackaging project. However, there are unique characteristics to each project that are determined by how end users use the repackaged application. To gather this information, interview the sponsor, which is the person who makes the initial packaging request. The sponsor is often a department supervisor who is familiar with how end users interact with the application. Use this to customize the package to the end users' needs and test it before it is distributed.

Creating the Proper Repackaging Environment

Creating the proper repackaging environment is important to the success of creating packages. It's important to standardize the software and hardware setup of a repackaging environment in order to isolate these variables so they don't affect packaging results. By not standardizing, it is difficult to guarantee the success or failure of packages and troubleshooting is a more challenging task.

Wise Solutions recommends setting aside specific, separate computers for package development and package testing. Each of these machines should be configured differently, because they require different software to be installed. The package development machine is called a "clean machine", and the package testing machine is called a "base build machine."

The Clean Machine

A clean machine creates more robust repackaged installations by making them less dependent on the existence of other applications. A clean machine is defined as a PC with only the minimum software installed as determined by a company's standards. The emphasis is on minimum — a clean machine does not contain business applications such as Microsoft Office or anti-virus software. It is also important to take your company's standards into account when setting up a clean machine; every company will have different requirements.

In general, a clean machine should have the following software installed:

- Operating system with the appropriate service pack that is installed on every PC in the organization.
- The version of Internet Explorer that is installed on every PC in the organization.
- A repackaging tool, such as Wise Package Studio. Installing Package Studio on a clean machine does not compromise the repackaging process as Package Studio files and registry entries are designed to not be included in captured packages.

A clean machine is the lowest common denominator for all packages that will be built. The clean machine will be restored frequently between packaging projects. Wise Solutions recommends using drive-imaging software to create an image of the clean machine in order to restore it quickly with a minimum amount of effort. It may be necessary to create images of multiple operating system configurations, depending on the needs of the organization.

The Base Build Machine

Use a base build machine for testing packages. The base build contains the software commonly found in an organization, this includes all the software installed on the clean machine plus the minimum software that would be installed on a machine before giving it to a new user.

A typical base contains the following software:

- Operating system with appropriate service pack installed on every PC in the organization.
- The version of Internet Explorer installed on every PC in the organization.
- Anti-virus software.
- Network software.
- Distribution client software.

The base build will also need to be frequently restored in order to ensure consistent testing results. It is easiest to image the base build with an imaging tool so it can be restored quickly. Store the images of both the clean machine and the base build in a central location accessible to all repackaging team members.

Using an Exclusion List

Using a good exclusion list helps ensure successful packages that produce consistent results. An exclusion list is a list of files, folders, registry keys, and registry values that should not be included in the captured installation. Exclusion lists are valuable because they help create clean installations by eliminating files, folders, registry keys, and registry values that are not related to the actual installation. Examples of items that can be excluded are file or registry changes related to operating system actions. A good exclusion list can also save hours of cleaning a repackaged installation and reduce the amount of quality assurance testing.

When creating an exclusion list, it can be difficult to discern the files, folders, registry keys and values to exclude. A general rule is, there is no harm in having extra files. There is, however, a substantial risk in removing files that contain vital information. That being said, there are some standard files and registry keys to include in an exclusion list. These include files related to the vendor's uninstall program, temporary Internet files, cookies, Internet Explorer history information, pagefile.sys, and user-specific files. Registry keys and values that can be removed safely include any user-specific keys, security provider entries, last known good entries, uninstall entries, and shutdown entries.

A snapshot tool, such as Package Studio's SetupCapture Configuration, can streamline the process by automatically analyzing a target installation for additional entries to exclude. To do this, start SetupCapture and open and close common applications that create files or registry entries unnecessary to the finished package. Once those unnecessary entries are captured, they can be added to the exclusion list.

The process for creating an exclusion list is as follows:

1. On a clean machine, start a snapshot tool such as SetupCapture.
2. Open Notepad. Notepad makes changes to the most-recently-used-file list in the Documents listing under the Start button. By capturing these changes now, they can be added to the exclusion list so they aren't included in the final package.
3. Open and close an Internet browser. This captures any registry keys that would change, related to proxies or firewalls, which should not be included in the final package.
4. Open Network Neighborhood and browse to various directories. This captures any registry keys that would change which should not be included in the final package.
5. Add an item to the Recycling Bin. This captures any registry keys that would change which should not be included in the final package.
6. Reboot the machine to capture changes the operating system makes during a reboot, which should not be included in the final package.
7. Complete the capture process. The resulting package is a good basis for creating a global exclusion list.
8. Save this list to a safe place for future reference and use.

In order for the exclusion list to be useful, it must be maintained and updated on a regular basis with any new files/registry entries that need to be excluded. As the repackager becomes more experienced, it will be easier to know what entries to add to the exclusion list.

Steps for Repackaging

The general steps for repackaging an application are:

1. Review packaging requirements with the project's sponsor
2. Analyze the vendor package
3. Repackage the application
4. Customize the package
5. Test the package
6. Release the package to end users

Step 1: Review Packaging Requirements With the Project's Sponsor

The first step in a successful repackaging project is to gather information from the sponsor of the project. This information is then included in a report that the repackaging team refers to throughout the repackaging process.

The sponsor is the representative of the end users who will use the application once it is rolled out for distribution. For instance, a senior sales engineer may be the sponsor of a software demonstration tool, because he/she manages sales engineers who frequently use it to create self-running sales demos. The sponsor should understand how to install the application correctly. Given that most application installations have a myriad of choices to choose from, this is very important information in order to repackage accurately. Without this information, the repackager will be unsure how to install the application correctly. It also makes testing efforts more robust, because testing can duplicate the common actions end users will take with the repackaged application.

The sponsor should answer the following questions, which the repackager compiles into a report:

1. Describe the application's target audience and how the audience typically uses the application.
2. Describe how to install the application correctly with the options end users require.
3. Describe how the application will be distributed.
4. List other applications that typically reside on the end users' machines. This is important for testing to ensure the repackaged application does not conflict with other applications.

Not only does the sponsor play a major role at this early stage, but he/she should also coordinate and lead end user testing of the repackaging application.

Documenting the information received during the sponsor interview is very important. This information will guide repackaging efforts and help make the process easier and more streamlined. If the sponsor is interviewed thoroughly in this phase, then repackers will have the necessary information to make decisions in subsequent phases. An alternative, or supplement, to documenting how end users use the application is to use a macro-recording tool to record how the application is commonly used. The macro can be referred to when testing the package to ensure that it works in the desired manner.

Step 2: Analyze the Vendor Package

The next step is to become familiar with the application's installation. The report created in Phase 1, which involved interviewing the sponsor, will be helpful in understanding how to proceed through the installation. The information learned in this phase should be added to the report for a thorough record of the actions to take when repackaging starts.

The goal in familiarization with the application's installation is to understand exactly how the installation functions. In other words, what happens on the PC when the installation is run? Questions to answer include:

What files are installed? What registry keys are created? Are shortcuts installed on the desktop? How is the program named in the Program group? Are certain user rights required to install the program successfully?

Based on the sponsor interview in Phase 1, the repackager should be familiar with how to answer every dialog box that appears during the installation. The best way to do this is to run the installation on a clean machine and review the entire installation sequence. The repackager should know how to answer any dialog that appears and understand what customizations are required. When the repackager is thoroughly familiar with how to install the application properly, the next step is to document it. This can be invaluable for troubleshooting issues in the package.

The best method for documenting the changes the installation makes to the clean machine is to run a packaging tool during this phase and save the resulting package. Following is a general method for doing this:

1. Run the packaging tool of choice on a clean machine.
2. Capture the vendor's installation; make the appropriate choices during the installation sequence that correspond to how the sponsor wants the application to be installed.
3. End the capture.
4. Save the resulting package file in a safe place for future reference. Package Studio prints an HTML report that lists all items and the sequence in which they were installed.
5. The repackager should read over the report to become familiar with actions taken during the installation.

Step 3: Repackage the Application

Completing phases 1 and 2 provides the information necessary to repackage the application's installation. All package development work should be done on a clean machine in order to not introduce extraneous dependencies into the package. For example, let's say that Microsoft Office is being repackaged on a PC that is not a clean machine. If there are files already installed on the repackaging machine that also exist in the Microsoft Office package, then it is possible that the capturing tool will not pick up the duplicated files. Therefore, the repackaged application will be dependent on the files that were already installed on the repackaging machine. To eliminate this issue, always repackage on a clean machine. Also, do not run any other applications on the clean machine while using the packaging tool, otherwise the integrity of the package is compromised.

Wise Solutions highly recommends using a package tool that uses exclusion lists. The importance of using a good exclusion list every time is critical, because it eliminates unnecessary files or registry entries from the newly created package.

Package Studio's SetupCapture uses a predefined exclusion list that can be used for all captures. This list includes files, directories, and registry entries that should be excluded from captures. For record-keeping purposes, an HTML report of the captured entries can be created and verified before the actual package is created. A robust capture tool such as SetupCapture re-uses a standard exclusion list each time an installation is captured, or the repackager can modify the list based on an application's particular requirements.

Here is the process for capturing an installation:

1. On the clean machine, start the packaging tool of choice. The packaging tool scans the hard drive of the development machine – this creates snapshot 1. In a later step, snapshot 1 will be compared to snapshot 2 to create the package.
2. After snapshot 1 is finished, start the installation to be repackaged. Answer all prompts in the installation based on information gathered in phases 1 and 2 of the repackaging process.

3. After the installation has completed, reboot the clean machine. The reboot is necessary in case there are application installation changes that will not be complete until after a reboot occurs. When the PC restarts, do not close the packaging tool. The next step is to customize the package.
4. Start the newly installed application. Make any necessary modifications to the application such as dismissing license agreement screens that appear the first time the application is started, turning off tool tips, and anything else that should not be included in the final package. When finished, exit the application.
5. Make any standard, environmental modifications. This is a logical place to customize the package so it satisfies a company's standards for installing and distributing applications (the same information covered in phase 1 of the repackaging process).

Steps to take may include renaming the shortcut that was created by the original installation and renaming the application's title from the default name. Also make any necessary security changes, such as moving information that was installed under a specific user's profile (such as Administrator) to the All Users profile so that any user running the application can access it properly.
6. After making all necessary modifications, exit the application. Restart the packaging tool so snapshot 2 can be created. Snapshot 2 will contain all the customizations that were done in steps 4 and 5, above. Once snapshot 2 is created, the packaging tool creates a package file that contains the differences between snapshots 1 and 2. This is the file that will be customized and tested before distributing to end users.
7. When the package is created, save it and exit the packaging tool.

Step 4: Repackage the Application

After finishing the capturing process, the repackager can customize the package in an .MSI editor or a script-based editor, depending on the format in which the package was initially created.

Customization steps might include:

- Creating a “silent” installation that does not require end user interaction. This involves suppressing the installation's dialogs so the installation runs unattended with no feedback or progress indicators appearing on the user's screen.
- Creating a “quiet” installation. Unlike a silent installation, a quiet installation displays some type of installation progress indicator on the end user's screen. This lets users know that a process is running on their machine so they don't inadvertently reboot while the package is installing.
- Editing feature names.
- Changing system requirements for the destination machines on which the package will be installed.
- Setting system requirements, such as requiring the destination machine to have Windows 98 or a higher operating system installed, for the installation to run.

After the repackager makes modifications, the installation should be tested to be sure it compiles without errors and functions in the intended manner.

Step 5: Test the Package

Testing is an important step in creating a robust package that installs and functions correctly. Complete package testing requires two kinds of tests: quality assurance testing by members of the repackaging team and user-acceptance testing by the sponsor and/or end users. The repackaging team needs to verify that the package performs as expected according to the report created in Step 1 with the project's sponsor; the uninstall functions

correctly, and there are no conflicts between the package and other software applications installed on the end users' PCs. The focus of user-acceptance testing is to confirm that the application conforms to users' expectations and that it functions as expected. This means that users can execute basic tasks such as opening the application and using common features.

Quality Assurance Testing

It is highly recommended that development of the package be separate from testing the package. This means that package developers should not test packages they have created. Rather, dedicated testing personnel can often do a better job of verifying a package's efficacy than the creator of the package.

This phase in testing should be conducted on the clean machine so the package can be tested in the lowest level base environment. The tester installs the package on the clean machine and then begins the testing process. Large organizations may wish to use automated testing tools to formalize and automate the testing process.

Package testing answers the following questions:

- Does the repackaged installation function as expected? Refer to the report created in Step 1.
- Does the application work in the expected manner? Again, refer to the report.
- Does the repackaged installation follow company standards?
- Are ODBC data sources created during the installation set up and functioning correctly?
- Are services created during the installation set up and functioning correctly?
- Does the installation comply with Microsoft application guidelines? This can be done easily in Package Studio by running the Validation Wizard, which checks that the installation follows Microsoft's standards for .MSI installations.

Testing the Uninstall

The next step is to verify that the repackaged installation's uninstall removes all items cleanly. An easy method for doing this is to capture the installation's uninstall and examine the resulting package – if everything was removed, then the uninstall package should be empty. In this section, the repackaged installation is referred to as the “original package” and the package created for the uninstall is referred to as the “uninstall package”.

Use this general procedure for testing the uninstall:

1. On a clean machine, start the packaging tool of choice. Snapshot 1 is created.
2. Install the original package created earlier. This installs all items in the original installation on the PC.
3. Run the application installed in step 2. This will capture any license files, INI file changes, or registry entry modifications that the application executes upon startup.
4. Run the original package's uninstall program to clean up everything installed by the original package.
5. Restart the packaging tool to create snapshot 2. The differences between snapshots 1 and 2 result in the uninstall package.
6. Exit the packaging tool.

Examine the uninstall package. If the uninstall removed all items that were installed by the original package, then the uninstall package should be empty. If it is not, then research the items that were in the uninstall package – perhaps there are items that should not be removed by the uninstall program and should be left in the original package. Or there may be items that should be removed by the uninstall. In that case, use a tool like Package Studio's Windows Installer Editor to remove them from the package. Consider adding these items to the exclusion list for future packages so they will be excluded permanently.

Identifying Potential Security Issues

Testing potential security issues within a package involves installing the package as a user with restricted privileges. Often security issues occur because a vendor package needs elevated, or administrator, rights in order to install correctly. If a user with restricted rights installs the repackaged installation, then it will not install successfully.

To test for this scenario, log on to a clean machine with a security profile that has access privileges similar those of end users in the organization. Install the repackaged installation. Test the application. Does it open correctly and function as expected? If not, restore the clean machine's image and install the repackaged installation as a user with an administrator profile. Again, test the installed application. If it functions successfully now, then the issue is related to the vendor's installation requiring elevated privileges to install correctly.

If the vendor installation does require elevated privileges to install successfully, review the company's policy on how to solve this issue. Is it permissible to give the end user elevated privileges for this repackaged installation? If not, contact the vendor to inquire if there are other options for installing the application.

Managing Conflicts

Up to this point, testing of the package has focused on bugs or unexpected behavior found in the package itself. Another important component of thoroughly testing a package, however, is testing interactions between it and other applications installed on end users' PCs. The consequences of distributing a package that conflicts with other applications can be disastrous. For instance, if the redistributed package installs a newer version of a .DLL file that is already in use by another application, the already-installed application may no longer function. If the non-functioning application is critical to the organization, then the support department has to clean up the package that was deployed to users' machines plus the damaged application that was already installed.

It is possible to manually test interactions between packages in a very controlled quality assurance environment, but it is quite time-consuming and tedious. To manually test for conflicts between applications, requires installing applications one at a time on a dedicated testing machine and running a battery of tests in order to determine if there are negative interactions. If a conflict is found, then the source of the conflict needs to be discovered, which is also a time-consuming task.

A much quicker and more reliable method is to automate the conflict checking process by using a tool specifically designed to check for conflicts between applications. Package Studio's ConflictManager is a database tool that helps solve the problem of conflicting files and registry entries that often occur on PCs with multiple applications installed. ConflictManager detects and resolves potential conflicts to reduce the risk of problems when a new application is deployed.

Testing the Package on the Base Build Machine

Testing the package on a base build machine involves giving the package one last trial before distributing it to users for acceptance testing. As described earlier, the base build machine should contain all software that is normally on a PC when it is given to a new employee. Testing on the base build machine allows the repackager to find additional conflicts that might occur between the package and applications commonly installed on the organization's PCs.

User-Acceptance Testing

User-acceptance testing occurs after successful quality assurance and conflict management testing. The sponsor should organize user acceptance testing and be responsible for signing off that the package meets user requirements. Users should install the package on their machines and use the application to be certain it functions in the desired manner.

Step 6:**Release the Package to End Users**

Once the package is tested, it is ready for distribution. Distribution can happen in a number of ways. The installation can be copied to a CD-ROM, network share point, or intranet site and users could install the package from that location. Or a distribution system, such as Microsoft SMS, Tivoli, CA Unicenter, Novadigm's Radia, Marimba, and others can distribute the package.

Depending on the number of end-users who will receive the package, a pilot test may be warranted before it is rolled out to the entire organization. This can be accomplished by rolling out the package in stages, in case there is a problem and the package needs to be recalled. Distribution of the package can occur in waves: distribute to 10% of users first, a few days later distribute the package to 20% of the remaining users and so on.

Conclusion

The goal of repackaging is to create reliable packages that produce consistent results. A process-oriented approach to repackaging helps ensure that dependable packages are produced consistently --- otherwise, the process becomes hit-or-miss and repeatability is reduced. A reduction in repeatability means more support calls for the corporate Help Desk and less confidence in the work of the repackaging team.

Wise Solutions' understands that producing reliable packages is key to the success of software rollouts for system administrators and IT managers. The repackaging best practices detailed in this white paper help administrators define and develop a process for repackaging, which can be customized to fit the specific needs of their organization.

Appendix A: Glossary of Common Repackaging Terms

Advertise – Windows Installer can advertise the availability of an application to users and to other applications without actually installing the application. If an application is advertised, only the interfaces required for loading and launching the application are presented to the user or other applications. Users can install the application by activating the advertised interface, Windows Installer then loads the necessary components. This functionality can save time and disk space.

Base build machine – A base build machine is used for testing packages. The base build contains the software commonly found in an organization, this includes all the software installed on the clean machine plus the minimum software that would be installed on a machine before giving it to a new user.

Capturing an application – The process of determining the changes made by an installation for the purposes of creating a new, customized installation.

Clean machine – a PC with only the minimum software installed as determined by a company's standards. It is important to use a clean machine when creating a package as it minimizes the possibility of files and registry keys not being captured correctly.

Conflict management – Conflict management is the process of analyzing packages for potential conflicts before they are distributed to end-users. Proactively identifying conflicts such as DLL conflicts or registry key conflicts can lessen the chances of negative consequences when the package is installed on an end-user's desktop.

Delta – Repackaging software captures or records the changes made to a PC's hard drive between the first and second scans. The difference between the scans is a "delta," which is used to create the new package.

Distribution systems – A distribution system is an automated method for delivering packages to end-users' desktops. Often large organizations use distribution systems to "push" out packages to end users' desktops but the technical method of how packages are delivered can vary by the vendor and the product. Examples of vendors and the distributions systems they create and sell include Microsoft SMS, Tivoli, CA Unicenter, Novadigm's Radia, Marimba, and others.

Exclusion list – An exclusion list is a list of files, folders, registry keys, and registry values that should not be included in the captured installation. Exclusion lists are valuable because they help create clean installations by eliminating files, folders, registry keys, and registry values that are not related to the actual installation.

.MSI – A distributable Windows Installer installation package file. The .MSI file extension is associated with the Windows Installer executable, MSIEExec.EXE. When a file with this extension is opened on a destination computer, the Windows Installer software executes it, thereby installing an application. You can open and edit .MSI files in Windows Installer Editor.

Package – The new installation that is created using the process of repackaging. An installation is designed to support company standards and distribution methods.

Repackaging – Repackaging is the process of capturing the and customizing it for the purpose of creating a new installation that is designed to support company standards and distribution methods.

SetupCapture – Wise Package Studio's technology that creates snapshots of a hard drive before and after the installation of an application, then uses the delta to construct a package. SetupCapture includes many options to shorten the capturing process, including functionality for creating automatic exclusion lists, reporting capabilities, and smart monitoring technology.

Snapshot – A system scan that records all of the resources (files, registry keys, etc.) installed on the hard drive, before or after the installation of an application.

Sponsor –The individual who makes the initial request to repackage a software application. The sponsor is often a department supervisor who is familiar with how the end users, who will use the final package, interact with the application.

Transform – Transforms alter the installation database and can be used to encapsulate the various customizations of a base package required by different groups of users. For example, in organizations where the finance and staff support departments require different installations of a product, the product's base package can be made available to everyone at one administrative installation point with the appropriate customizations distributed to each group of users separately as transforms. (excerpted from the Microsoft Windows Installer SDK)

Windows Installer – Microsoft created the Windows Installer service to create a streamlined process for installing and managing applications. It consists of a set of guidelines, an Application Programming Interface, and a runtime service to help make application installation and ongoing management part of the basic Windows system services. The Windows Installer service is not an installation-authoring tool, but rather an installation engine and rule set for installation packages.

Appendix B: About Wise Solutions

Company Overview

Wise Solutions is a leading global provider of software repackaging solutions, installation tools and services for Windows-based applications. We offer powerful, reliable installation products specifically designed for software application developers, web managers, software vendors, network administrators and system administrators.

Our philosophy of achieving nothing less than technical excellence has helped shape Wise into a strong, rapidly growing company with a bright and highly motivated workforce. Our success is the result of the close working relationships with our customers to develop superior installation tools and services.

Our dedication to customer relationships and passion for technology has allowed Wise to receive many coveted industry awards and recognition, including the prestigious Visual Basic Programmer's Journal Readers' Choice Award four years in a row.

Thousands of companies and developers have switched to Wise Solutions and our award-winning installation tools because they expect unparalleled functionality, ease of use, exceptional reliability . . . and Wise delivers.

Solutions for System Administrators

Wise Solutions offers advanced software packaging and installation solutions with a full range of capabilities and features for system administrators and software developers. Whether you need a simple setup tool or a complex, enterprise solution for software packaging, Wise Solutions can provide the perfect combination of power, ease of use, and versatility.

Wise Package Studio

Wise Package Studio is designed to help system administrators and desktop integration labs streamline the process of packaging software applications for delivery to end-users. Applications packaged with Wise Package Studio can be deployed by all leading software distribution systems. Wise Package Studio is available in three editions: Enterprise, Professional and Standard.



www.wise.com

Phone: (734) 456-2100 • Orders: (800) 554-8565 • Fax: (734) 456-2456