



realtimepublishers.com[™]

The Definitive Guide[™] To

Windows Installer
Technology
for System Administrators



Darwin Sanoy and Jeremy Moskowitz

Chapter 2: MSI Tools Roundup.....	21
by Jeremy Moskowitz.....	21
Basics of the Repackaging Approach.....	22
Microsoft’s Offerings.....	22
WinInstall LE.....	23
WinInstall LE Operation.....	23
SMS Installer.....	25
The SMS Installer Repackage Installation Wizard Tool.....	26
The SMS Installer Watch Tool.....	27
The SMS Installer Script Editor.....	28
Creating MSI Files with the SMS Installer.....	30
Commercial Third-Party MSI Tools.....	33
Commercial Third-Party Tools at a Glance.....	33
Wise Package Studio.....	34
AdminStudio 3.5.....	38
Prism Pack.....	40
Added Functionality.....	42
Wise Package Studio 4.0 Repackaging Innovations.....	42
AdminStudio 3.5 Repackaging Innovations.....	43
Shareware and Freeware Third-Party Tools.....	43
Summary.....	44

Copyright Statement

© 2002 Realtimepublishers.com, Inc. All rights reserved. This site contains materials that have been created, developed, or commissioned by, and published with the permission of, Realtimepublishers.com, Inc. (the “Materials”) and this site and any such Materials are protected by international copyright and trademark laws.

THE MATERIALS ARE PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. The Materials are subject to change without notice and do not represent a commitment on the part of Realtimepublishers.com, Inc or its web site sponsors. In no event shall Realtimepublishers.com, Inc. or its web site sponsors be held liable for technical or editorial errors or omissions contained in the Materials, including without limitation, for any direct, indirect, incidental, special, exemplary or consequential damages whatsoever resulting from the use of any information contained in the Materials.

The Materials (including but not limited to the text, images, audio, and/or video) may not be copied, reproduced, republished, uploaded, posted, transmitted, or distributed in any way, in whole or in part, except that one copy may be downloaded for your personal, non-commercial use on a single computer. In connection with such use, you may not modify or obscure any copyright or other proprietary notice.

The Materials may contain trademarks, services marks and logos that are the property of third parties. You are not permitted to use these trademarks, services marks or logos without prior written consent of such third parties.

If you have any questions about these terms, or if you would like information about licensing materials from Realtimepublishers.com, please contact us via e-mail at info@realtimepublishers.com.

Chapter 2: MSI Tools Roundup

by Jeremy Moskowitz

In the last chapter, you were introduced to the current crop of installation headaches and how the Windows Installer technology in conjunction with MSI files helps get you closer to the way you should be installing your software. Remember, Windows Installer allows you several tangible benefits that you can't get via the widespread setup.exe methods. A quick review of the major highlights follows:

- Transactional install and rollback—If the package fails to install midway, the package automatically removes itself as if the installation was never attempted.
- Self-healing (or self-repair) of corrupt or deleted critical files—If the installation is damaged, a file (or many files) can be grabbed from the installation source to fix the application.
- Just-in-time (JIT) installation—A package need not be *fully* installed right away. If a feature or component of the package is needed later, it can be grabbed from the source and loaded JIT—all while the application is already started.

These benefits all sound terrific, and they are. But if your application isn't packaged as an MSI file already, what are you going to do? Quite simply, you'll need to get your applications to the MSI "promised land." And you'll do that with one or more of the myriad tools available. You'll use these MSI creation tools to manage the applications you buy or develop in-house and repackage them into new MSI packages. This process is usually as simple as rounding up the existing setup.exe-type applications and repackaging them into the Windows Installer and MSI format.

Some tools are free and some cost a bit, and they all have different angles and philosophies for repackaging. That's what this chapter is about—to expose you to the different options you have for authoring or repackaging your application. You'll get glimpses into how to use each tool; however, this chapter isn't meant to discuss every nook, cranny, and feature that any specific application provides. In addition, this chapter isn't meant to guide you to a specific tool, and it doesn't include specific recommendations for a tool that you should use.

Each of these tools will lead you to the end goal, which is to repackage your application into the MSI format. Indeed, each tool will do the job. I will be minimizing the step-by-step instructions, and, instead, try to emphasize each application's overall methodology differences. It is also my goal to briefly expose you to a little bit of each application's interface so that you can be somewhat comfortable and familiar with each one should you encounter it on your own. By being exposed to many tools, you can make an informed purchasing decision as well as be a little more comfortable with using the tool upon first use. You will use this information about the tool or tools you choose in the next several chapters, especially Chapter 4 about the best practices for building packages.

Basics of the Repackaging Approach

Many of the tools that we'll discuss share a common approach. Specifically, these tools perform a *snapshot*; they take a before and after picture and generate a package based on the *delta* (the differences between the two pictures). Although these steps represent the basic approach of the majority of the packages we'll discuss, each tool will bring something extra to the table, to enhance the experience. A snapshot occurs as follows:

- Start the tool to perform the snapshot.
- Have the tool examine the contents of the hard drive to see the current state.
- Run the Setup.exe program for the application that you want to repackage.
- Set the desired installation options.
- Customize the installation as desired.
- Have the tool re-examine the contents of the hard drive to see the changes to the original state.
- The tool creates the repackaged application.

Regardless of which tool you're working with, you should keep one main principal in mind: ensure that the machine from which you're creating the snapshot is as *clean* as possible. A clean machine is one with the OS installed and almost nothing else. That way, when the tool performs the snapshot, it has only the most minimal interference. Even seemingly innocuous applications, such as screensavers, could impact what is seen in the before and after scans of the hard drive.

However, if you cannot use a totally clean machine (for instance, the application you want to repackage has dependencies on another preloaded application), you can use the following tips to improve your chances at a successful repackage:

- Close all applications
- Stop all unnecessary services
- Clear out the Recycle Bin
- Disable screensavers
- Disable antivirus programs
- Disable anything else that runs in the background

Microsoft's Offerings

Because Microsoft is the biggest proponent of the Windows Installer technology, it would stand to reason that the company would have several tools to assist in MSI file creation. After all, MSI does stand for Microsoft Installer. With that in mind, we're going to explore three tools that Microsoft provides for administrators to repackage an application into the MSI format.

WinInstall LE

WinInstall LE is a free tool provided by Microsoft to help with creating MSI files. WinInstall LE wasn't designed by Microsoft, rather, it has a long and strange history. WinInstall LE was developed by OnDemand Software. Seagate bought WinInstall LE, then Veritas bought Seagate. WinInstall LE didn't get much attention from Veritas, as the company's focus is mainly backup software. OnDemand Software reformed and took the product back into development.

It's easy to get confused by the naming convention that OnDemand Software has chosen. WinInstall LE is the tool we're discussing that you can use to repackage setup.exe programs into MSI files. The primary job of WinInstall (without the LE) is to deploy and install applications throughout your environment. However, the full package of WinInstall does contain a standalone repackaging application that is basically a revised cousin of WinInstall LE.

WinInstall LE Operation

To get started on your WinInstall LE journey, you'll need to grab hold of either a Win2K Professional or Win2K Server (or Advanced Server) CD-ROM. Once you have it, locate SWIADM.LE, which can be found in the {cdrom}:\VALUEADD\3RDPARTY\MGMT\WINSTLE directory, as Figure 2.1 shows.

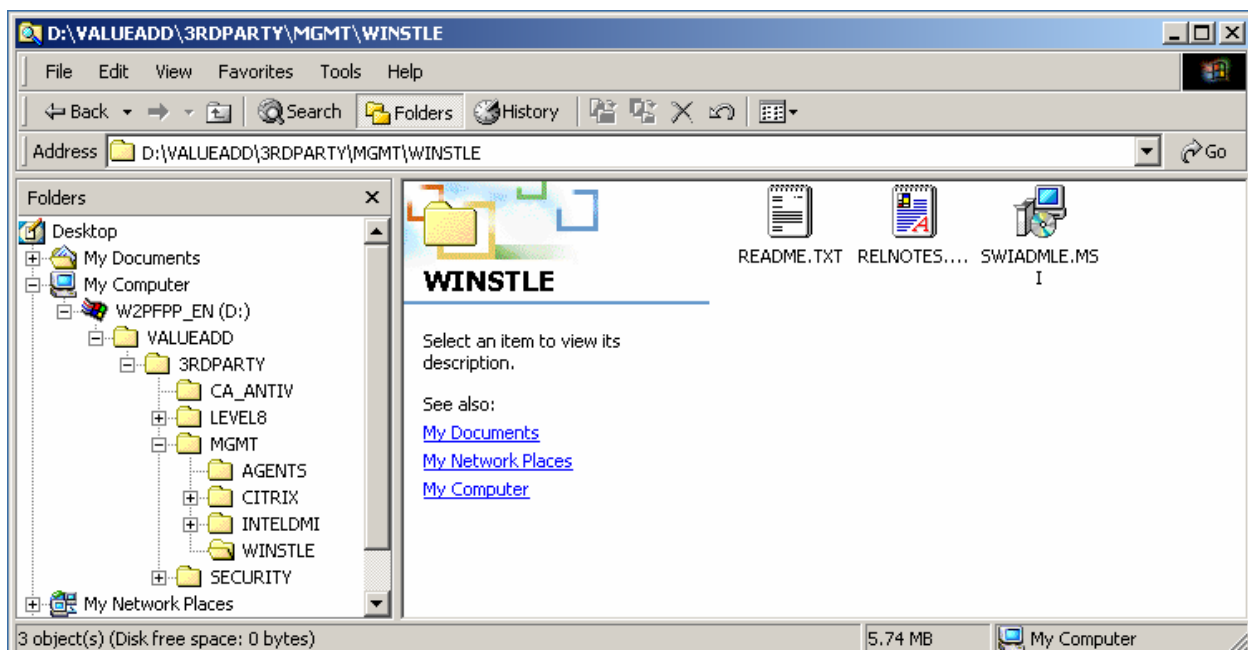


Figure 2.1: Traverse to the appropriate directory to reach the SWIADM.LE file.

The snapshot utility of WinInstall LE is called Discover. After running Discover, you'll title your application and give the tool a location at which to store the application, as Figure 2.2 shows.

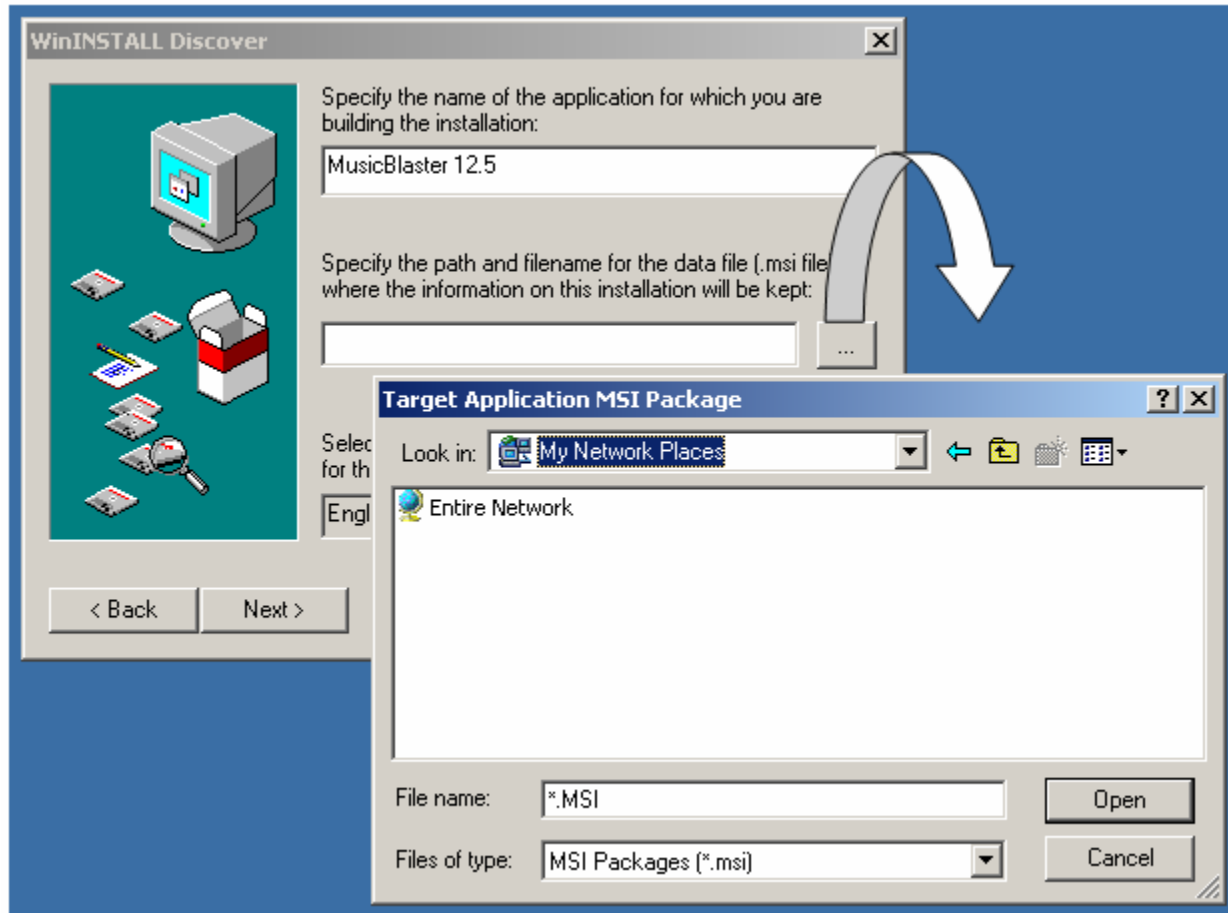


Figure 2.2: Put the MSI file you are creating on a network point.

Then, you'll simply run the setup.exe program from your application and complete the snapshot; out will pop your first MSI file.

After you have your MSI file, you could, if you were so inclined, venture into the murky world of WinInstall LE's package-editing tool. At this point, WinInstall LE's usefulness starts to break down. The WinInstall LE editing tool is called the Veritas Software Console, and can be launched via Start menu, Program Files, Veritas Software, Veritas Software Console. With this program, you can load an MSI package into the Veritas Software Console to manipulate your package, as Figure 2.3 shows.

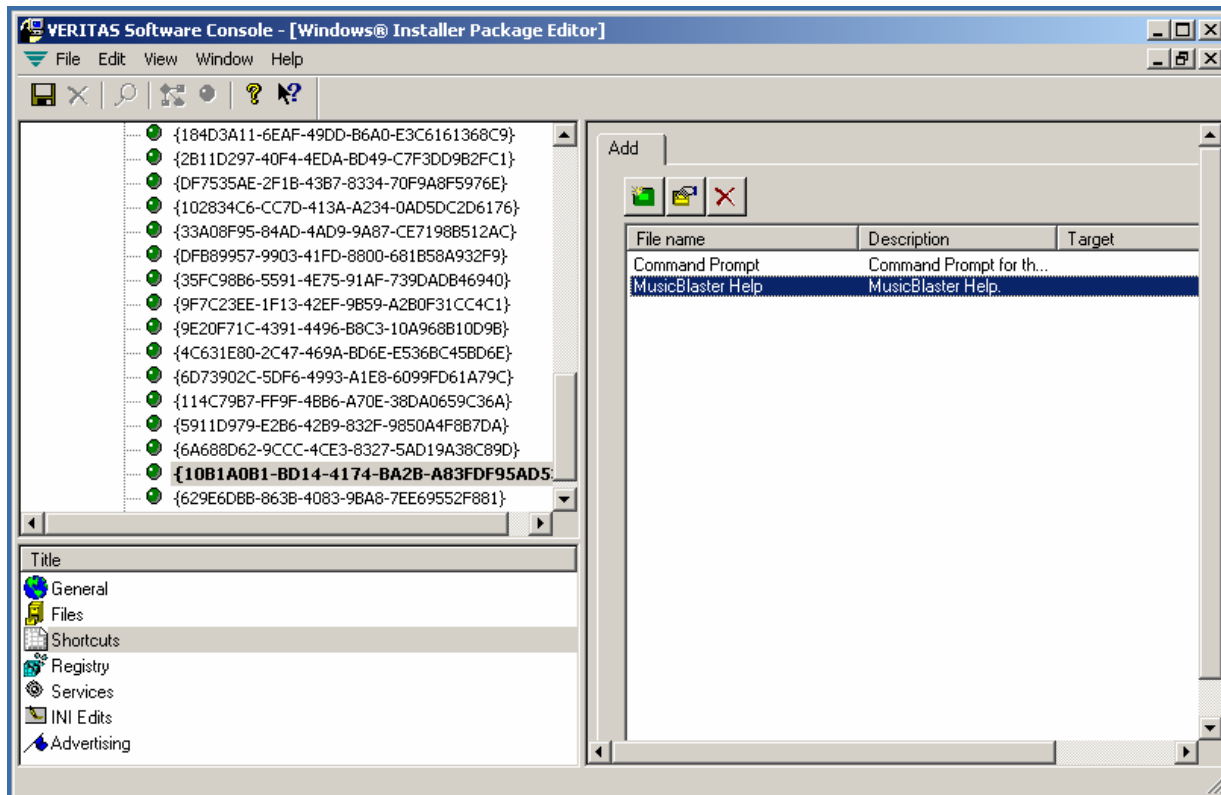


Figure 2.3: The post-editing tool is complex to navigate and negotiate.

Trying to customize a package using this tool is difficult at best. It's a fairly cumbersome, abstract, and difficult-to-use interface and requires an intimate knowledge of the relationships between features, components, and the like.


SMS Installer

The SMS Installer utility is another tool available from Microsoft for those who own and operate SMS environments. The SMS Installer utility was originally a Wise product. Wise licensed Microsoft a chunk of source code that became the SMS Installer, and Wise continued on developing the Wise Installer, and later, Wise Package Studio (which we'll explore later).

The SMS Installer is a poorly named product, but has some powerful features. It's poorly named because the name suggests that this program offers only a specific installation method using SMS, which doesn't incorporate all that the program does. The SMS Installer provides three ways to capture events and eventually create MSI files:

- Straight snapshot (similar to WinInstall LE)
- Via the SMS Installer Watch utility
- Through a custom/manual script editor

We'll be briefly exploring each of these three methods.

 You'll only be able to download and unpack the setup routine for the SMS Installer if you've actually got an SMS 2.0 site server up and running. This requirement is simply a protection mechanism to ensure that only people who have SMS licensed and up and running are able to use the SMS Installer tool. You can find the SMS Installer tool at <http://www.microsoft.com/smsserver/downloads/20/tools/installer.asp>.

MSI creation is a fairly new feature in the SMS Installer feature set. Historically, the SMS Installer didn't create MSI files; rather, its only output was in the form of self-installing executables (.EXE files.) The latest revisions of the SMS Installer can create MSI files—via both the SMS Installer tool and an external program called the Installer Step-Up utility (ISU).

The SMS Installer Repackage Installation Wizard Tool

After you've installed the SMS Installer and launched it for the first time, you'll see plenty of options to get started, as Figure 2.4 shows.

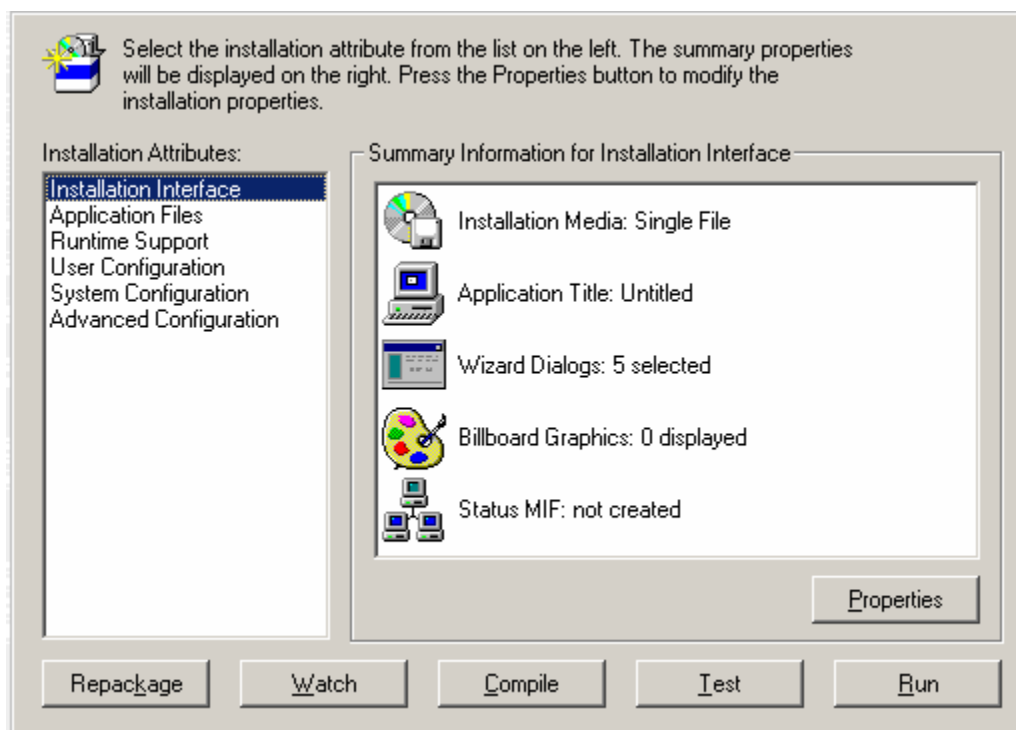


Figure 2.4: The SMS Installer has a lot of nooks and crannies.

I suggest that you start by exploring the Repackage option, which, as you might expect, repackages applications. After you click Repackage, you'll be prompted for the location of the setup.exe file as well as which portions of the hard drive to scan. After you provide this information, the first scan is performed, as Figure 2.5 shows.

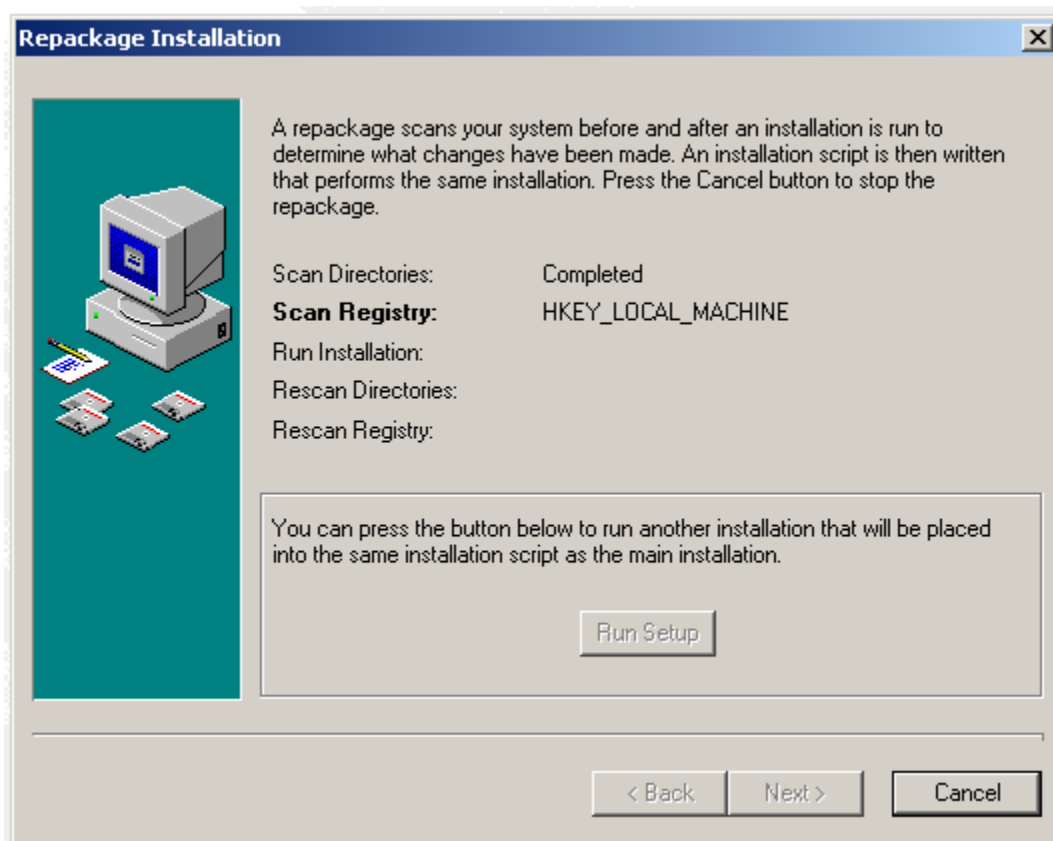



Figure 2.5: The repackager option starts off in a manner similar to how WinInstall LE starts off.

 You can have the tool capture multiple installations at the same time by clicking Run Setup when it becomes available (after the scan completes). However, it's best to repackager one application at a time as doing so lets you create individual MSI applications, as opposed to one MSI file with lots of applications inside.

After the scan completes, the setup.exe program that you selected will be launched. Complete the program's installation, and finish the Repackage Installation Wizard. At this point, you'll have captured the setup.exe program's actions but you're not quite to an MSI file. In a moment, you'll see two ways to make that capture into an MSI file.

The SMS Installer Watch Tool

In those rare circumstances in which your application is already installed (and you cannot find the setup.exe program) or the application doesn't come with a setup.exe file, you can use the SMS Installer Watch tool, as Figure 2.6 shows.

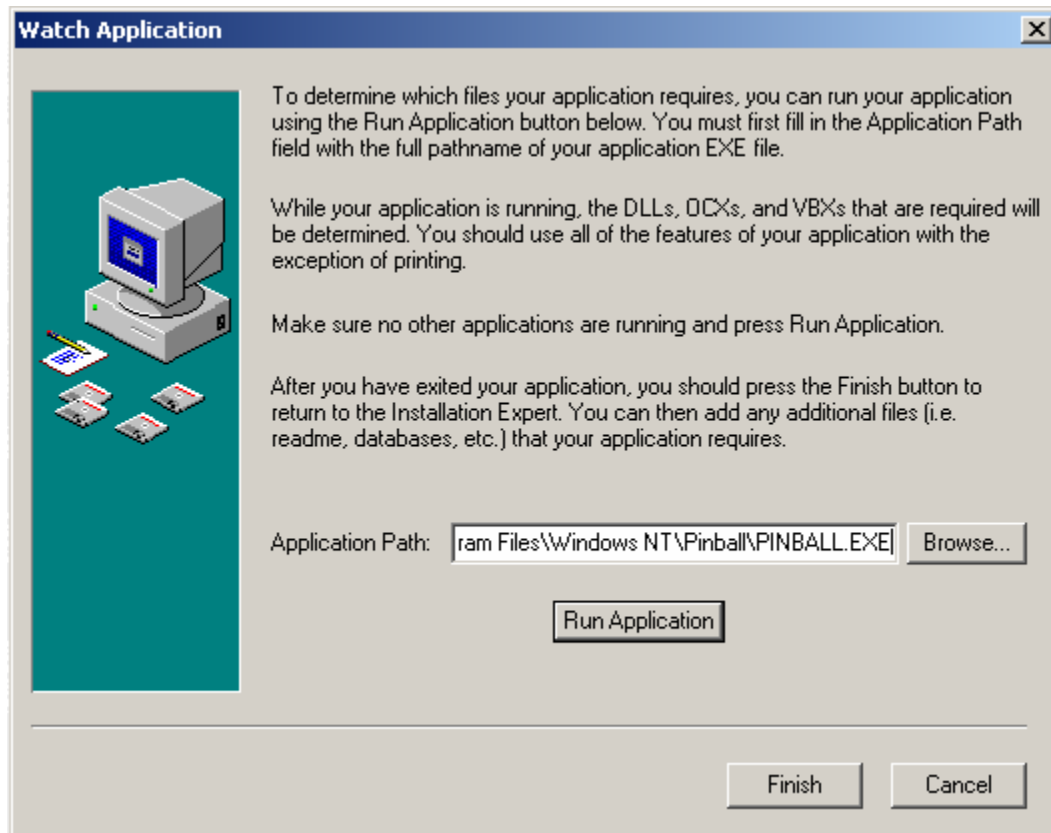


Figure 2.6: Use the Watch application when you don't have a setup.exe program for your application.

After running the application, SMS Installer looks for every call the OS makes to see which files are being accessed. When this process completes, you'll be able to perform a capture using the method I previously discussed.

This option is worthwhile in a pinch, but is ultimately not going to be as accurate as a true snapshot or capture. The reason is that when you use the Watch method, you'll have to be sure that you're selecting *every* option that the application provides. If the application is a word processor, you'll have to make sure you run the spell checker, mail merge, and Help features—everything that could possibly be selected—to ensure that every DLL, OCX, and VBX file are all “seen” by SMS Installer.

The SMS Installer Script Editor

The SMS Installer Script Editor is the most powerful feature of the SMS Installer. After you've either performed a recapture or used the Watch application, you have the option to make modifications to *how* the package is going to be installed and which options will be seen during the actual installation process. You perform these changes inside the Script Editor. What if you wanted to make sure your resulting repackaged only ran on Win2K? What about making specific changes to certain INI files? You can do these things and more with the Script Editor, which Figure 2.7 shows.

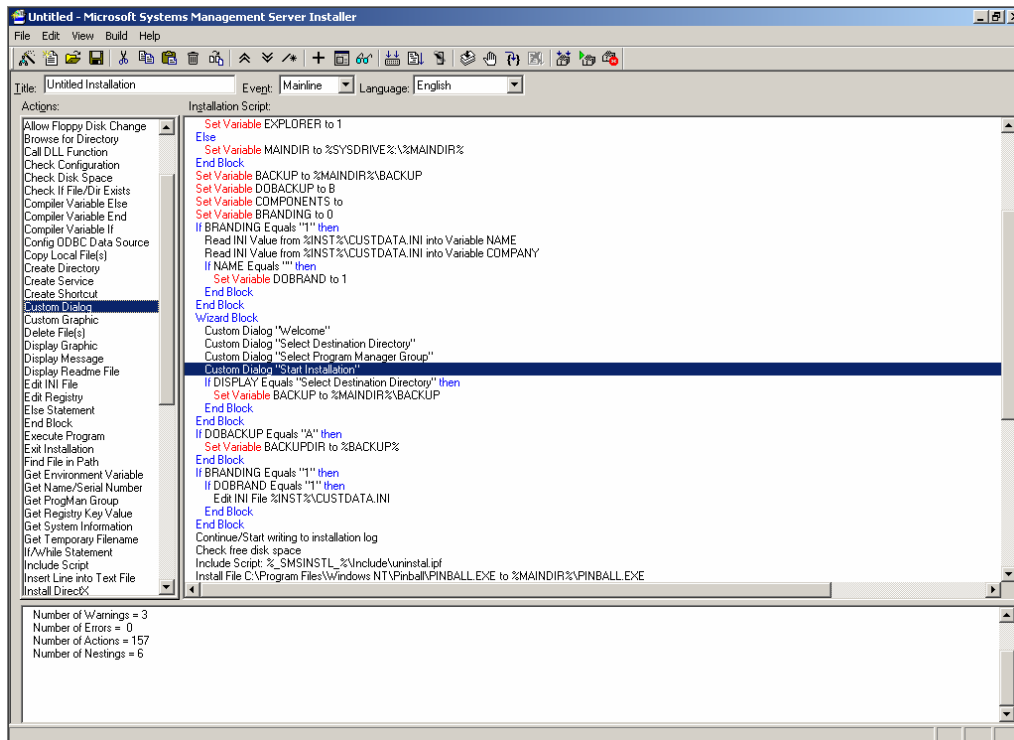


Figure 2.7: The SMS Installer Script Editor has ton of options for configuring installation options.

You simply drag Actions from the left to a point in the script on the right. After the action is where you want it in the script, you can modify the action. For instance, if you wanted to modify the standard setup screens by adding some custom information, you can edit the dialog box within the Script Editor, as Figure 2.8 illustrates.

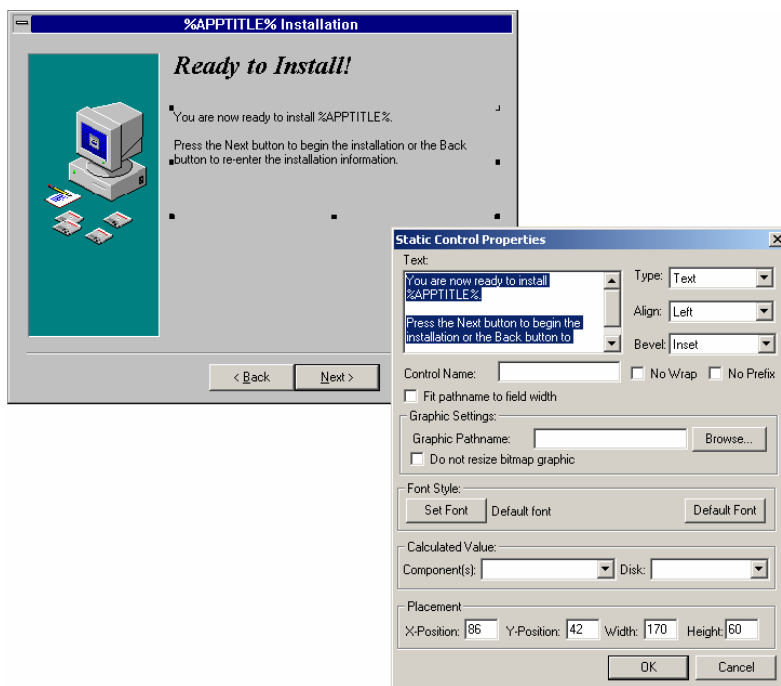



Figure 2.8: You can customize each installation screen using the Script Editor.

The SMS Installer Script Editor is an immensely powerful tool and has simply too many options to discuss in detail here. However, there are many resources that provide step-by-step directions for its use; I recommend Rod Trent's *Microsoft SMS Installer* (McGraw-Hill Osborne Media). Rod also has an excellent Web site that is full of both SMS and SMS Installer information <http://myitforum.com>.

 If you make customized script additions (as discussed previously), then choose to make MSI files from them (as we're about to explore), you'll need to thoroughly test your resulting MSI files. Sometimes customized scripting made inside the SMS Installer will not be perfectly reflected in a resulting MSI file (but will be perfectly reflected in a compiled .EXE file).

Creating MSI Files with the SMS Installer

Your last step is to actually create an MSI file. You have two options to do so: by using the built-in MSI Compilation Method or the external ISU utility.

After the capture or Watch-application process is completed (as well as the optional script editing), you can simply choose the Build menu option to *Compile As Windows Installer Package*, as Figure 2.9 shows.

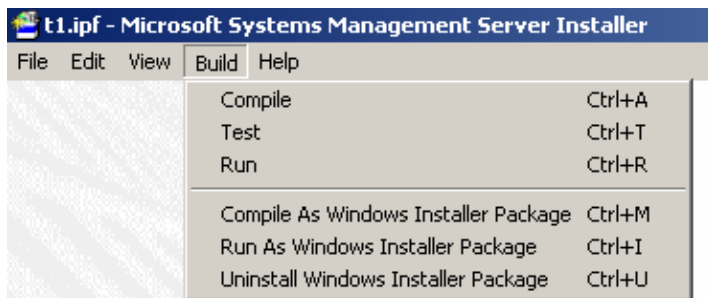


Figure 2.9: You can create MSI packages directly from SMS Installer.

When you do, you'll see the MSI file being created, as Figure 2.10 shows.

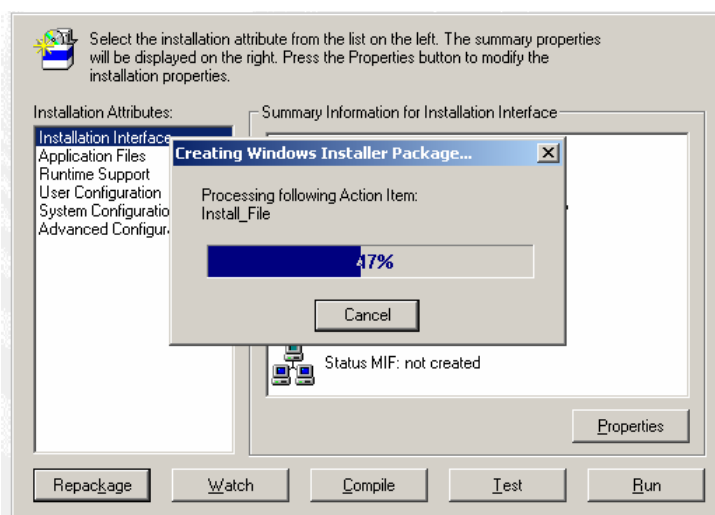


Figure 2.10: The MSI file is created from within the SMS Installer.

Alternatively, you could use the command-line ISU utility, which converts SMS Installer self-executables to MSI files. Figure 2.11 shows the ISU utility's options.

```

C:\Output>u:isu
Installer Step-up Utility for Microsoft (R) Systems Management Server
Version 1.0.448.0
Copyright (C) Microsoft Corporation 2001. All rights reserved.

Syntax: isu <file> [/xf dir] [/co dir] /j /s /e /c /v /t /langid:xxx
       isu [/? | /help]

Where
file           Specifies the file to migrate.
/j             Adds support for Windows Installer install on demand.
/s            Migrates files in current or specified directory and
             subdirectories.
/xf dir        Places the extracted files in the specified directory.
/co dir        Places migrated files (.msi) in the specified directory.
/e            Extracts only.
/c            Converts the script and files only.
/langid:xxx    Sets the codepage to use during the conversion process.
/v            Sets output to verbose.
/t            Tests to make sure an EXE file is an SMS Installer-generated
             setup package.

/?            Displays command-line help.
/help         Launches the ISU help file, which includes full usage
             instructions.

C:\Output>u:isu pinball.exe
Extracting C:\Output\pinball.EXE.

Converting C:\Output\%pinball%\pinball.ipf to C:\Output\pinball.ipf.

Converting action number: 2; In package number: 1; Open_Close_INSTALL_LOG
ISU Warning 3001: The following SMS Installer script action is not supported: Op
en/Close INSTALL.LOG
Converting action number: 47; In package number: 1; Open_Close_INSTALL_LOG
ISU Warning 3001: The following SMS Installer script action is not supported: Op
en/Close INSTALL.LOG
Converting action number: 60; In package number: 4; Add_ProgMan_Icon
ISU Warning 3024: Windows Installer does not support variables with filenames. T
he following cannot be converted: %APPTITLE%

Summary:
  Number of Warnings = 3
  Number of Errors = 0
  Number of Actions = 89
  Number of Nestings = 6
Finished.

C:\Output>

```

Figure 2.11: The ISU utility options as well as a compile.

The ISU utility is useful when you have existing SMS Installer scripts that you want to batch-style convert to MSI files. However, as Figure 2.11 illustrates, not every scriptable option converts to a perfect MSI action. When the SMS Installer converts your packages to MSI, it needs to make decisions about what your intentions are, and thus might perform an action on your behalf that you didn't intend. In other words, you might script an action, but that action doesn't work or work properly when run as an MSI. This behavior is the SMS Installer's biggest shortcoming. That is, although it's a powerful tool that creates useful self-executable .EXE installation files, it doesn't always produce the best-running MSI files. In such cases, you might need to get into the nitty-gritty of the MSI file. To do so, you'll need the Microsoft Orca tool, which the following sidebar describes.

Microsoft Orca Tool

As we've previously stated, each MSI file is really just a database. Microsoft provides Orca, a low-level tool, to let you get into an MSI file and look around. Orca is part of the Windows Installer SDK, which is most useful for programmers. But systems administrators can use the SDK—and Orca—to find the most nitty-gritty documentation for how something works within a Windows Installer executable or an MSI package.

The full SDK can be found at <http://www.microsoft.com/msdownload/platformsdk/sdkupdate/>. If you're not interested in the full SDK, you can get a subset of the SDK, which includes Orca at <http://msdn.microsoft.com/downloads/default.asp?URL=/code/sample.asp?url=/msdn-files/027/001/457/msdncompositedoc.xml>.

If you choose to load a file into the Orca tool, you'll be able to see the file's rawest elements: the tables and rows that make up the database, as Figure 2.12 shows.

Tables	File	Component	FileName	FileSize	Version	Language	Attributes	Sequence
Condition	iadstools.doc	CoReplmon	iadsto~1.doc iads...	170496			16384	406
Control	windiff.hlp	CoWindiff	windiff.hlp	17357			16384	116
ControlCondition	clonepr.doc	CoClonepr	clonepr.doc	97280			16384	302
ControlEvent	clonepr.dll	CoClonepr	clonepr.dll	108304	5.0.21...	1033	16384	252
CreateFolder	pviewer.exe	CoPviewer	pviewer.exe	25872	5.0.21...	1033	16384	104
CustomAction	netdiag.exe	CoNetdiag	netdiag.exe	376080	5.0.21...	1033	16384	34
Dialog	msicuu.exe	CoMsicuu	msicuu.exe	176128	1.0.0...	1033	16384	248
Directory	tlst.exe	CoTlist	tlst.exe	13584	5.0.21...	1033	16384	40
DrLocator	sidwalk.exe	CoSidwalk	sidwalk.exe	89360	5.0.21...	1033	16384	238
DuplicateFile	sidhist.vbs	CoClonepr	sidhist.vbs	4404			16384	262
Environment	showaccs.exe	CoSidwalk	showaccs.exe	102160	5.0.21...	1033	16384	236
Error	search.vbs	CoSearch	search.vbs	19909			16384	234
EventMapping	sdcheck.exe	CoSdcheck	sdcheck.exe	25360	5.0.21...	1033	16384	232
Extension	windiff.exe	CoWindiff	windiff.exe	93456	5.0.21...	1033	16384	112
Feature	w2rksupp.chm	CoW2rksupp...	w2rksupp.chm	405095			16384	318
FeatureComponents	ldp.exe	CoLdp	ldp.exe	330000	5.0.21...	1033	16384	30
File	rstools.dll	CoRstools	rstools.dll	1212176	5.0.21...	1033	16384	230
Font	rsdir.exe	CoRstools	rsdir.exe	16144	5.0.21...	1033	16384	228
Icon	rsdiag.exe	CoRstools	rsdiag.exe	15632	5.0.21...	1033	16384	226
IniFile	repadmin.exe	CoRepadmin	repadmin.exe	53520	5.0.21...	1033	16384	38
IniLocator	remote.exe	CoRemote	remote.exe	35088	5.0.21...	1033	16384	36
InstallExecuteSequence	gflags.exe	CoGflags	gflags.exe	23312	5.0.21...	1033	16384	22
InstallUISequence	reg.exe	CoReg	reg.exe	49424	2.0.0.0	1033	16384	224
LaunchCondition	dskprobe.exe	CoDskprobe	dskprobe.exe	98576	5.0.21...	1033	16384	208
	nsani.dll	CoGflags	nsani.dll	28944	5.0.21...	1033	16384	24

Figure 2.12: An MSI file loaded into the Orca tool.

The frame on the left has the heading of Tables; each table represents a portion of the MSI file. The larger pane on the right represents each row within the table, representing a record or an entry. In Figure 2.12, I've highlighted the table named File, and each row of the File table shows an entry for each file contained within the MSI.

This tool isn't the most user friendly tool for administrators, but with a little poking around you can discover some useful information. Specifically, whenever you use any tool to create and/or edit an MSI file, that tool might veil what's really going on in the file. With Orca, you can be confident that you're seeing what's really going on in the MSI file. However, the beauty of third-party tools is that they provide the information that you need in a user-friendly format—the raw elements that Orca reveals are quite messy. So use Orca only when necessary if you suspect troubles. You can find additional information about the Orca tool (from an administrator's perspective) at <http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q255905&>.


Commercial Third-Party MSI Tools

Microsoft isn't the only vendor that offers tools that can make MSI packages. Indeed, there are such tools available from third-party vendors. Like the SMS Installer, most third-party tools are dual-purpose; that is, they are capable of building scripts to get your home-grown package packaged and equally able to capture your application and wrap it up into an MSI package. For instance, you can use them to both repackage a setup.exe application into an MSI file and repackage the latest virus definition file components into an MSI that is prepared for deployment.

The feature set of third-party tools is usually quite robust. The best ones offer additional and innovative capture methods as well as advanced conflict management to assist in ensuring that in the case of DLLs that have the same name but are different versions, the best one is used for your MSI. Not every tool has these advanced features, but you might not need all these features. Check out all the tools to see which one fits your needs.


Commercial Third-Party Tools at a Glance

There are quite a few third-party tools that could fit the bill for your environment. To help you decide, Table 1.1 shows some of the most popular third-party MSI repackaging tools as well as a bit of information to help you get started on your third-party MSI tool investigation.

 The tools in this table are commercial programs. I highlight free tools in the next section.

Tool	Vendor	Web site
Wise Package Studio	Wise Solutions	http://www.wise.com
Wise for Windows Installer	Wise Solution	http://www.wise.com
AdminStudio	InstallShield	http://www.installshield.com
Prism Pack (formally Picture Taker)	Lanovation	http://www.lanovation.com
Unity Installer	PriceWaterhouseCoopers	http://www.unitysite.com/
E-Wrap	Novadigm (formerly ChicagoSoft)	http://www.novadigm.com

Table 2.1: Some popular third-party tools.

 There is an additional unofficial list on InstallSite.org, a site dedicated to MSI development (http://www.installsite.org/cgi-bin/frames.cgi?url=http%3A%2F%2Fwww.installsite.org%2Fpages%2Fen%2Fw2k_msiauth.htm). You can find another unofficial list at AppDeploy.com (<http://www.appdeploy.com/tools/browse.asp?r=1>). Note that InstallSite.org is geared for developers and AppDeploy.com is geared for administrators.

Highlighting every commercial third-party tool available is beyond the scope of this chapter. Instead, I'll review some of the top tools and draw attention to some of their distinguishing qualities.

Wise Package Studio

The Wise Package Studio is a popular tool for system administrators who want to repackage installations into MSI tools. In fact, if you're already comfortable with the SMS Installer, there are places in the Wise Package Studio that are similar to the SMS Installer, such as the Wise Script Editor.

After Wise licensed the code to Microsoft for the SMS Installer, Microsoft didn't develop the SMS Installer much beyond its original inception besides bug fixes, the ISU utility, and the in-program MSI builder capabilities. However, the development of that original code didn't stop for Wise. It became Wise for Windows Installer, and has since evolved to become Wise Package Studio.

As I previously mentioned, there are too many features offered by third-party tools to highlight them all (though I will highlight some specific competitive distinguishing qualities in an upcoming section). Instead, I'll highlight where the Wise Package Studio can help administrators most: in gaining a process around their MSI development process.

The Wise Package Studio wraps its package creation procedure into an approach that contains two main parts: a *process* and a *project*. That is, you start off with a *process*—either a pre-defined process or one that you define. Then, for each MSI package that you want to build, you leverage the process and create a new *project*, say, MSIFILE1.MSI. In other words, perhaps not every MSI file you'll build will require the exact same set of rules to create it, and the Wise Package Studio is flexible in this manner to help you ensure you have a well-defined *process* for your *project*. In the following example, I have modified the standard process to add a reminder for myself to ensure that I won't forget to virus scan before repackaging the application, as Figure 2.13 shows.

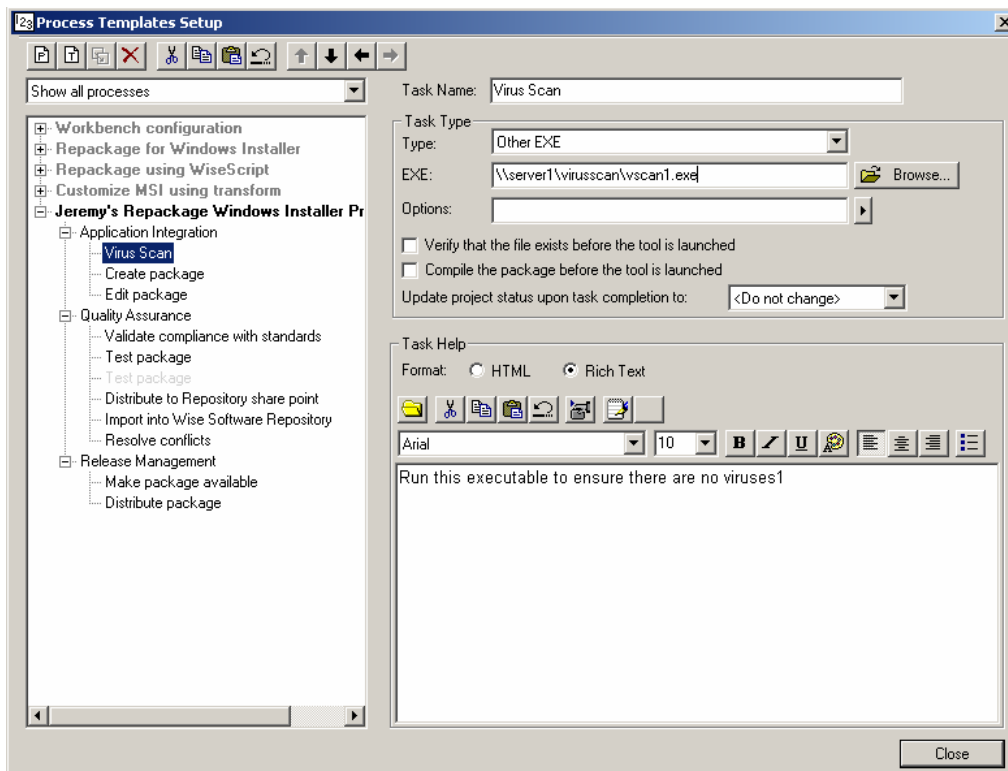


Figure 2.13: You can modify the process with specific steps that you can customize.

Then, when I'm ready to create a new project, I can leverage my own customized process (the one that includes a step ensuring that I won't forget to virus scan). I simply create a new project, and pick the process that I want, as Figure 2.14 shows.

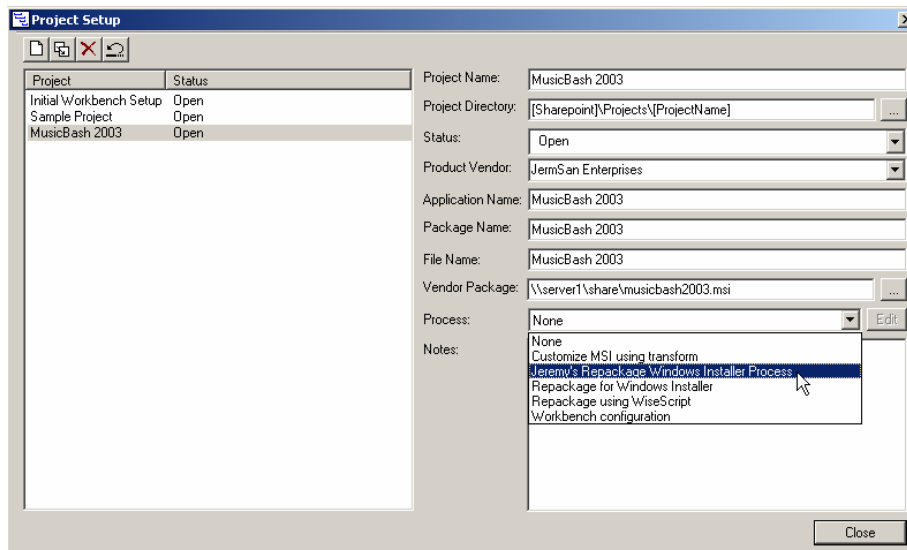


Figure 2.14: Select a process for each project that you want to repackage.

After you've established a process for your project, you're ready to work inside that project. Again, the process is what keeps you on track when wrapping up the MSI. The interface of the Wise Package Studio helps keep you on that track once a project is initiated. For example, the program offers a feature called the Workbench, which Figure 2.15 shows, that helps with the repackaging process.

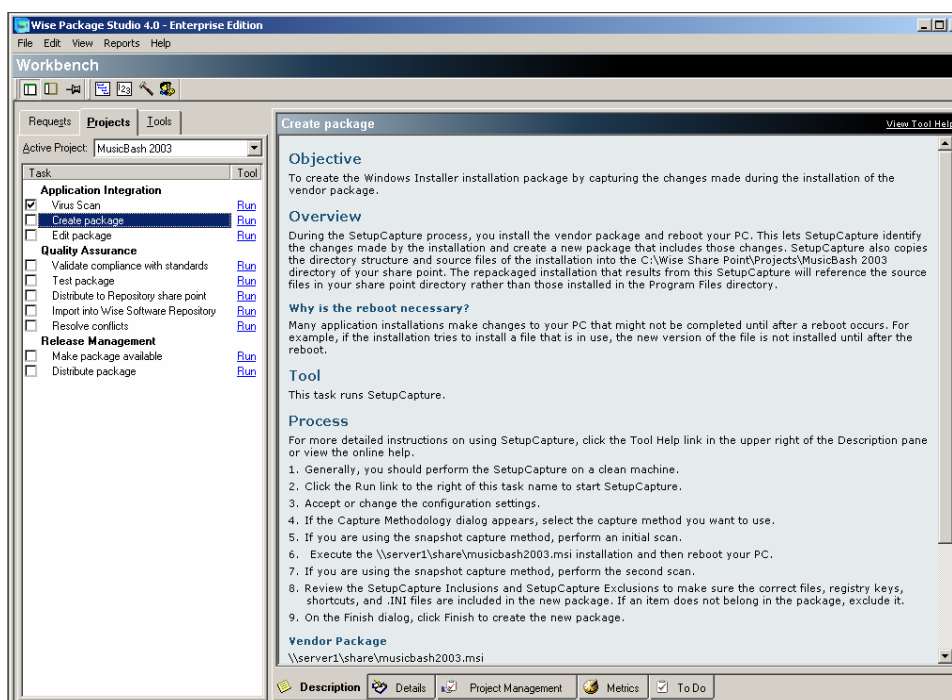


Figure 2.15: Use the Workbench to step you through your specific process for your specific project.

The process steps are seen on the left. Each step in the process should be executed as described in the notes on the right. Simply click the blue Run tags next to each step in a process, and a specific tool will be launched to assist in accomplishing that step in the process. Figure 2.16 highlights the idea that each step can be associated with one or more tools.

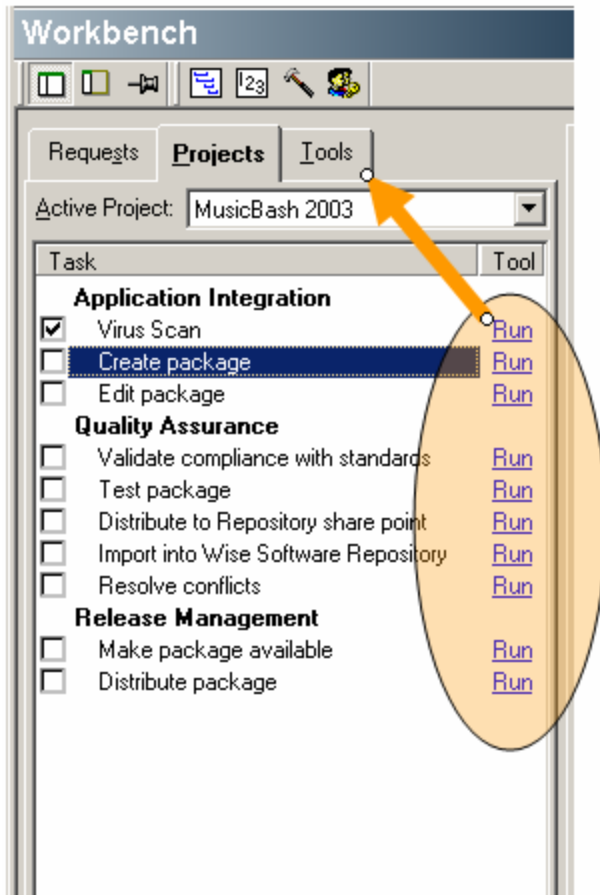


Figure 2.16: Each step in the process can correspond to one or more tools.

Again, selecting Run for a specific task will run a specific tool. Alternatively, you could select the Tools tab to expose all the tools that comprise Wise Package Studio, as Figure 2.17 shows. Each tool has a step-by-step description of how it accomplishes a task as well as a preview of those tasks.

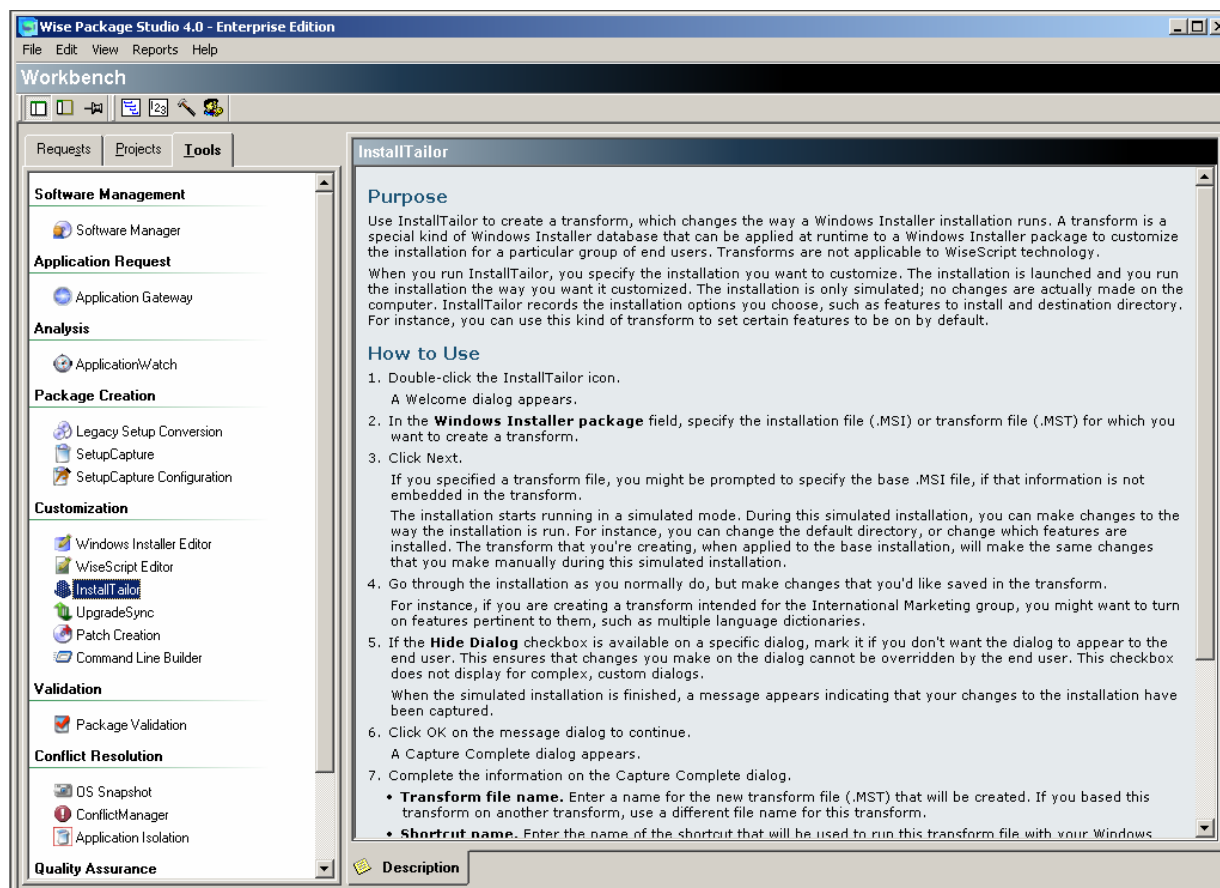


Figure 2.17: Each tool may be selected individually if required.

Wise Package Studio comes in three flavors: Standard, Professional, and Enterprise. Each comes with a different feature set:

- Standard Edition—Performs basic MSI repackaging via “ad hoc” approach. Wraps up packages and validates that they were packaged correctly.
- Professional Edition—Performs more advanced MSI repackaging. Geared for both a single administrator as well as multiple administrators working on the same project. This edition adds increased testing and conflict management and performs the process-oriented approach I’ve been describing.
- Enterprise Edition—This version is unique because it enables a full end-to-end lifecycle approach for MSI files. Specifically, this version allows for requests to be submitted about a package that an end user might want developed and can track the project’s progress through the life of the project.

AdminStudio 3.5

InstallShield is best known in the developer community for its tool that packages *new* applications—called InstallShield Developer. This tool is really a developer’s tool. In fact, it even hooks into Visual Studio .NET while the developer is working with it. However, InstallShield also offers an application that is geared toward administrators—AdminStudio. This product allows for both process-and-project driven MSI creation and a manner to launch individual tools to perform specific tasks.

For example, with the AdminStudio’s QuickStart Guide (see Figure 2.18) an administrator can simply hover the cursor over one of the proposed questions, and the program will replace the question box with an “answer” that describes which tool will be launched.

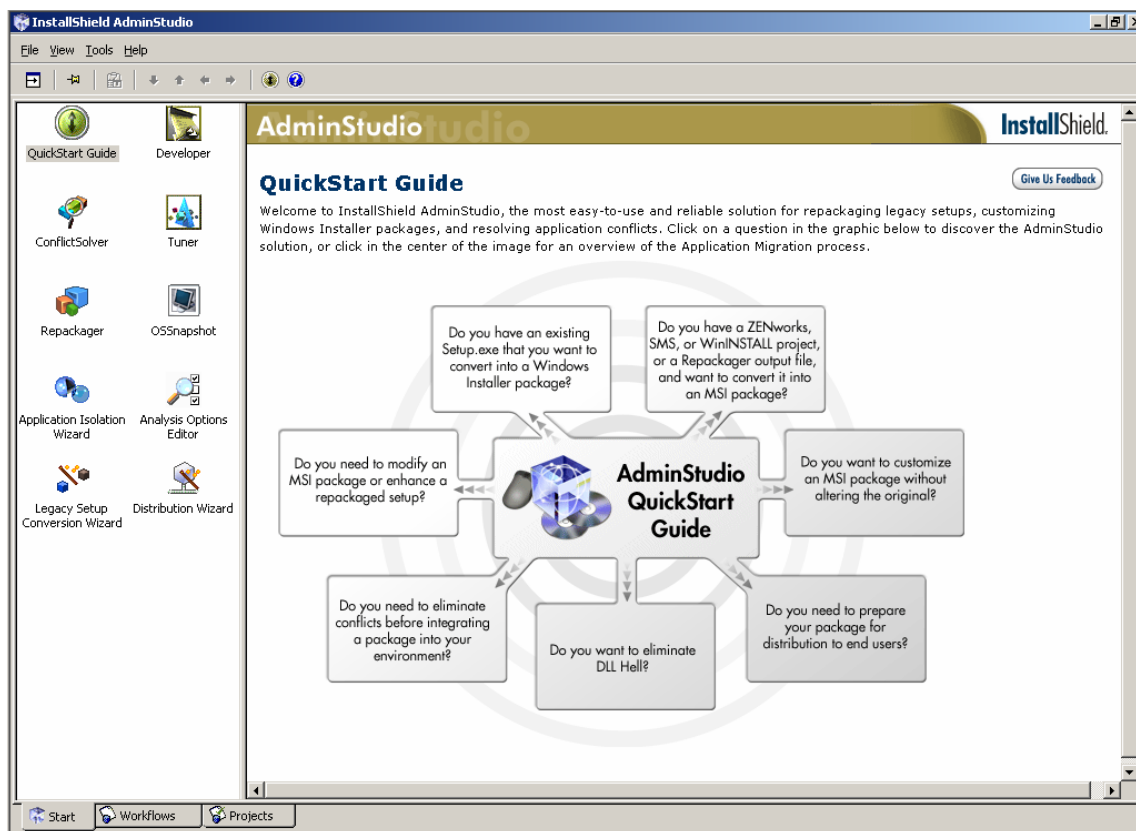


Figure 2.18: AdminStudio’s QuickStart guide.

Alternatively, if the administrator knows which tool she or he wants to use, the administrator can simply select that tool from the Start menu, as Figure 2.19 shows.

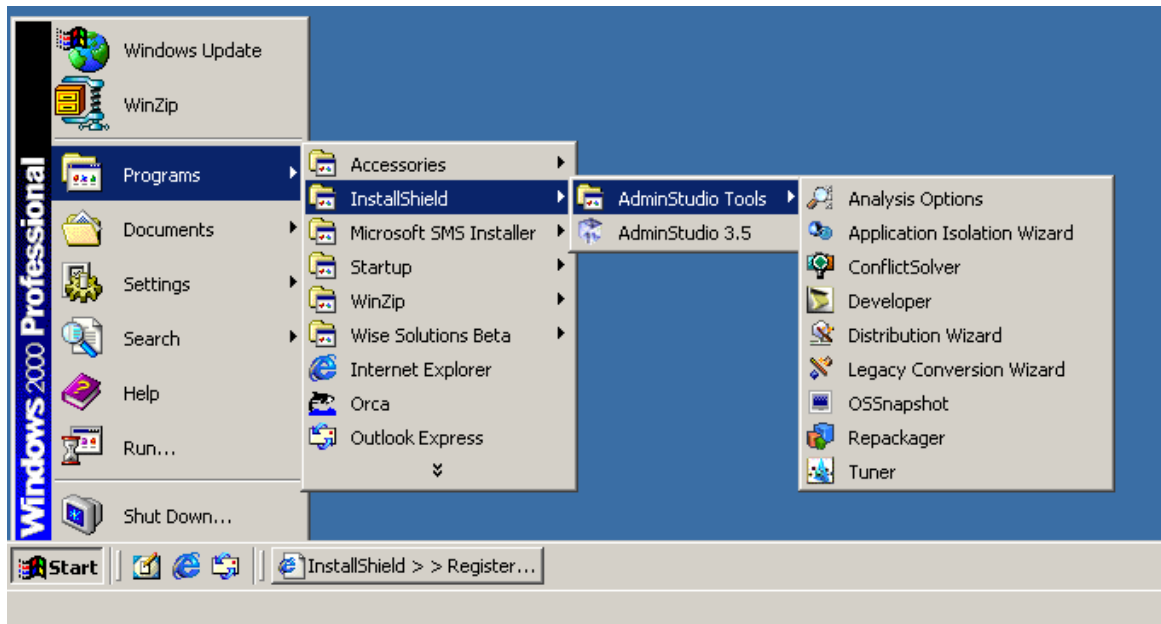


Figure 2.19: The individual tools are also available from the Start menu.

After launching a tool, a tutorial for the tool is displayed. For example, as Figure 2.20 shows, launching the Repackager tool will launch an 8-page tutorial that you can scroll through to get a better idea of what the tool is supposed to do.



Figure 2.20: The tutorial screens warm you up to each tool.

After you read through the tutorial screens, you have the option to actually launch the tool. Alternatively, AdminStudio can be set up to have a customized “flow” process, called a Workflow, as Figure 2.21 shows.

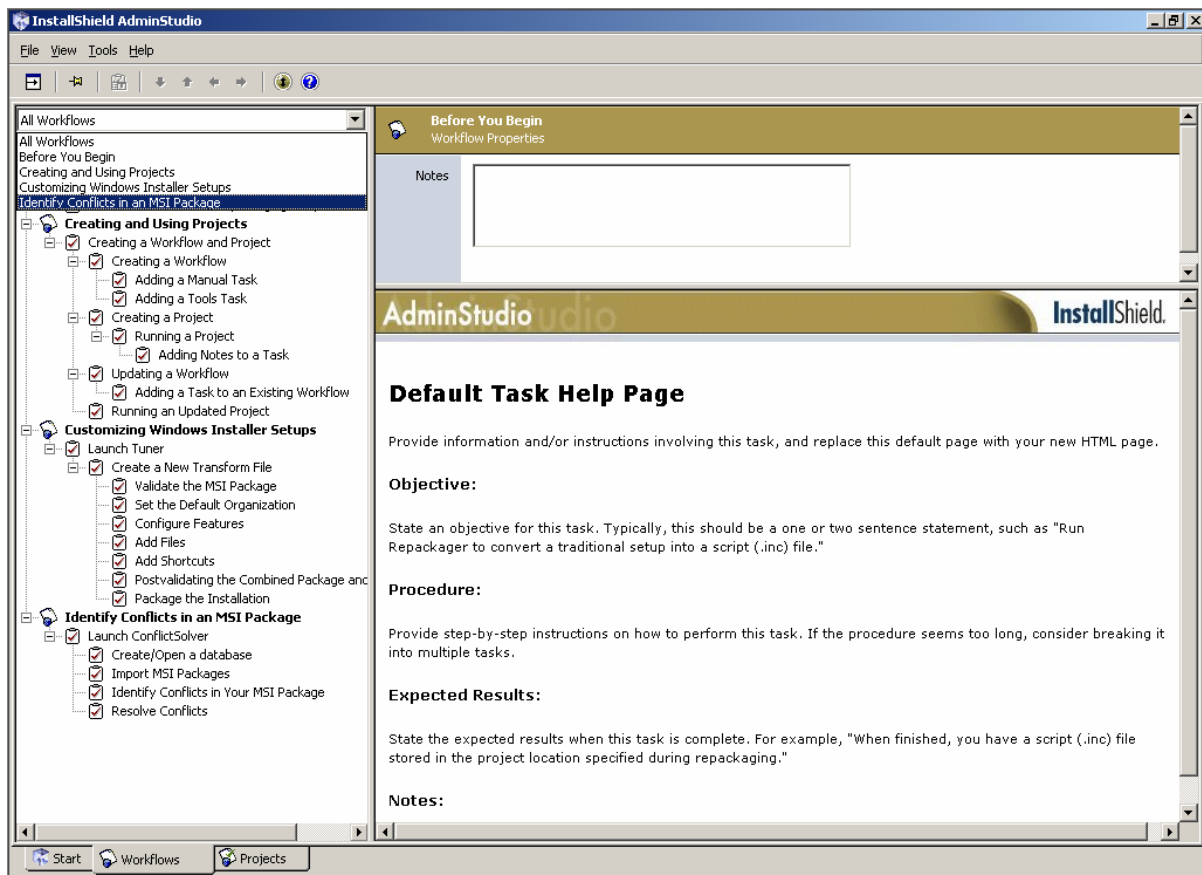


Figure 2.21: AdminStudio offers its Workflow to keep the projects on track.

Workflows have defined tasks and specific tools that can be associated with the task. Administrators can create new projects based on specific workflows.

AdminStudio 3.5 is really meant for a single administrator to perform the repackaging. It is not really a “collaborative” enterprise-ready tool allowing for an end-to-end process. This version comes in two flavors: Standard and Professional. Although the standard version will certainly do the job to get your packages into MSI format, the professional version is much more adept in the automated detection and correction of conflicts.

Prism Pack

Some commercial products are similar to WinInstall LE, but with a friendlier overall ease of use. Such is the case with Lanovation’s Prism Pack utility, which Figure 2.22 shows.

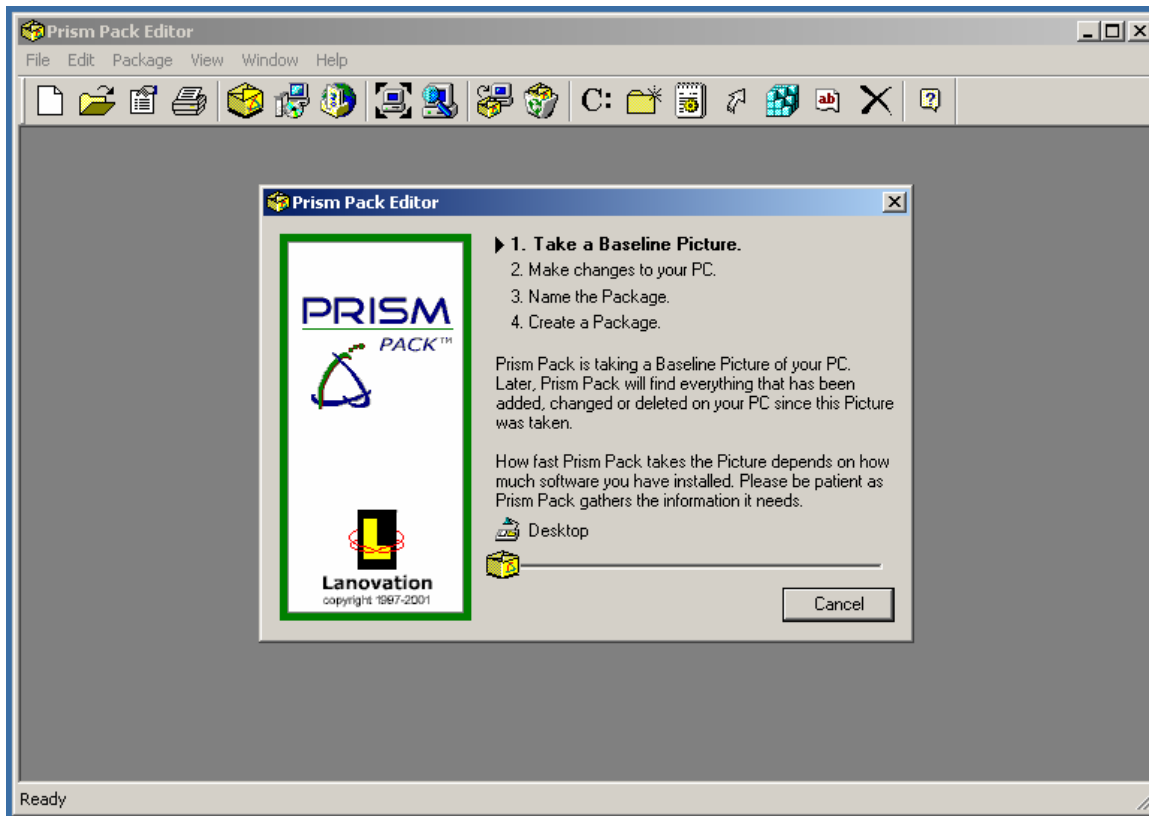


Figure 2.22: Prism Pack “takes off” upon first launch.

Once launched, the tool immediately begins performing a snapshot. It later provides the ability to make basic changes to the created package using the Prism Pack Editor (see Figure 2.23).

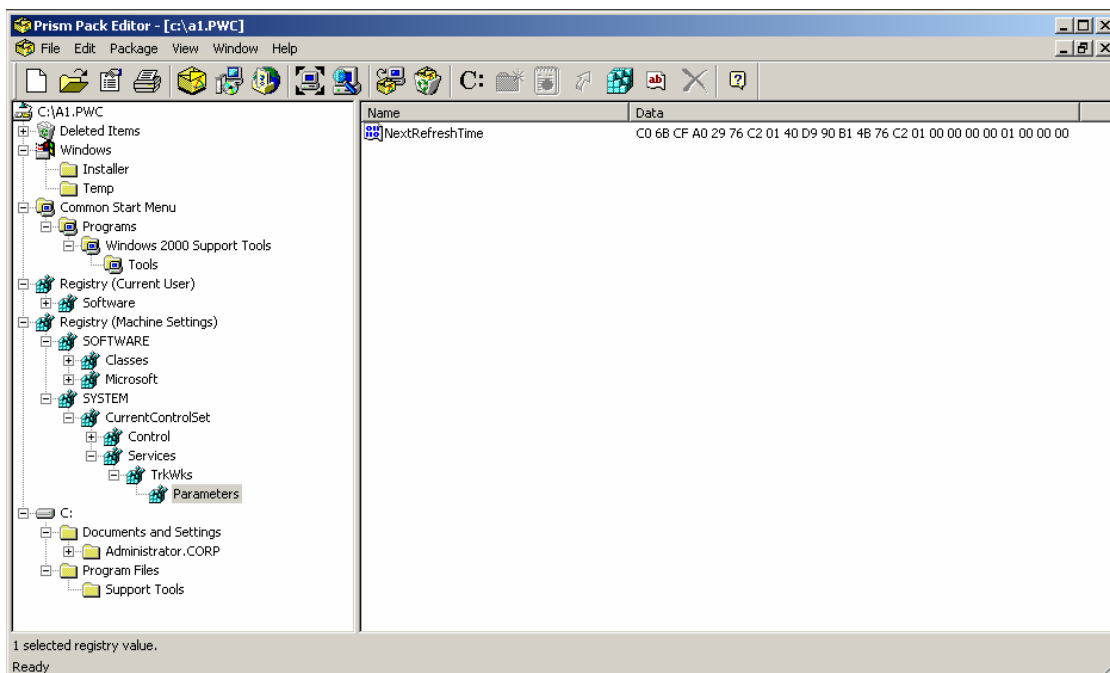


Figure 2.23: The Prism Pack editor is basic and easy to use.

This tool doesn't provide the collaboration abilities, process/workflow/project guides, and elaborate wizards of other third-party tools. However, what Prism Pack lacks in sophistication, it makes up for in ease of use—it's a solid tool that does the job of repackaging well. Sometimes you just need or want a tool that is simple and easy to use.

Added Functionality

As I've already stated, the idea behind third-party tools is to bring additional functionality beyond simple repackaging. Indeed, when choosing a tool, you might want to start with the marketing materials, check out the feature sets, then settle in for a test drive. Most third-party tools will offer the functionality you need to perform day-to-day MSI management functions:

- Snapshot repackaging
- MST (transform) development
- MSP (patch) development
- Easy package transfer to third-party *deployment* tools (such as SMS)
- Customization of the MSI via script, direct edit, or other method
- Ability to manage DLL and other component conflicts
- Ability to key files for self-healing

These features alone might make the ideal tool for you. However, there is still a challenge for the manufacturers of third-party MSI repackaging tools. Today, their challenge is to bring out new innovative features which either augment or replace the traditional "snapshot" methodology. I'll highlight two of the vendors that are doing so and how.

Wise Package Studio 4.0 Repackaging Innovations

Wise Package Studio 4.0 has new functionality beyond performing simple snapshots. This latest version of the Wise Package Studio allows for what Wise calls Virtual Capture (see Figure 2.24). Virtual Capture allows administrators to perform captures on machines that aren't totally clean. Such functionality is a boon for administrators, as cleaning a machine for another repackaging job is a major source of administrative headaches. Usually, administrators simply reformat and reload the machine to get back to a clean machine. But anytime a machine needs to be fully reinstalled, the task can take quite a while. Virtual Capture eliminates this annoyance.

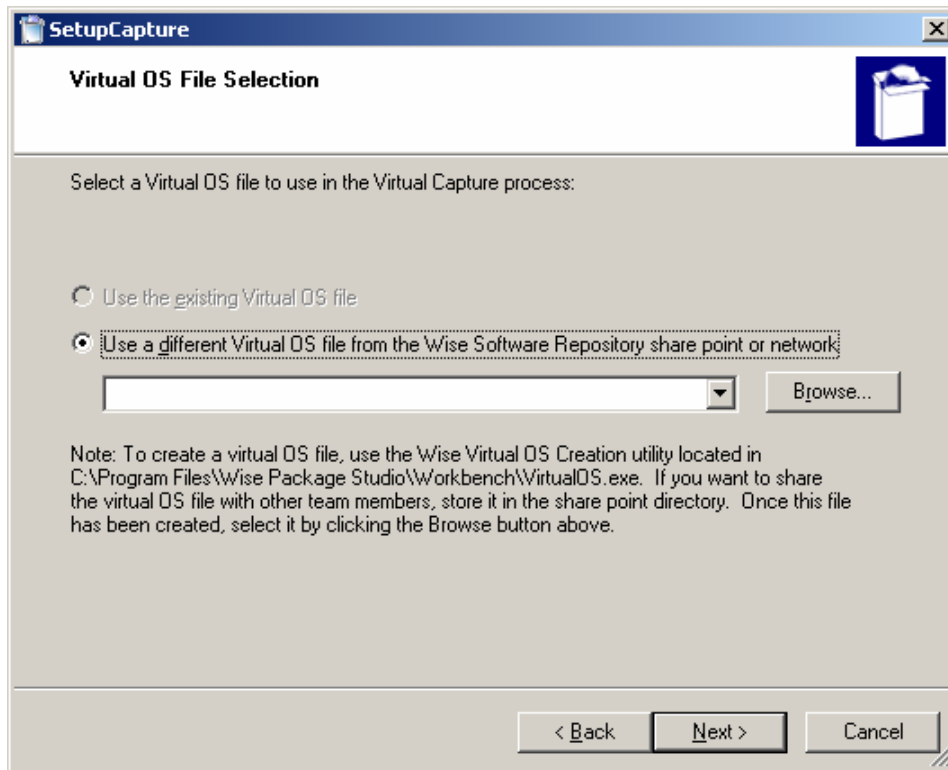


Figure 2.24: Virtual Capture eliminates the need for a clean system.

AdminStudio 3.5 Repackaging Innovations

InstallShield’s AdminStudio 3.5 also has functionality beyond simple snapshots. Its latest innovation is called InstallMonitor. This feature is unique because it runs while the application is being installed, monitors what’s going on, and actively records the changes occurring on the system. Therefore, there is no before or after snapshot. You simply start InstallMonitor, fire up the setup.exe on the application you want to create the package for, and voila. Over time, the lost minutes waiting for before and after snapshots can really add up. However, it’s still best to reimage your machine with a fresh installation after every package capture.

Shareware and Freeware Third-Party Tools

There is an emerging “home brew” market for shareware and freeware tools that help with MSI creation. Shareware tools can be useful, but ensure that if you work with a shareware vendor that you’ll get the same level of support as if you went with a commercial vendor. Working with a freeware or shareware vendor one-on-one might produce a relationship you can be comfortable with. And if you’re comfortable with the relationship and the product, you’ve got a good candidate for a useful tool.

Two tools that fit in this category include MSICreate from Corner House (<http://www.cornerhouse.ca/en/msi.html>) and Setup2Go for MSI from Offshore Development Software/SDS Software (<http://www.dev4pc.com/downloads.html>).

There are also two free tools that are in various stages of readiness. One is called izfree and can be found at <http://izfree.sourceforge.net/>. This program doesn't set out to be a huge application, rather, it just wants to get the job done. This tool is in 1.0 alpha release, so use it at your own risk. The other free tool is called NInstall, and it's not even ready for primetime, but you can find more information about it at <http://www.chimpwithkeyboards.com/projects/ninstall/index.shtml>.

Summary

Exposure to as many MSI repackaging tools as possible can only be a good thing. Otherwise, how will you know what you like? Although the Microsoft tools do an adequate job, they usually fall a little short in the features category. Third-party tools can be pricey, but ultimately worth it if you take the job of repackaging seriously. Shareware and freeware tools sound great, but be sure that you're comfortable with whatever support relationship you work out with the developers.

No matter which tool you investigate, remember that each tool has a different approach, user interface, feature set, and price point. Getting a trial copy and taking a tool for a test drive is really the best approach to figure out which tool is ultimately right for you.