

**DESKTOP AUTHORITY**  
**MSI STUDIO™**



**Desktop Authority MSI Studio**  
**Quick Start Guide**

**SCRIPTLOGIC**

© 2007 by ScriptLogic Corporation

**All rights reserved.**

This publication is protected by copyright and all rights are reserved by ScriptLogic Corporation. It may not, in whole or part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from ScriptLogic Corporation. This publication supports Desktop Authority MSI Studio. It is possible that it may contain technical or typographical errors. ScriptLogic Corporation provides this publication "as is," without warranty of any kind, either expressed or implied.

### **ScriptLogic Corporation**

6000 Broken Sound Parkway NW  
Boca Raton, Florida 33487-2742

1.561.886.2400

[www.scriptlogic.com](http://www.scriptlogic.com)

### **Trademark Acknowledgements**

ScriptLogic and the ScriptLogic logo are either registered trademarks or trademarks of ScriptLogic Corporation in the United States and/or other countries. The names of other companies and products mentioned herein may be the trademarks of their respective owners.

## DOCUMENTATION CONVENTIONS

### Typeface Conventions

**Bold** Indicates a button, menu selection, tab, dialog box title, text to type, selections from drop-down lists, or prompts on a dialog box.

## CONTACTING SCRIPTLOGIC

ScriptLogic may be contacted about any questions, problems or concerns you might have at:



ScriptLogic Corporation  
6000 Broken Sound Parkway NW  
Boca Raton, Florida 33487-2742



561.886.2400 Sales and General Inquiries  
561.886.2450 Technical Support



561.886.2499 Fax



[www.scriptlogic.com](http://www.scriptlogic.com)

## SCRIPTLOGIC ON THE WEB

ScriptLogic can be found on the web at [www.scriptlogic.com](http://www.scriptlogic.com). Our web site offers customers a variety of information:

- Download product updates, patches and/or evaluation products.
- Locate product information and technical details.
- Find out about Product Pricing.
- Search the Knowledge Base for Technical Notes containing an extensive collection of technical articles, troubleshooting tips and white papers.
- Search Frequently Asked Questions, for the answers to the most common non-technical issues.
- Participate in Discussion Forums to discuss problems or ideas with other users and ScriptLogic representatives.

# Contents

<b>WHY USE PACKAGING SOFTWARE?</b> .....	5
<b>WHAT IS WINDOWS INSTALLER?</b> .....	5
<b>TYPES OF VENDOR SUPPLIED SOFTWARE INSTALLATIONS</b> .....	5
LEGACY SOFTWARE INSTALLER .....	5
EXISTING WINDOWS INSTALLER SOFTWARE PACKAGE .....	6
INSTALLATIONS THAT CANNOT BE REPACKAGED IN MSI FORMAT .....	6
<b>THE PACKAGING PROCESS USING MSI STUDIO</b> .....	6
<b>MODIFYING AN EXISTING MSI FILE</b> .....	8
EXISTING MSI FILE EXAMPLE .....	9
<b>REPACKAGING A (LEGACY INSTALLATION) APPLICATION</b> .....	14
REPACKAGING EXAMPLE .....	16
<b>CREATING A WRAPPER INSTALLATION</b> .....	23
WRAPPER INSTALLATION EXAMPLE .....	24

# Getting Started

Packaging Software is the process of creating standardized software installation packages for the applications that are to be installed on computers on your network. Packaging using Windows Installer allows for easy deployment using Group Policy, Active Directory and commercial deployment tools.

## WHY USE PACKAGING SOFTWARE?

Traditional software installers come in many formats, many are script based, some are executable code, and others just come with a set of manual instructions of where to copy files and how to register them.

The problem with having a variety of installation methods is that there is often no way of knowing what changes are being made to a system, and no way of monitoring potential conflicts between applications, or repairing an application if it is placed into an unstable state. It is common with legacy installations for one application to place another in an unstable state by modifying a component.

## WHAT IS WINDOWS INSTALLER?

Windows Installer is a service which runs on Windows and installs, repairs, or removes software according to instructions contained in .MSI files. A local cache is always maintained of this .MSI file and Windows Installer makes checks of applications to maintain them in a working state. Checks are made when running shortcuts and file extensions or other 'entry points' into an application.

## TYPES OF VENDOR SUPPLIED SOFTWARE INSTALLATIONS

Software applications can generally be grouped into one of three categories for packaging purposes. These are described below:

### **Legacy Software Installer**

This is a software installation which is in non-MSI format (could be script, .exe, or manual). These types of software installations will be "repackaged". Essentially what will happen is the changes that they make to a system will be captured and a new Windows Installer-based installation package will be created for the application.

## Existing Windows Installer Software Package

---

As software vendors come to understand the benefits of using Windows Installer-based installations, more of them are distributing their software as .msi files (or a .msi file wrapped in a .exe).

It is generally accepted that vendor-supplied Windows Installer packages should not be modified. There are two main reasons why you should not repackage vendor supplied .msi files.

It's a waste of time repackaging something back into its original format (although the shoddy state of some of them might make you think otherwise!)

If you repackage the original msi file then any updates or patches supplied by the vendor will also need to be repackaged. This can lead to problems further down the track.

However, it is possible to modify a .MSI file using a 'transform' which is a type of Windows Installer database that is merged with the original at runtime. The main reason you use transform files is to create 'responses' to options given during the UI sequence of an install, for example, to supply the serial number and select the features you want installed so the installation can be run silently.

## Installations that cannot be repackaged in MSI Format

---

This includes such things as driver installations, windows service packs, updates and hotfixes. Hardware changes should not be captured by doing a repackage as changes differ on each machine. Windows Updates should not be repackaged as they often change protected files which Windows Installer cannot access.

In cases like these it is often easier to wrap the executable or vendor supplied installation in an .msi file to make deployment easier.

## THE PACKAGING PROCESS USING MSI STUDIO

### Step 1. Test Your Application Before Starting

The first step, and one which is often over looked is to test the software before you start to repackage it. The aim of this step is to make sure that it works on your system. If it doesn't work then there is not much point in repackaging it. Also it is worth checking with the vendor to see if there is a more recent release of the application which the product owner may wish to purchase.

Therefore, install, using the vendors installation procedure, the application and make sure that it works. Also, while installing it you should note the method of installation (check to see if it is using Windows Installer, see step 2), and the steps involved in configuring it.

## **Step 2. Find out the Method of Installation**

This step involves deciding which of the repackaging methods outlined in the previous section is the correct method to follow for your application. The first step in deciding this is to see whether it is a Windows Installer installation.

First, check the CD or installation source for signs of an .msi file. Often, an installation may come with a .msi file and an executable, which might serve no other purpose than to check that the Windows Installer runtimes are present.

If there is no .msi file but an executable, you should also check that the .msi is not compressed inside the executable. To do this you can run it and check the files in the temporary directory. Often an .msi file and related dependant files might be uncompressed in here. Also check the Task Manager and see whether or not an extra msiexec.exe process starts - this is another sign that the installation is using Windows Installer.

If it is not Windows Installer based you need to check as to whether it can be repackaged or not. Often it is not possible to determine this until well into the repackaging process but earlier checks can save you time. Windows Updates often update protected system files and therefore cannot be repackaged. Legacy driver installations are also another example of an application which can cause problems when repackaging. Although you should try and repackage everything into MSI format just be aware that this is not always possible in which case you might want to 'wrap' the applications installation in a .msi.

## **Step 3. Follow the Appropriate Process**

Below we will outline the 3 main processes to follow.

### **Getting Started With MSI Studio**

When repackaging it is recommended that you have at least 3 machines which can be easily rebuilt with your clean build image. This can be achieved by having 3 physical machines and rebuilding them with a utility such as Ghost, or by using virtual machines. One machine can be used as your development machine, one as a test machine and one as a clean machine to do the repackaging on. In this document these will be referred to as your 'Development', 'Repackaging', and 'Test' machine.

It is important to keep the Repackaging machine clean by rebuilding it between capturing changes made by an application. You will need administrator privileges on the Repackaging machine and need to be able to access the main drives either by a share (c) or admin share (c\$). Also you will need permission to launch remote applications through Windows Management Instrumentation.

When repackaging be sure that you are logged onto the Development and Repackaging machine using the same username. When testing on the Test machine, make sure you use different users with limited privileges.

## MODIFYING AN EXISTING MSI FILE

This is normally the quickest and easiest of the repackaging processes. It occurs when the vendor's application package is in MSI format. What you wish to accomplish is to be able to run this silently with the correct configuration options. These might include accepting a license agreement and selecting features to install and the directory to install them.

This is accomplished by creating a 'Response Transform' which is then deployed with the vendor's .MSI and provides it with information specific to your system.

1. Copy the Source Media to Your Network Repository  
'\\All Applications\My Application\'
2. Start IDS and on the first welcome screen select '*Repackage an Application for Deployment around a Corporate Network*'
3. Select '*Modify an Existing MSI file*'
4. Fill in the required information
  - In the first field select the vendor's msi file.
  - Select '*Create a Response Transform*'
  - In the bottom field enter the name you wish to save the created transform to.
5. Press the GO Button.
6. Create Response Transform Screen
  - Now you are ready to step through the UI sequence of the msi file you wish to create a response transform for. Note that it will **not** be installed, only the dialogs will be displayed. Enter the details required and make the selections necessary to install it on your system.
7. Once you have finished running through the UI sequence a dialog will be displayed prompting you to open the created transform.
8. Make Further Changes to the Transform. Sometimes you might have environment or company specific registry settings which need to be added. Do this now and then save *File>Save* (Ctrl-S).
9. Test

You should now be ready to test the transform and msi. Go to your test machine and run the original msi with the transform you created. This can be done using the following command line:

**Msiexec -i originalmsi.msi TRANSFORMS="mytransform.mst" -qn**
10. Open and Make Further Changes (if necessary)

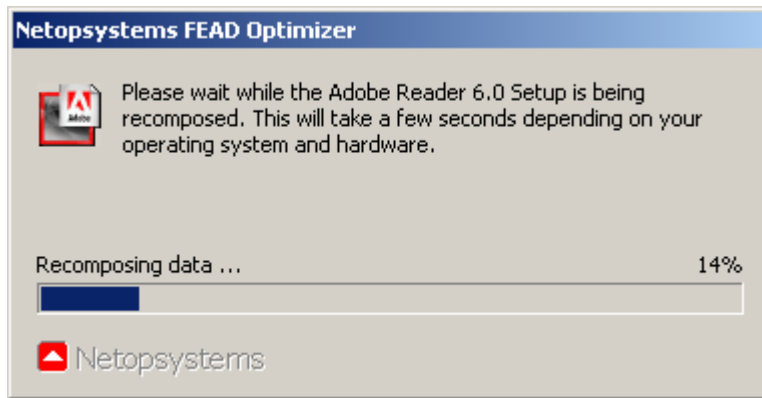
11. If you need to open the transform and make further changes, from the file menu select *File>Open* and then in the browse dialog box select the transform file. A popup dialog box will appear asking if you wish to open it with the original .msi file, select 'yes' and enter its name if it does not already appear.

## Existing MSI File Example

In this example we will create a Response Transform for Adobe Reader 6.

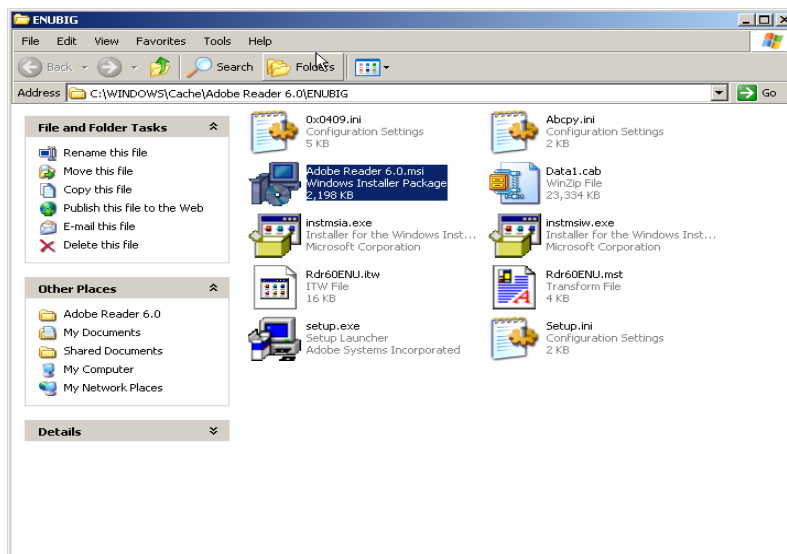
1. What Type of Vendor Installation?

When you download Adobe Reader it comes as one file **AdbRdr60\_enu\_full.exe**. This does not give many clues to the type of installation as it looks like all the files are compressed inside. Go to your test machine and run the executable.

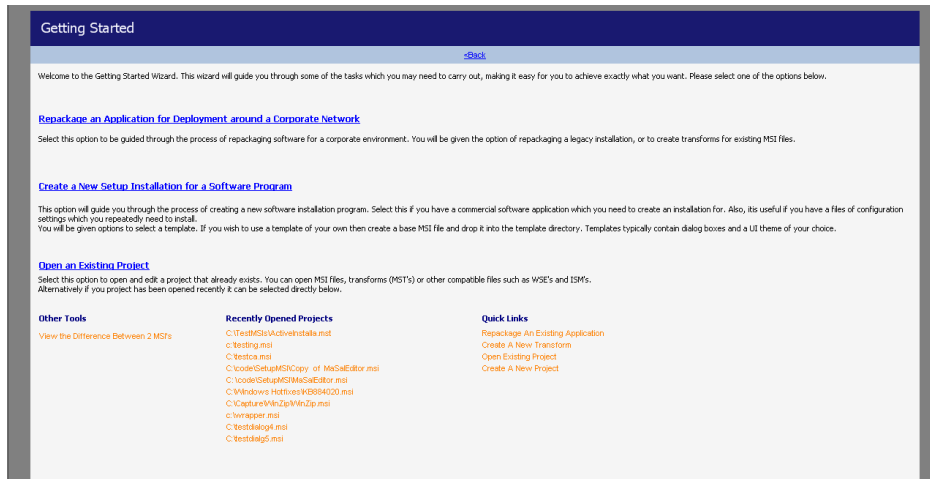


It appears to be extracting temporary files, so go to your temporary directory and look for the files. In this case I found them in the **Windows\Cache** directory...

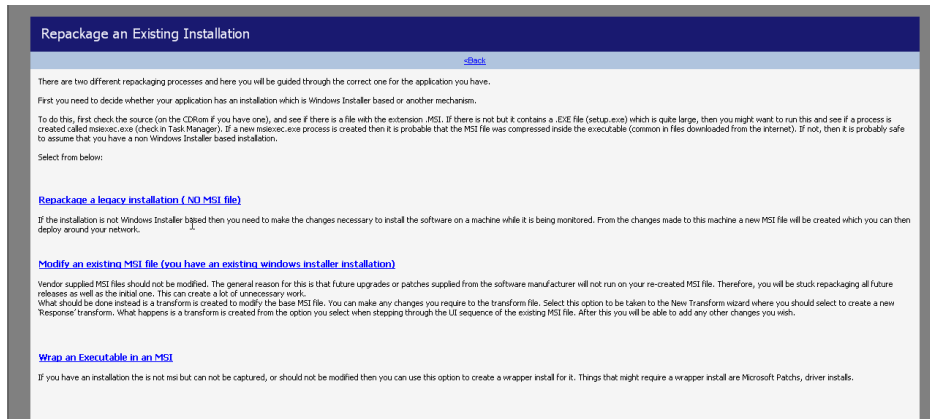
Copy these files to your project source directory. The files that are actually required are the .msi file, the .cab file, and the .mst file.



## 2. Run IDS and Select "Repackage An Application For Deployment around a Corporate Network"

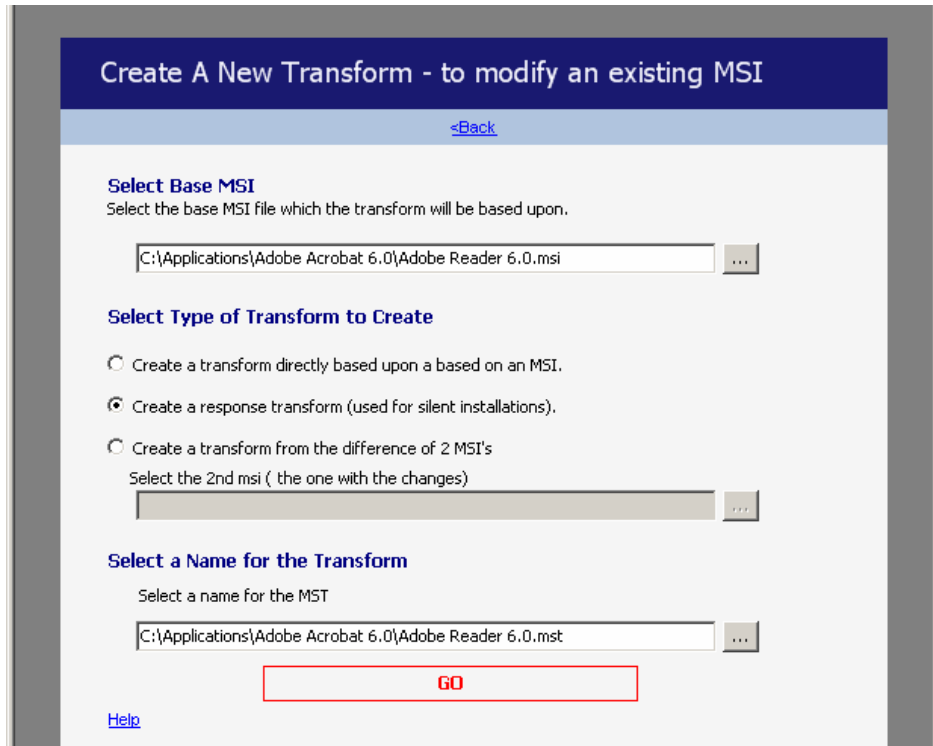


## 3. Select 'Modify an existing MSI file'

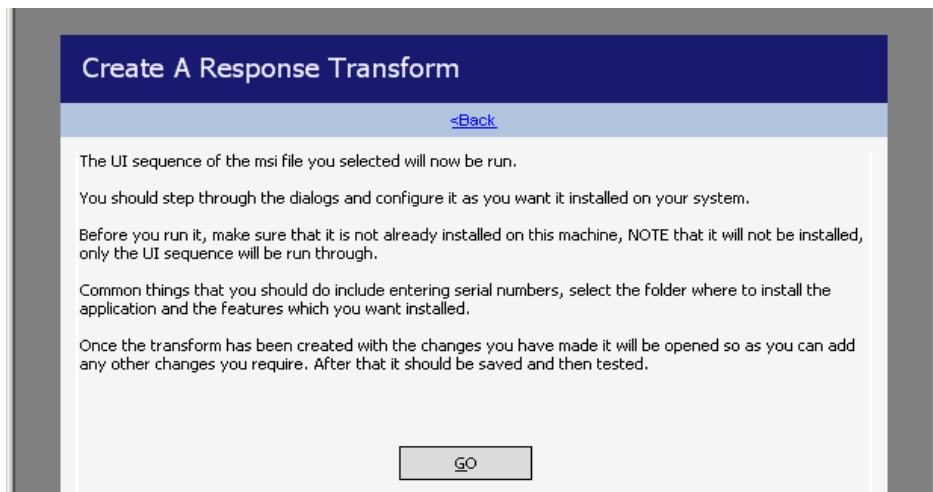


4. Enter Details

- Browse to the msi file in the top field.
- Select "Create a Response Transform"
- Enter the name of the transform file in the bottom field.
- Press the "GO" button.

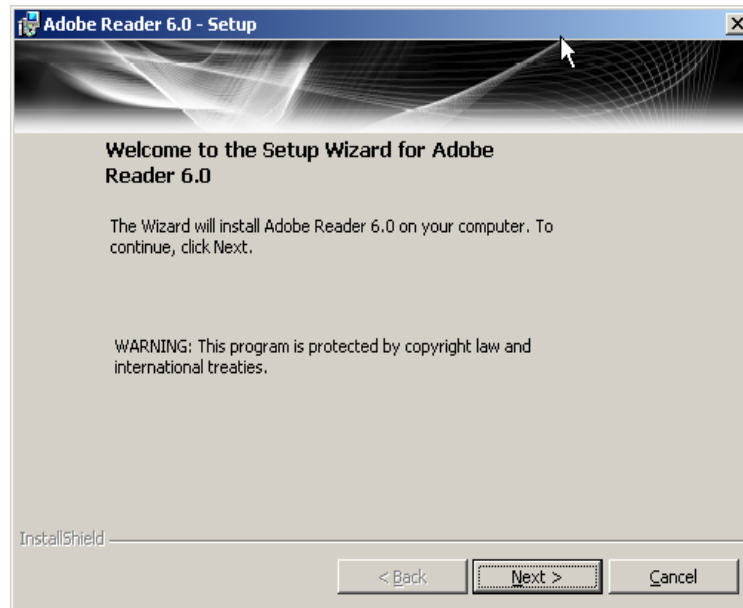


5. This screen just gives some instructions. Press the "GO" button.

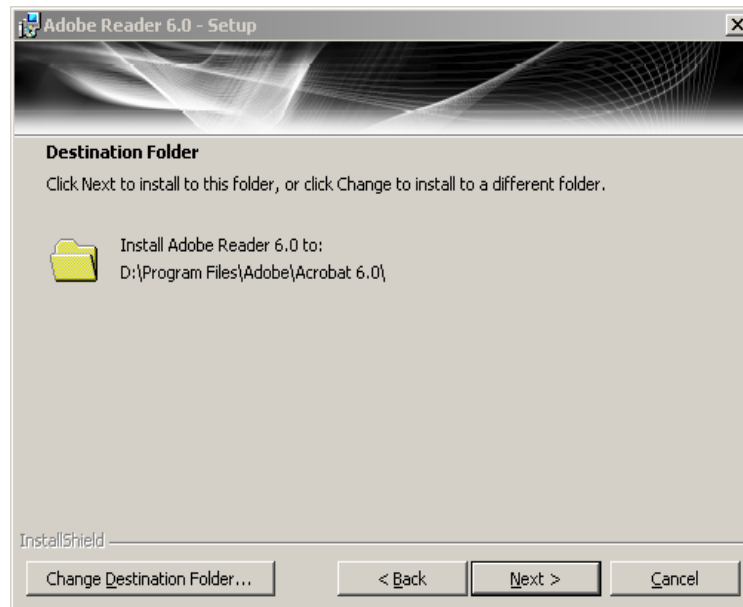


6. Now run through the UI.

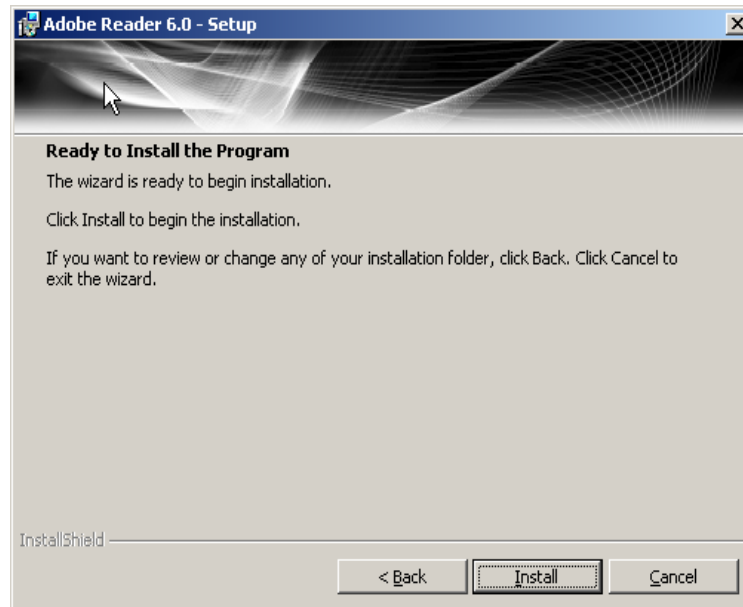
- Click 'Next' on the welcome screen



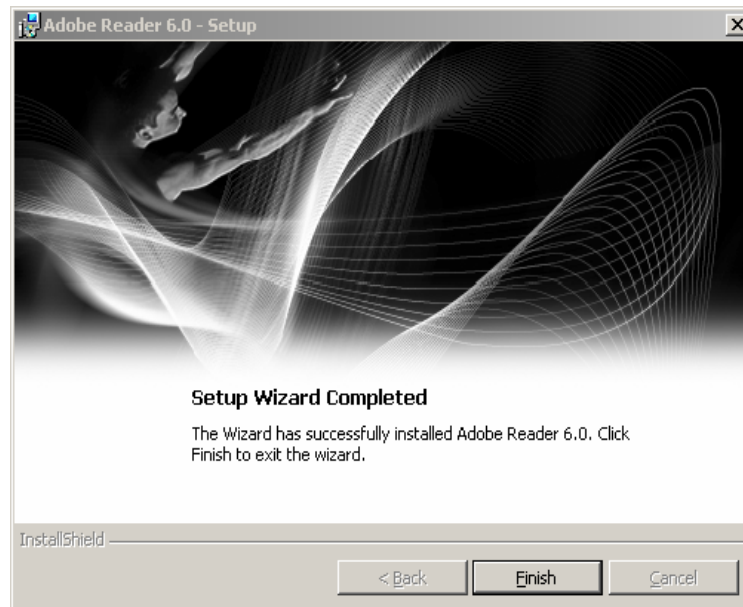
- The program directory is correct so click 'Next'



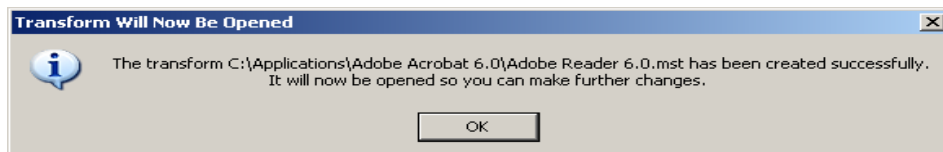
- Click 'Install'



- Click 'Finish'



7. Click 'OK' and the transform will be opened.



The transform will be opened and changes will appear in the table view in yellow for added rows and blue for changed rows.

8. Test the MSI and Transform

9. We don't need to make any further changes to the transform so we can save and close it, move to our test machine and run the .msi and transform using the command line:

```
msiexec -i "\\desktop\c\Applications\Adobe Acrobat 6.0\Adobe Reader 6.0.msi" TRANSFORMS="\\desktop\c\Applications\Adobe Acrobat 6.0\Adobe Reader 6.0.mst" -qn
```

## REPACKAGING A (LEGACY INSTALLATION) APPLICATION

The second process is to repackage an application into an .MSI as its current installation is in non-Windows Installer format.

1. Start IDS and on the first screen select 'Repackage an Application for Deployment around a Corporate Network'
2. Select 'Repackage a legacy installation' on the second wizard screen.
3. Fill in the fields accordingly:
  - In the top field enter the name of the file to save the .msi to.
  - Enter the name of the machine on which you want to perform the capture of the changes as well as the username and password to use. This should be the name of the Repackaging machine (if you have entered this before it should be in the drop down list). The username and password should be the same as you are currently logged on to the development machine and should have administrator privileges.
  - In the next field select whether to do an initial scan of the Repackage machine. You should always select to do a scan unless you have previously done one and NOT made any changes to the repackaging machine since that time.
  - The next field gives 3 options. Generally it is recommended not to check any of these options as it is possible to do all these things later once the repackaging process has been completed. Making these changes at a later time gives you the option of first making a copy of the unchanged .msi. The three options are:
    - a. To search for, and, add merge modules containing files which have been captured in the repackaging process instead of the files themselves.
    - b. To display a dialog listing the files and registry entries which are about to be added to the project, allowing you to remove those that are not required.
    - c. To create advertising table entries from the registry table entries. This involves creating Appid, Class, Typelib, Extension, and ProgId table entries.

- The next option is whether to copy files to the source repository. It is recommended you select this option and specify a sub-directory below where you are saving the .msi file to. Selected files which are captured during the repackaging process will be copied from the repackaging machine to the directory specified allowing you to rebuild this machine and maintain a copy of the files required to rebuild the .msi. If you do not have a copy of these files then you can run into problems making changes at a later date.
  - The next option is to select whether to use file system monitoring against doing a before-and-after scan of the files on the repackaging machine. It is recommended you select this option as it is much quicker. If you have experienced any problems then you might wish to do a full scan.
  - Next click the *'Edit Drives to Watch'* to display the directories to watch during the repackaging process. Generally include just the drives where Windows, your program files folder and profiles are stored.
  - Finally you can edit the exclusions list. This is a list of registry keys, directories and files which should be excluded from any project.
4. Select *'Next'* and wait while an initial scan is made of the Repackaging machine.
  5. Once this is complete go to the Repackaging machine and run the legacy software installation program. Also make any configuration changes necessary and start up the application.
  6. Once all necessary changes have been made go back to the Development machine and click "Next" to start the post scan of the Repackaging machine.
  7. Once the scan has been finished, files and registry entries will be added to the project and the project opened.
  8. The first thing you need to do is select the *IQViews* and go to the *Product Details* panel. Enter details relating to the project name, manufacturer, version number and other relevant information.
  9. Next go to the *File* view and click through all the folders in the lower project view. Delete any files which you know are not required in the project, these might include any installation and uninstallation files and log files. Knowing what to remove comes with experience - so keep practicing!
  10. In a similar manner go to the registry view and remove any registry entries which are not required.

11. Convert Registry Information to Advertised Information.

- This step is not always necessary and should only be done if it's part of your packaging team's policy. Advertised entries are necessary if you are implementing Install-On-Demand. Create advertised entries by selecting from the menu *Tools>Extract/Convert Registry COM*. Select only the middle option *'Convert Registry Tables Entries to COM Tables'* in the resulting dialog box and press OK.

12. Validation of the Package.

- The amount of validation that you do of repackaged applications depends on your QA process. It is recommended that you remove all validation errors except for those that you are sure will not cause problems. IDS comes with 2 forms of validation, the first is MCE which is a super fast validation developed by Scriptlogic, and ICE validation which is the validation provided by the SDK. We recommend that you first use the MCE validation and once you have got rid of all those errors you think necessary, you then use the ICE validation as a final check. MCE is based upon ICE and you will see similar results.

13. After validation, fix the problems that occur, and repeat the process until you are satisfied with the state of the project.

14. Test the project.

- On your test machine test both the running of the installation - that no errors occur - and that the application works after installation. If you are having problems with the installation it is a good idea to use logging (see SDK help).

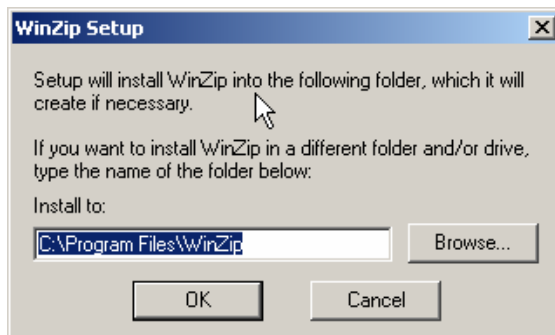
### Repackaging Example

---

For an example of an application that needs to be repackaged we will use WinZip 9.0.

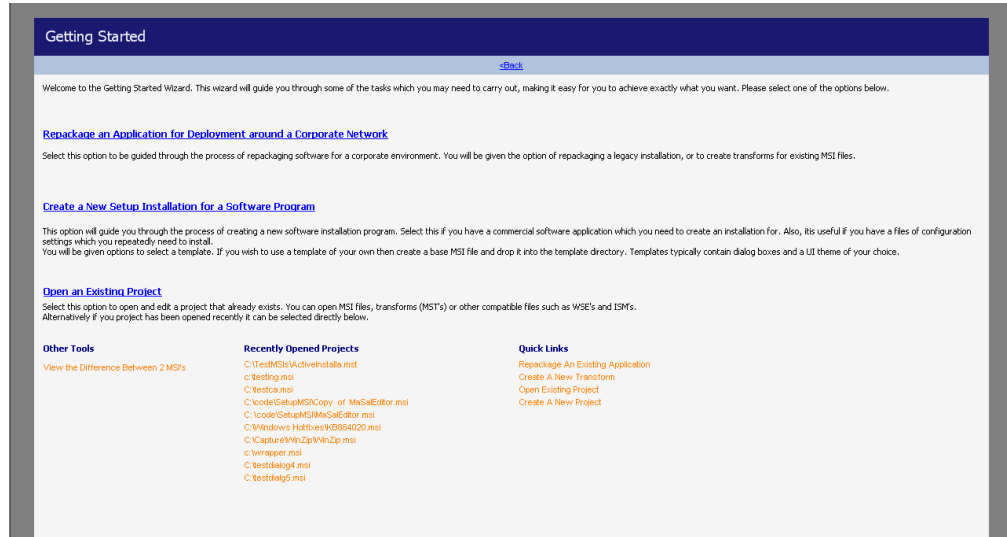
1. Select the Process.

Winzip 9.0 is downloaded as a file called winzip90.exe. This does not offer too many clues as to whether it is a Windows Installer file or not. Running it produces the following dialog box :

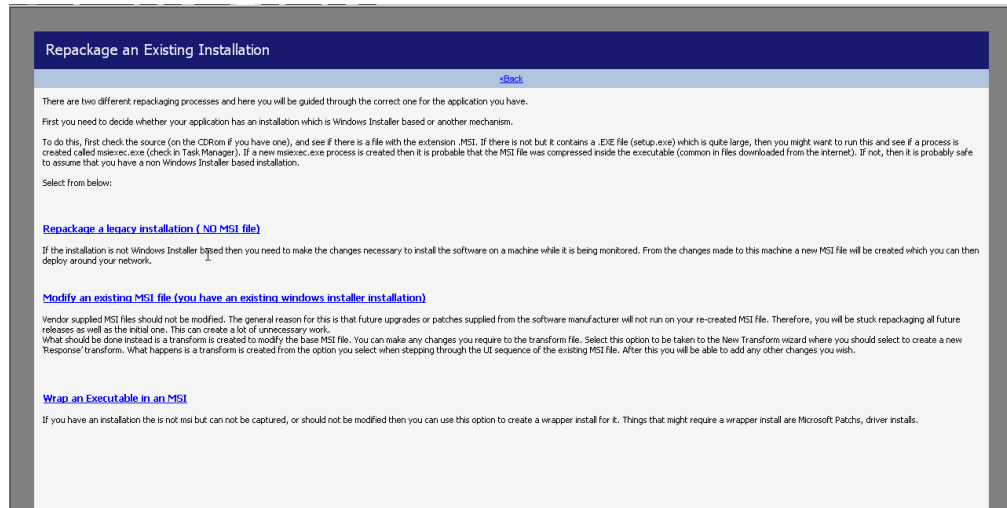


If we check the Task Manager there has been no extra Windows Installer processes started (msiexec.exe) therefore we can conclude that this is a non Windows Installer installation.

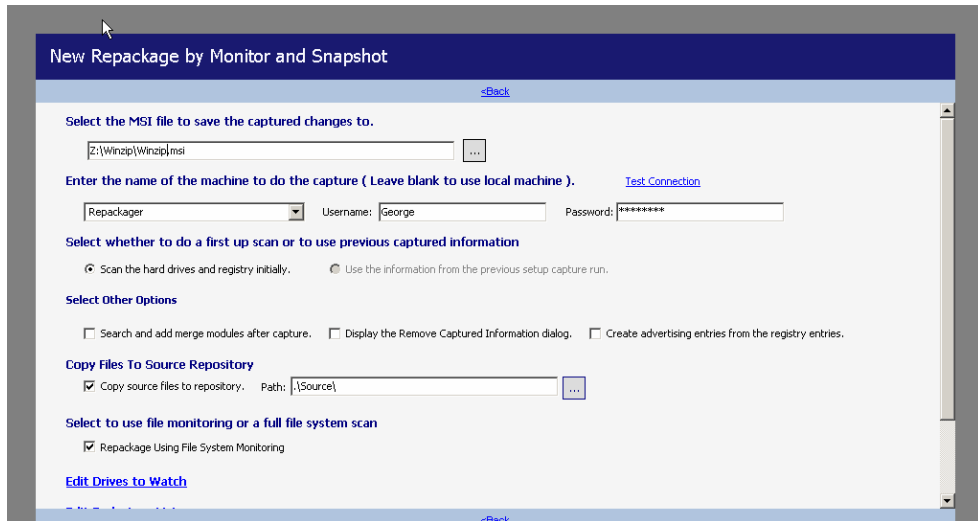
## 2. Run IDS and Select 'Repackage An Application For Deployment around a Corporate Network'



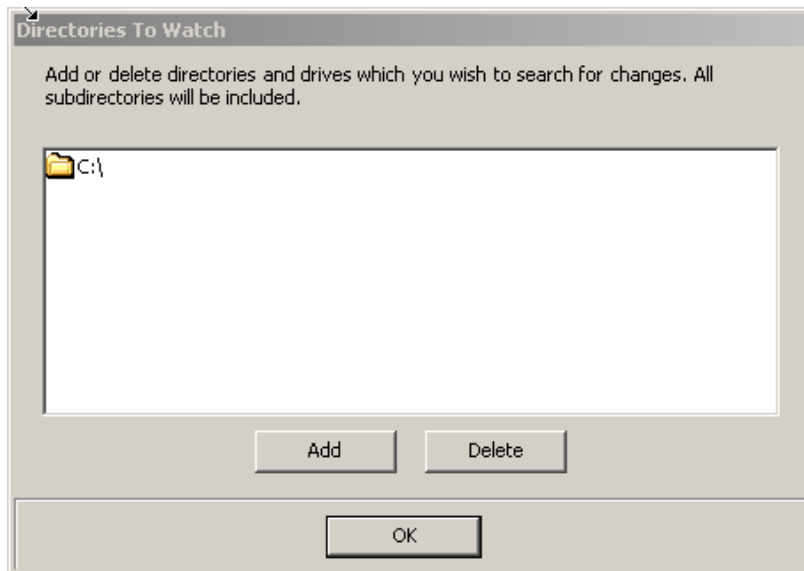
## 3. Select 'Repackage a legacy installation'



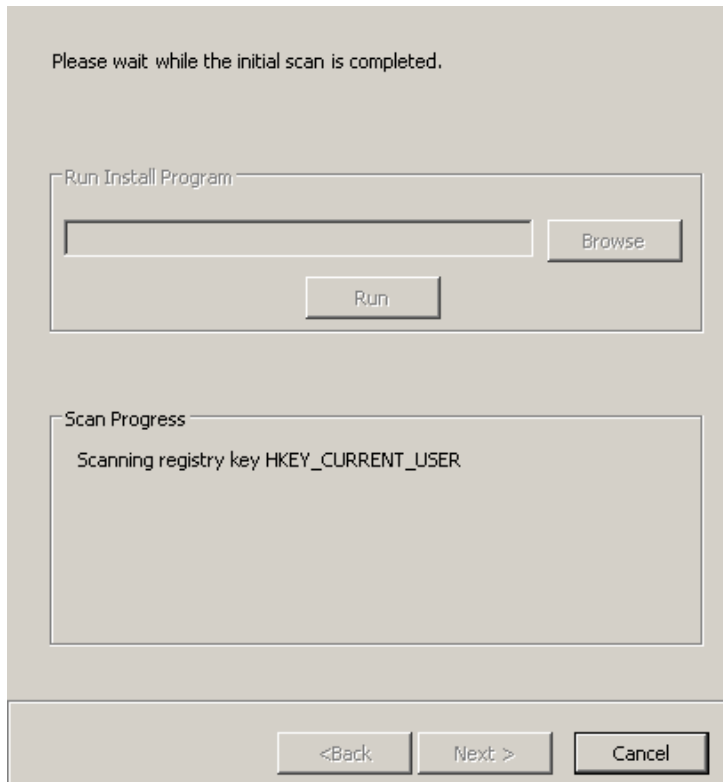
- Here we specify the name of the file to save the project to and make the selections as outlined in the previous section. The name of the machine we are doing the repackaging on is 'Repackager' and this has been recently rebuilt with a fresh image.



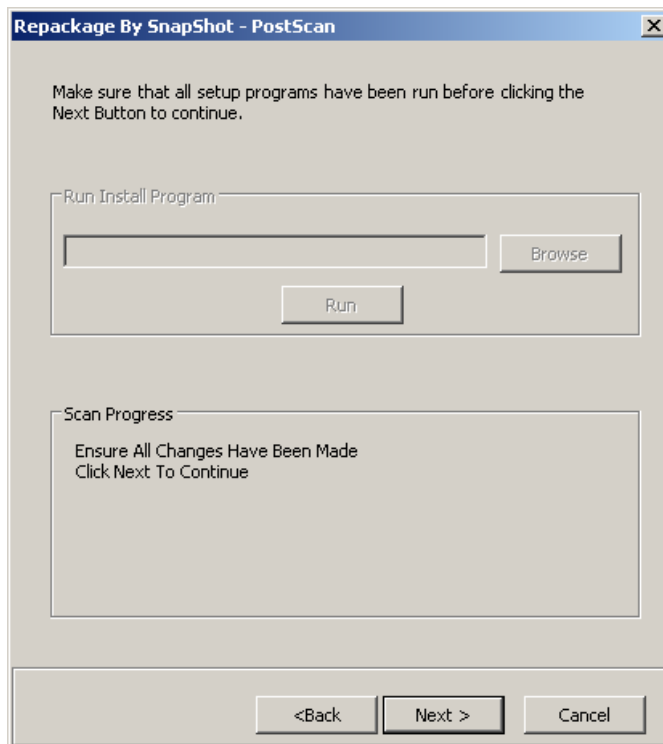
- Make sure that the correct drives have been specified in the Drives to Watch. In this case we only need drive c:\



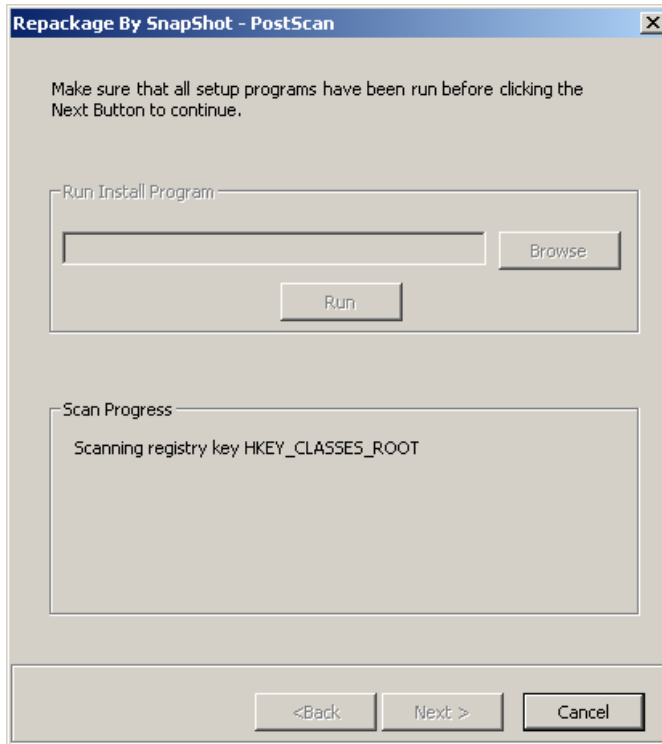
6. Press 'Start' and the repackaging process will begin.



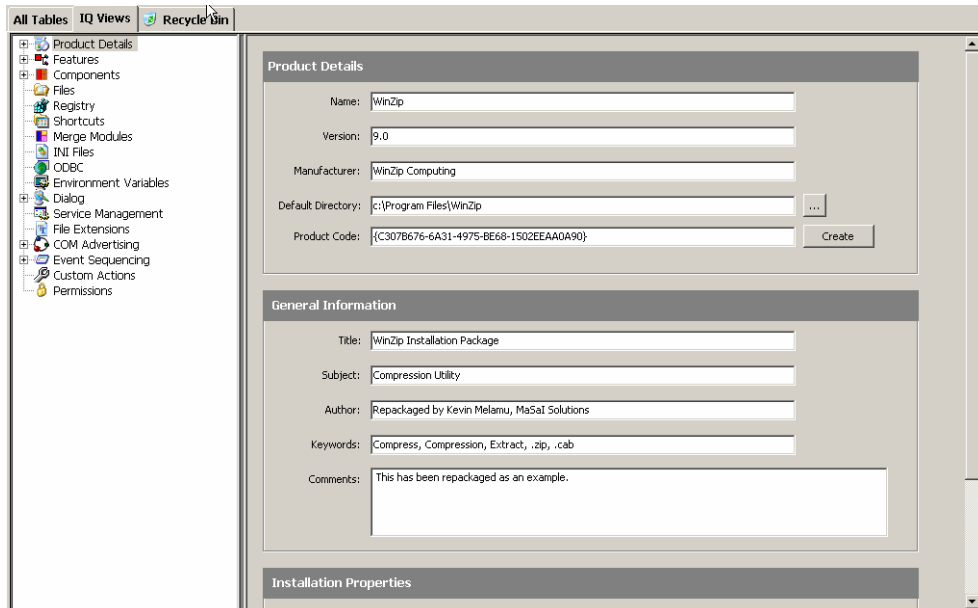
7. Once the initial scan has been done, go to the Repackaging machine and install WinZip selecting the options we want when we deploy it around our networks.



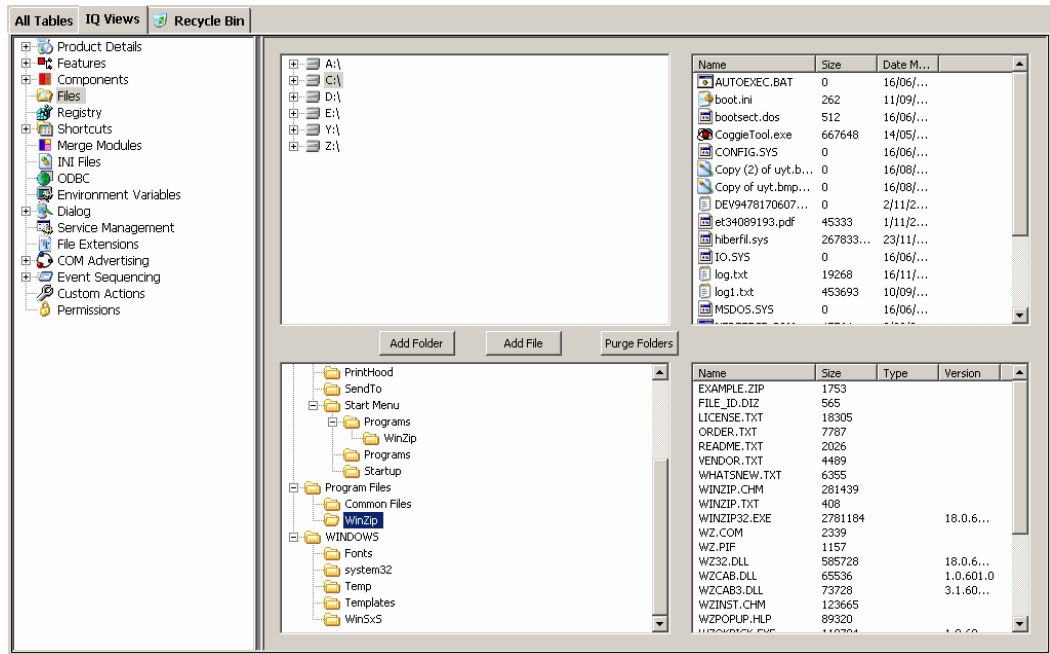
- After installing our application we press the 'Next' button back on our development machine and wait while the after scan is done and then items added to the project.



- First thing we do once the package has loaded is enter some details about the product in the *Product Details*.

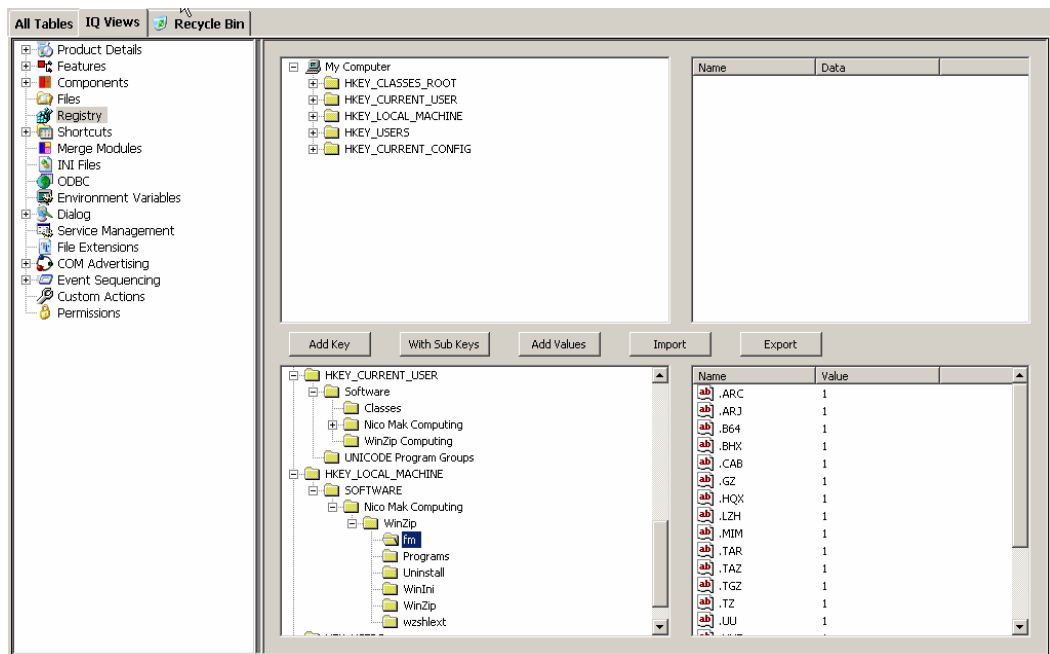


- Next we go to the file view and search through the project folders for files that should not be included. In this case none are found.



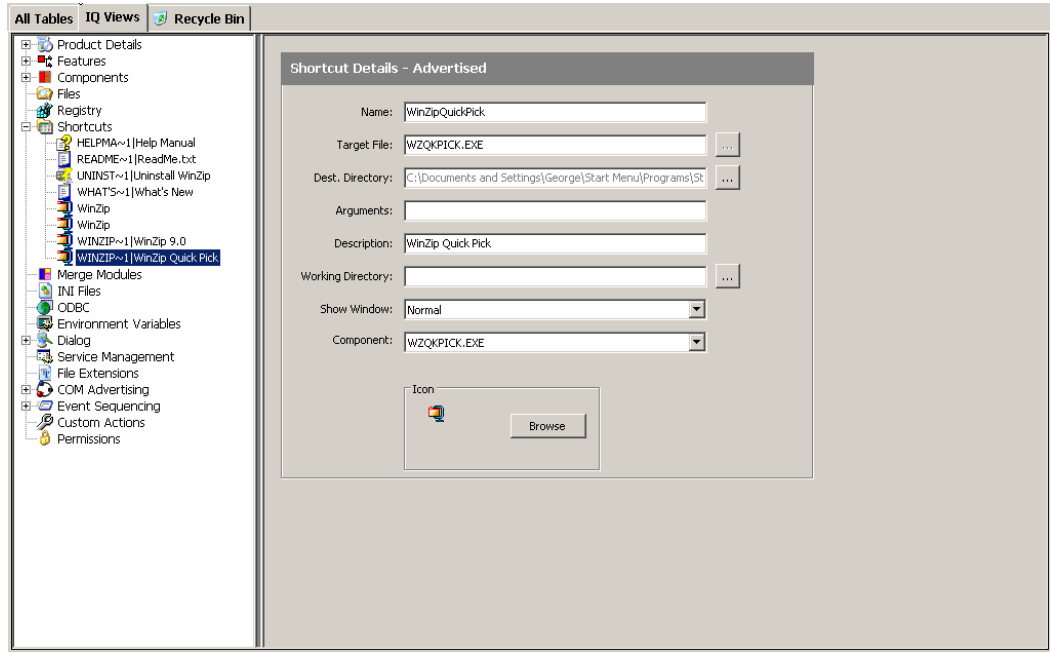
- Now do the same thing with the registry view.

Here we delete several keys under Windows\CurrentVersion\Explorer



12. Check the shortcuts.

Here we will delete the shortcut on the desktop as it is against our policy to have icons on the desktop. Also we delete the shortcuts to the text files WhatsNew.txt and ReadMe.txt



13. Next we save and then do a MCE Validation.

Number	Message	Table	Key	Column
33	Reg key Registry20 is used in an unsupported way. Extensions should be registered via the extension table.	Registry	Registry20;	Registry
33	Reg key Registry21 is used in an unsupported way. Extensions should be registered via the extension table.	Registry	Registry21;	Registry
33	Reg key Registry44 is used in an unsupported way. Shell extension verbs info should be registered via the Verb table.	Registry	Registry44;	Registry
33	Reg key Registry45 is used in an unsupported way. Shell extension verbs info should be registered via the Verb table.	Registry	Registry45;	Registry
33	Reg key Registry46 is used in an unsupported way. Shell extension verbs info should be registered via the Verb table.	Registry	Registry46;	Registry
33	Reg key Registry47 is used in an unsupported way. Shell extension verbs info should be registered via the Verb table.	Registry	Registry47;	Registry
36	Icon Bloat. Icon Icon1.ico is not used in the Class, Shortcut, or ProgID table. This adversely affects performance.	Icon	Icon1.ico;	Name
36	Icon Bloat. Icon Icon4.ico is not used in the Class, Shortcut, or ProgID table. This adversely affects performance.	Icon	Icon4.ico;	Name
36	Icon Bloat. Icon Icon6.ico is not used in the Class, Shortcut, or ProgID table. This adversely affects performance.	Icon	Icon6.ico;	Name
57	Component 'WZSEPE32.EXE' has both per-user and per-machine data with a per-machine KeyPath.	Component	WZSEPE32.EXE;	Component
64	The directory 'Programs' is in the user profile but is not listed in the RemoveFile table.	Directory	Programs;	Directory
64	The directory 'WinZip' is in the user profile but is not listed in the RemoveFile table.	Directory	WinZip;	Directory

14. We find 3 errors, the first can be fixed by moving the registry entry in HKEY\_CURRENT\_USER from the component WZSEPE32.EXE to CurrentUser.

The next 2 we fix by adding entries in the RemoveFile table.

Also we can delete the icons specified in the warning #36.

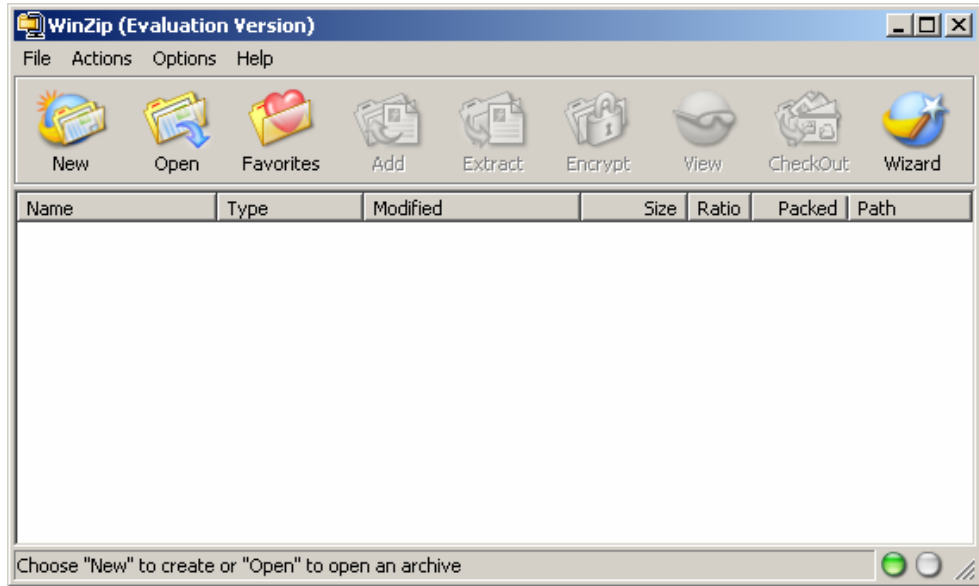
**Note:** How to go about fixing errors and warnings is beyond the scope of this document. What is important at this stage is the process of fixing them.

- Next we do a full ice validation to check there are no more errors which there are not.

Ice	Message
ICE33	Reg key Registry16 is used in an unsupported way. Extensions should be registered via the Extension table. This entry may overwrite a value created through that table.
ICE33	Reg key Registry16 is used in an unsupported way. Extensions should be registered via the Extension table. This entry may overwrite a value created through that table.
ICE33	Reg key Registry18 is used in an unsupported way. Extensions should be registered via the Extension table. This entry may overwrite a value created through that table.
ICE33	Reg key Registry13 is used in an unsupported way. Extensions should be registered via the Extension table. This entry may overwrite a value created through that table.
ICE33	Reg key Registry12 is used in an unsupported way. Extensions should be registered via the Extension table. This entry may overwrite a value created through that table.
ICE33	Reg key Registry10 is used in an unsupported way. Extensions should be registered via the Extension table. This entry may overwrite a value created through that table.
ICE33	Reg key Registry9 is used in an unsupported way. Extensions should be registered via the Extension table. This entry may overwrite a value created through that table.
ICE33	Reg key Registry4 is used in an unsupported way. Extensions should be registered via the Extension table. This entry may overwrite a value created through that table.
ICE33	Reg key Registry3 is used in an unsupported way. Extensions should be registered via the Extension table. This entry may overwrite a value created through that table.
ICE33	Reg key Registry2 is used in an unsupported way. Extensions should be registered via the Extension table. This entry may overwrite a value created through that table.
ICE33	Reg key Registry1 is used in an unsupported way. Extensions should be registered via the Extension table. This entry may overwrite a value created through that table.
ICE33	Reg key Registry9 is used in an unsupported way. ProgId should be registered via the ProgId table. This entry may overwrite a value created through that table.
ICE33	Reg key Registry22 is used in an unsupported way. CLSIDs should be registered via the Class table. This entry may overwrite a value created through that table.
ICE33	Reg key Registry25 is used in an unsupported way. CLSIDs should be registered via the Class table. This entry may overwrite a value created through that table.
ICE33	Reg key Registry28 is used in an unsupported way. CLSIDs should be registered via the Class table. This entry may overwrite a value created through that table.
ICE33	Reg key Registry31 is used in an unsupported way. CLSIDs should be registered via the Class table. This entry may overwrite a value created through that table.
ICE69	Mismatched component reference. Entry 'Registry113' of the Registry table belongs to component 'CurrentUser'. However, the formatted string in column 'Value' references file 'W2SEPE32

**NOTE** that the ICE33 relate to advertised COM entries in the registry table which we can ignore in this case as we are not advertising products. However we could get rid of many of these by running Tools>Extract/Convert Registry COM

- Now we are done so we go to our test machine and test.



## CREATING A WRAPPER INSTALLATION

This process is used to wrap installations which cannot be repackaged. Reasons installations should not be repackaged are if they are installing settings which differ on different machines, or if they are updating windows protected system files. Examples might include windows updates and security patches. Changes should not be captured but rather the original installation or patch should be wrapped in a Windows Installer .msi file for deployment.

1. First make sure that the installation can and should be installed by testing the vendor installation. Also during this step find out whether it can be uninstalled. Often installations of this type cannot easily be uninstalled.
2. Find out the correct command line parameters with which you wish to install and uninstall the vendor installation. Remember that you need to be able to do this silently.
3. Open IDS and from the start page select '*Repackage An Application For Deployment Around a Corporate Network*'
4. Select '*Wrap an Executable in an MSI*'
5. On the '*Create a Wrapper Install for An Executable*' screen
  - In the top field enter the name of the file to save the msi database to.
  - In the next field select the vendor's installation program which should be an executable file.
  - Enter the command line parameters to run on installation i.e. /quiet
  - If you wish to create an uninstall select the radio button '*Run the executable on uninstall*' and enter any command line parameters i.e. /quiet /uninstall
6. Press '*GO*' and the msi will be created and opened. Now you can add any other things you might need in the project such as files and registry entries.

**NOTE:** There will already be one registry entry in the project, this is just a dummy key which is required by Windows Installer as it must install 'something' and it must contain at least one component.

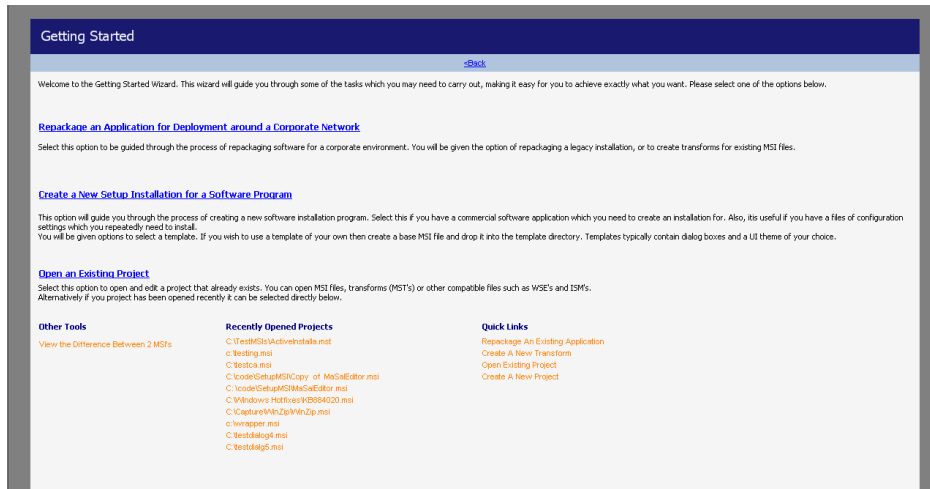
### Wrapper Installation Example

---

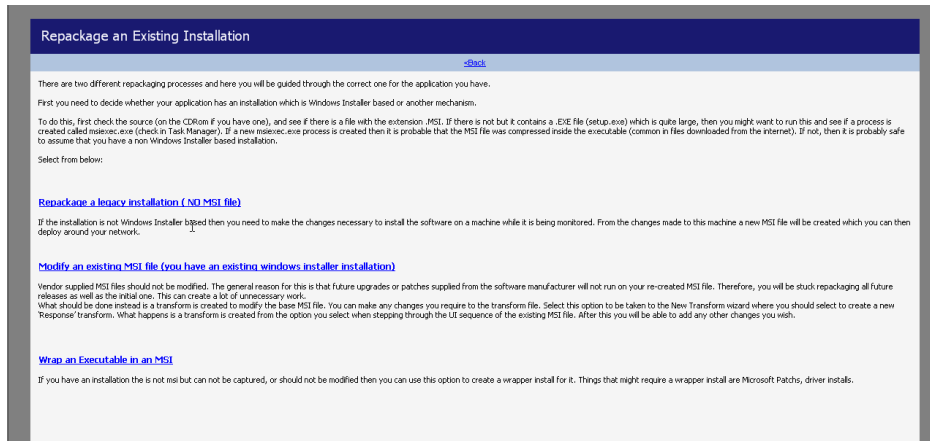
In this example we will repackage the Microsoft Update KB884020. When we download this from the Microsoft site it comes as an executable. Knowing that it is a Windows Update we decide not to repackage it, but rather to wrap it in a .msi to make for easy deployment.

1. Test the installation on the test machine using the command lines /quiet /norestart on install and /quiet /norestart /uninstall on uninstall. We got these command lines from the Microsoft website. As these seem to work well we can start the wrapping process.

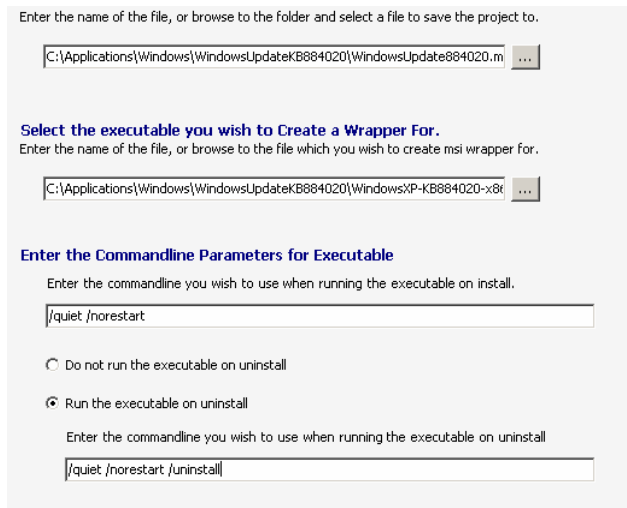
2. Run IDS and Select 'Repackage An Application For Deployment around a Corporate Network'



3. Then select 'Wrap an Executable in an MSI'



4. Fill in the required fields with the information gathered above.



5. Hit the 'GO' button and the project is created and opened.  
Enter the project details on the *Project Details* form.  
That is all the changes we need so we can save and then test.

The screenshot shows the 'Product Details' form in MSI Studio. It contains the following fields and values:

Product Details	
Name:	Windows Update KB804020
Version:	1.0
Manufacturer:	Microsoft
Default Directory:	<input type="text"/> ...
Product Code:	{81596E96-6370-4530-8CDA-F8FAFDF16AAD} <input type="button" value="Create"/>

General Information	
Title:	Windows Update KB804020
Subject:	Windows Update for XP SP2
Author:	Repackaged by Johan Lomu for MaSaI Solutions
Keywords:	Update, Hotfix, XP
Comments:	<input type="text" value="This package has been used as an example of how to create a wrapper installation for a Windows Update."/>

Installation Properties	
-------------------------	--

If you have any questions or suggestions about using MSI Studio, please visit the [ScriptLogic website](http://www.scriptlogic.com).