Bear Creek Package Development

Bob Kelly

BCO Package Development Documentation

The following document explains the custom processes implemented into Wise Package Studio (WPS) to support operations for package creation at Bear Creek Corporation.

Creating a New Project

It is important to take care when creating a new project, as the information entered here is used to control the package creation process in identifying files and folders and how details will be provided for the creation of files generated by the process.

Project Name

Enter an identifying name for the project. The name chosen will not impact the resulting package, but should uniquely identify the application and version for easy identification.

Example: Microsoft Visio 5.0.2

Project Directory

The project directory folder is dynamically generated and does not require modification. This is the location where the package sits during the development cycle (when completed the needed files are placed in the "Available Packages[1]" folder).

Status

The Status of the package at the time a project is created may be set to open or closed:

- Open – Package is ready for development
- Close – Package is on hold at this time, use "Notes" field to identify reason for "closed" status.

Product Vendor

Enter the name of the company that provides the application being packaged. If the company name already exists in the list provided, please choose it so that packages may be consistent with one another. To avoid the desire to modify a company name after it has been entered; please take care to ensure proper case and spacing when entering new company names.

Example: Microsoft Corporation

---

[1] \\Myidtc01\sharepoint$\Available Packages

Application Name

The application name should consist of the vendor and application name only (details such as the version or edition of an application is specified in the Package Name field.)

Example: Microsoft Visio

Package Name

The package name should uniquely identify the package being developed for this application. Make use of both the name and version number for the application being packaged.

Example: Visio 5.0.2

File Name

The filename is used wherever a file is generated by the process as: <file name>.<extension>

It is best to avoid spaces in this field and the name should help to identify the specific package you are working on. By default the filename is identical to the project name.

Example: VISIO502

Vendor Package

For command line or repackaged installations: enter (or browse to) the setup executable provided by the vendor for installation of this application.

For vendor-provided Windows Installer packages (MSI): enter (or browse to) the MSI file for which you will be creating a transform (MST).

It is important to note that this field will dictate what files are copied to the network as the source for the installation. The folder where the specified file exists, and its subdirectories, will be copied to the FILES subfolder of the package for use by command line installations. The same goes for Windows Installer (MSI) packages, but for MSI packages an admin install will be performed to place the MSI source files (and the MSI itself) in a subfolder named MSI. Like the FILES subfolder, this is where the executable wrapper for the package will point (as a relative path) for its installation files and the copy that remains in the FILES subfolder exists for archival purposes so that an original copy of the source files remain available for future use.

Repackaged Applications

For an installation to be repackaged, provide the default executable to be run for the repackaging process.

MSI Setups Requiring a Transform

For a vendor-provided MSI, provide the MSI to be launched by InstallTailor in order to create a transform (MST).

Command Line Installations

For command line installations, the provided executable will be used to populate some details for the BCO EXE Wrapper Generator.

Process

This choice dictates the steps provided in the process for packaging this application. *If the setup is a vendor-provided MSI, choose "BCO Vendor Provided MSI". Otherwise, choose "BCO Command Line Installation".*

The BCO Command Line Installation process includes research steps to determine if a command line installation is available. If it is determined that the setup should be repackaged- the process may be changed to "BCO Repackaged Legacy Setup as MSI".

BCO Command Line Installation

If the setup is not a Windows Installer setup, this process is designed to help generate a package using vendor-provided command line installation support, or if deemed unavailable, the process may be changed to “BCO Repackaged Legacy Setup as MSI”.

Application Integration

The following steps in the process revolve around the creation of the package and its documentation.

Investigate Command Line Options

The Run link for this process task starts a Google search for information available on the creation of the package. Check product documentation, any vendor provided knowledge base and the AppDeploy website for documented options for automated installation.

If you cannot determine the availability of a command line installation method, you may choose “Project” from the menubar’s “Edit” drop-down menu to update the process for the current project.

```
Manual Processing: Simply search any vendor knowledge bases and the
AppDeploy package knowledge base for deployment details for the
application.
```

Document Package

A new Word or text document (dependant upon the availability of Word on the local system) is also generated for the initial documentation of the package.

If available, document any contact information for the point of contact (POC) you have for this application. Whenever available a POC should be established as a authority for validating installation decisions, and operational use of the application. If such an individual can be specified, this documentation should reflect it.

To begin with, perform a manual installation of the application and take note of the installation software used (InstallShield, Wise, InnoSetup, etc.) as this may help to expose command line deployment options.

During the manual installation, document the choices made. Any installation decisions should be provided by the individual or group requesting the application. It is best not to accept “Just Accept All Defaults” as installation instructions unless there are no decisions to be made in the installation wizard. For example, if a Typical, Minimal, Custom installation choice is available- get a definitive answer as to what components of the installation are to be included in your deployment package in order to avoid having to restart the process due to a bad assumption.

Will you use a command line or repackage? Justify the reason for your decision as it pertains to this application in the documentation. This can be valuable information in the future when a new version is made available, a patch is offered by the vendor or if problems surface in the deployment and the package must be examined by another user.

```
Manual Processing: Create a TXT or HTML formatted document in the
root of the package folder with the name
<filename>_Documentation.<ext> and include research, installation,
packaging and testing notes for the application. For a template, you
may copy a documentation file from another project package folder.
```

Capture Changes

Start this process on a clean system that does not contain the application being packaged- preferably a system with as little installed as possible.

Import changes made by setup into Software Manager for use in troubleshooting. Even though the application will be installed from the command line, this step exists to generate a list of changes this command line installation will be implementing to target systems. The list of file and registry entries detected in this step will be imported into the Software Manager database for use by Conflict Manager. Conflict Manager is not a part of the package development process, but may be leveraged as a valuable tool in the troubleshooting of packages in the future.

Accept the default of utilizing both Snapshot and InstallMonitor technologies for the capture process. The decision to make use of a previously created initial snapshot will depend upon your situation- if you are working in a VM where you know nothing to have changed since the previous snapshot was taken, you may make use of it. Otherwise, disregard the previous snapshot and rescan the system. If you are unsure, disregard the previous snapshot and rescan the system.

When presented with the setup execution dialog, you may either provide the setup.exe and silent command line options or just the setup.exe (where you would interactively complete the installation wizard, if any). When possible, make use of the command line installation switches you intend to use with your package. In fact, this step need not be performed from the installation dialog. You may use the Start|Run option or File Manager to trigger your command line installation as well and the result will be the same.

Choose “Next” once the application has been installed and allow the “after” snapshot to proceed. Once completed, ensure the application name and version is properly entered before completing the SetupCapture wizard.

```
Manual Processing: Run SetupCapture from the Wise Package Studio
Workbench. If prompted, specify <project folder>\<filename>.wsi as
the package to be created and check the option to have source files
copied up to the repository using relative paths.
```

Generate Executable Wrapper

The command line switches to be used for this package must be specified here for use by the executable wrapper. Other details are filled in based on Wise Project specifications; however they should be reviewed for accuracy- particularly the “version” field.

This phase also copies the files in the directory where the vendor file is located, along with any subfolders that may exist to the “FILES” subfolder for the current project.

This is a custom tool, for details see Customization Details later in this document.

```
Manual Processing: Copy “EXETemplate.ipf” to the project folder and rename
it to <filename>.ipf. Next, create a file in the %TEMP% folder named
WrapperValues.dat and using notepad, add the following entries to this INI
formatted file:
```

Defined	**Example**

[WrapperValues]
PACKAPPTITLE=Package Title
PACKVENDOR=Package Vendor
PACKVERSION=Package Version
EXEFILENAME=Executable Filename
EXEFILEARGS=Switches for silent install
MIFCODE=Unique 7-digit code
	[WrapperValues]
	PACKAPPTITLE="Visio"
	PACKVENDOR="Microsoft"
	PACKVERSION="5.0"
	EXEFILENAME="./FILES/setup.exe"
	EXEFILEARGS=”-silent -noboot”
	MIFCODE="VIS5MIC"

```
Note that all value names must be in quotes.

Next, open your IPF file from the project folder using the Wise Script
Editor tool from the Wise Package Workbench and compile the executable. At
this point, the WrapperValues.dat file may be discarded or copied to the
project folder in case it is needed for manual processing again in the
future.
```

Review Capture Results

Despite the fact that this captured installation will not be used for deployment purposes, some cleanup is necessary before proceeding. Any temporary files or items that clearly do not belong to the installation should be removed in the Windows Installer Editor.

Focus on the File and Registry views of the Windows Installer Editor, choosing “Hide Empty Folders” from the context menu to more easily identify your captured contents.

Cleaning up this capture information provides two key benefits:

- It will avoid errors that may occur trying to import temporary/non-existent files into the Software Manager database and file repository.
- It will avoid extraneous conflict information during Conflict Management operations that may be leveraged when troubleshooting this or other packages in the future.

```
Manual Processing: Open the WSI captured earlier in the process using
the Windows Installer Editor from the Wise Package Studio Workbench
and review the package for items as described above. When complete,
save and exit the Windows Installer Editor.
```

Package Testing

No matter what the installation method, proper testing and documentation of the installation is extremely important. In order to streamline the installation process, ICE validation and Conflict Management steps have been eliminated and may instead be leveraged only if problems in testing of the package surface.

Implement Security Updates

If you are already aware of special security requirements for this application, implement/automate those changes prior to installation testing. If you are unaware of any requirements at this stage, they may surface in the next step of the process (Test Installation).

Testing the use of all application operations that may result in changes to the system, or in the generation of new files. Common security changes include:

Requirement to relax NTFS permissions on application folder. For example:
%PROGRAMFILES%\<Application>
%PROGRAMFILES%\<Application>\Data
%PROGRAMFILES%\<Application>\Work

Requirement to relax registry permissions on application machine subkeys. For example:
HKEY_LOCAL_MACHINE\Software\<Application>

FileMon and RegMon tools from SysInternals can be very helpful in identifying files and keys that may need to have security restrictions relaxed for the application:

http://www.sysinternals.com/ntw2k/utilities.shtml#monitor

Test Installation

Test your executable wrapper on a machine containing the expected baseline application set (a typical configuration reflecting an operational system). Run through your documented test procedures or notify the POC for the package that the application is ready for testing and invite them to the lab to test the package as installed by the executable wrapper. Alternatively, you may deploy the executable wrapper to the POCs computer for testing.

It is essential that any application testing be performed:

- Using a non-administrative account
- By someone who understands the application and how its operational use

If problems are encountered, you may make use of the Conflict Manager tool to identify applications that share files with your package (this will help to narrow the focus of your QA testing).

Move Capture Results to Repository

This step simply takes the changes you have captured in the "Capture Changes Made by Installation" step and introduces them to the Software Manager database. At this stage, the application will be listed as a "New Application" and will not include file details for use by Conflict Manager (this is rectified in the following step).

```
Manual Processing: Launch the Package Distribution tool from the Wise
Package Studio Workbench, enter the full path to the WSI file created
in your project folder with SetupCapture and press Next. Choose
"Distribute to Share Point Directory" and next to begin the move.
```

Import Changes Into Software Manager

This step copies the files that were discovered by the SetupCapture results and places them in the Wise Repository (a single-instance store located in the Wise Share Point subdirectory named "000". The status of the application is updated to "Under Development".

```
Manual Processing: Launch Software Manager from the Wise Package
Studio Workbench, press the "Import" button on the toolbar and choose
"Import a single file into the Software Manager database". Browse to
the WSI project file you created and cleaned earlier in this process
and press Next to begin the import process.
```

Release Management

Proceed to release management steps once the package is deemed ready for deployment. This part of the process makes the required elements of the package (the wrapper and associated source files) available to administrators for deployment or manual execution on the network.

Make Package Available

This step copies the executable wrapper to:

[SharePoint]\Available Packages\[ApplicationName]\[PackageName]

It also copies the FILES subdirectory used by command line installations, or the MSI subdirectory used by repackaged and transformed installations, used by the executable wrapper.

The wrapper and source files are referenced using a relative path so they may be executed directly or copied together as a distribution point source.

```
Manual Processing: Copy the EXE Wrapper to
<sharepoint>\<application>\<package>\<filename>.exe and copy the
FILES subfolder from the project folder to
<sharepoint>\<application>\<package>\FILES

Launch Software Manager from the Wise Package Studio Workbench,
select the package being completed and update its status to
“Complete”.
```

Repackaged Legacy Setup as MSI

This process will walk you through the steps needed to repackage an application using Windows Installer technology.

Uses SetupCapture to generate WSI, Edits WSI in Windows Installer Editor, Compiles the project to an MSI, establishes admin installation point for testing, creates executable wrapper and copies package to Available Packages folder for deployment.

Application Integration

The following steps in the process revolve around the creation of the package and its documentation.

Document Package

A new Word or text document (dependant upon the availability of Word on the local system) is also generated for the initial documentation of the package.

If available, document any contact information for the point of contact (POC) you have for this application. Whenever available a POC should be established as an authority for validating installation decisions, and operational use of the application. If such an individual can be specified, this documentation should reflect it.

To begin with, perform a manual installation of the application and take note of the installation software used (InstallShield, Wise, InnoSetup, etc.) as this may help to expose command line deployment options.

During the manual installation, document the choices made. Any installation decisions should be provided by the individual or group requesting the application. It is best not to accept “Just Accept All Defaults” as installation instructions unless there are no decisions to be made in the installation wizard. For example, if a Typical, Minimal, Custom installation choice is available- get a definitive answer as to what components of the installation are to be included in your deployment package in order to avoid having to restart the process due to a bad assumption.

Will you use a command line or repackage? Justify the reason for your decision as it pertains to this application in the documentation. This can be valuable information in the future when a new version is made available, a patch is offered by the vendor or if problems surface in the deployment and the package must be examined by another user.

```
Manual Processing: Create a TXT or HTML formatted document in the
root of the package folder with the name
<filename>_Documentation.<ext> and include research, installation,
packaging and testing notes for the application. For a template, you
may copy a documentation file from another project package folder.
```

Create Package

This step in the process utilizes SetupCapture to create a Windows Installer Setup for the application. The detailed use of this tool is provided in WPS.

```
Manual Processing: Run SetupCapture from the Wise Package Studio
Workbench. If prompted, specify <project folder>\<filename>.wsi as
the package to be created and check the option to have source files
copied up to the repository using relative paths.
```

Edit Package

This step opens the created Windows Installer project in the Windows Installer Editor, where you may review and update the package as necessary.

Ensure the package is “clean”. If available, utilize PackageCleaner to identify elements that may not belong. In addition keep in mind that most all applications place elements in the following locations:

%PROGRAMFILES%\<application name>
%PROGRAMFILES%\<company name>\<application name>
%PROGRAMFILES%\Common Files\<company name>
%USERPROFILE%\Application Data

HKLM\Software\<application name>
HKLM\Software\<company name>\<application name>
HKEY_CLASSES_ROOT

Note: it should be unnecessary to update any changes detected in HKEY_CLASSES_ROOT. These updates are typically handled automatically when adding or removing DLL or OCX files form your package or when adding or removing file associations using the Windows Installer Editor GUI.

```
Manual Processing: Open the WSI using the Windows Installer Editor
from the Wise Package Studio Workbench and review the package for
items as described above. When complete, save and compile your WSI at
this time. This generates an MSI setup, allowing you to bypass the
next step "Compile Setup to Create MSI".
```

Compile Setup to Create MSI

This step can be performed within the Windows Installer Editor when you are done cleaning up the Windows Installer Project (WSI) by pressing the "Compile" button, or by running this step in the process which will do the same. The WSI project is compiled into MSI format creating <filename>.msi.

Generate MSI Source Files

This step in the process performs an administrative installation of the MSI to the "<project folder>\MSI" folder. This essentially places the MSI and its required, decompressed source files in the subdirectory that will be used by the executable wrapper to trigger the command line installation of the MSI.

Files are stored in this way for two key reasons: First it removes limitations of repackaged MSI files that are in place concerning the number of files and the total size of the files. Second, when client systems need to pull files from the network, it is faster to do so from decompressed external files than it is from compressed source (internal to the MSI). For the process to maintain consistency between packages, this step is performed for repackaged and vendor-provided MSI packages alike.

```
Manual Processing: Perform an administrative installation using the
following command line:

msiexec /a <sharepoint>\<project folder>\<filename>.msi
TARGETDIR="<sharepoint>\<project folder>\MSI" /qb+
```

Generate Executable Wrapper

The contents of the Executable Wrapper dialog should be filled out automatically. Review the entries for accuracy- particularly the "version" field.

This phase also copies the files in the directory where the vendor file is located, along with any subfolders that may exist to the "MSI" subfolder for the current project.

This is a custom tool, for details see Customization Details later in this document.

```
Manual Processing: Copy “EXETemplate.ipf” to the project folder and rename
it to <filename>.ipf. Next, create a file in the %TEMP% folder named
WrapperValues.dat and using notepad, add the following entries to this INI
formatted file:
```

Defined	**Example**

[WrapperValues]
PACKAPPTITLE=Package Title
PACKVENDOR=Package Vendor
PACKVERSION=Package Version
EXEFILENAME=Executable Filename
EXEFILEARGS=Switches for silent install
MIFCODE=Unique 7-digit code
	[WrapperValues]
	PACKAPPTITLE="Visio"
	PACKVENDOR="Microsoft"
	PACKVERSION="5.0"
	EXEFILENAME="msiexe.exe"
	EXEFILEARGS=”./msi/mymsi.msi /qb”
	MIFCODE="VIS5MIC"

```
Note that all value names must be in quotes.

Next, open your IPF file from the project folder using the Wise Script
Editor tool from the Wise Package Workbench and compile the executable. At
this point, the WrapperValues.dat file may be discarded or copied to the
project folder in case it is needed for manual processing again in the
future.
```

Package Testing

No matter what the installation method, proper testing and documentation of the installation is extremely important. In order to streamline the installation process, ICE validation and Conflict Management steps have been eliminated and may instead be leveraged only if problems in testing of the package surface.

Implement Security Updates

If you are already aware of special security requirements for this application, implement/automate those changes prior to installation testing. If you are unaware of any requirements at this stage, they may surface in the next step of the process (Test Installation).

Testing the use of all application operations that may result in changes to the system, or in the generation of new files. Common security changes include:

Requirement to relax NTFS permissions on application folder. For example:
%PROGRAMFILES%\<Application>
%PROGRAMFILES%\<Application>\Data
%PROGRAMFILES%\<Application>\Work

Requirement to relax registry permissions on application machine subkeys. For example:
HKEY_LOCAL_MACHINE\Software\<Application>

FileMon and RegMon tools from SysInternals can be very helpful in identifying files and keys that may need to have security restrictions relaxed for the application:

http://www.sysinternals.com/ntw2k/utilities.shtml#monitor

Test Package

Test your executable wrapper on a machine containing the expected baseline application set (a typical configuration reflecting an operational system). Run through your documented test procedures or notify the POC for the package that the application is ready for testing and invite them to the lab to test the package as installed by the executable wrapper. Alternatively, you may deploy the executable wrapper to the POCs computer for testing.

It is essential that any application testing be performed:

- Using a non-administrative account
- By someone who understands the application and how its operational use

If problems are encountered, you may make use of the Conflict Manager tool to identify applications that share files with your package (this will help to narrow the focus of your QA testing).

Move Capture Results to Repository

This step simply takes the changes you have captured in the "Capture Changes Made by Installation" step and introduces them to the Software Manager database. At this stage, the application will be listed as a "New Application" and will not include file details for use by Conflict Manager (this is rectified in the following step).

```
Manual Processing: Launch the Package Distribution tool from the Wise
Package Studio Workbench, enter the full path to the WSI file created
in your project folder with SetupCapture and press Next. Choose
"Distribute to Share Point Directory" and next to begin the move.
```

Import Changes Into Software Manager

This step copies the files that were discovered by the SetupCapture results and places them in the Wise Repository (a single-instance store located in the Wise Share Point subdirectory named “000”. The status of the application is updated to “Under Development”.

```
Manual Processing: Launch Software Manager from the Wise Package
Studio Workbench, press the “Import” button on the toolbar and choose
“Import a single file into the Software Manager database”. Browse to
the WSI project file you created and cleaned earlier in this process
and press Next to begin the import process.
```

Release Management

Proceed to release management steps once the package is deemed ready for deployment. This part of the process makes the required elements of the package (the wrapper and associated source files) available to administrators for deployment or manual execution on the network.

Make Package Available

This step copies the executable wrapper to:

[SharePoint]\Availabile Packages\[ApplicationName]\[PackageName]

It also copies the MSI subdirectory for use by the executable wrapper.

The wrapper and source files are referenced using a relative path so they may be executed directly or copied together as a distribution point source.

```
Manual Processing: Copy the EXE Wrapper to
<sharepoint>\<application>\<package>\<filename>.exe and copy the
MSI subfolder from the project folder to
<sharepoint>\<application>\<package>\MSI

Launch Software Manager from the Wise Package Studio Workbench,
select the package being completed and update its status to
“Complete”.
```

BCO Vendor Provided MSI

This process is used to customize an existing Windows Installer package by creating a transform file.

Creates a transform for a vendor-provided MSI, imports changes into Software Manager, creates executable wrapper and copies package to Available Packages folder for deployment.

Application Integration

The following steps in the process revolve around the creation of the package and its documentation.

Document Package

A new Word or text document (dependant upon the availability of Word on the local system) is also generated for the initial documentation of the package.

If available, document any contact information for the point of contact (POC) you have for this application. Whenever available a POC should be established as an authority for validating installation decisions, and operational use of the application. If such an individual can be specified, this documentation should reflect it.

To begin with, perform a manual installation of the application and take note of the installation software used (InstallShield, Wise, InnoSetup, etc.) as this may help to expose command line deployment options.

During the manual installation, document the choices made. Any installation decisions should be provided by the individual or group requesting the application. It is best not to accept "Just Accept All Defaults" as installation instructions unless there are no decisions to be made in the installation wizard. For example, if a Typical, Minimal, Custom installation choice is available- get a definitive answer as to what components of the installation are to be included in your deployment package in order to avoid having to restart the process due to a bad assumption.

Will you use a command line or repackage? Justify the reason for your decision as it pertains to this application in the documentation. This can be valuable information in the future when a new version is made available, a patch is offered by the vendor or if problems surface in the deployment and the package must be examined by another user.

```
Manual Processing: Create a TXT or HTML formatted document in the
root of the package folder with the name
<filename>_Documentation.<ext> and include research, installation,
packaging and testing notes for the application. For a template, you
may copy a documentation file from another project package folder.
```

Generate MSI Source Files

This step in the process performs an administrative installation of the MSI to the "<project folder>\MSI" folder. This essentially places the MSI and its required, decompressed source files in the subdirectory that will be used by the executable wrapper to trigger the command line installation of the MSI.

This is done primarily for consistency with the repackaged MSI process. However, this also provides greater speed when accessing files from the network for repair and reinstallation operations. If desired, you may simply create a project subfolder named "MSI" and place the single MSI file in this directory.

```
Manual Processing: Perform an administrative installation using the
following command line:

msiexec /a <sharepoint>\<project folder>\<filename>.msi
TARGETDIR="<sharepoint>\<project folder>\MSI" /qb+
```

Create Transform

This step utilizes InstallTailor to create a response transform for the specified MSI package (vendor-provided). Simply respond to the installation wizard as the application is to be deployed, and the options chosen will be gathered and created as "<project folder>\MSI\<Vendor MSI Filename>.mst".

InstallTailor is documented in detail within the WPS help file.

```
Manual Processing: Launch InstallTailor from the Wise Package Studio
Workbench and choose the vendor provided MSI when prompted.
```

Customize Transform

The transform file is opened in the Windows Installer Editor for further edits. If no further modifications to the package are necessary, this step need not be performed. However, if the Windows Installer package provided by the vendor requires any modification at all, such changes may be implemented here. These changes may include, but are not limited to:

- Moving/Adding/Deleting Shortcuts
- Moving/Adding/Deleting Files
- Moving/Adding/Deleting Registry Entries

```
Manual Processing: Open the MST using the Windows Installer Editor
from the Wise Package Studio Workbench and perform any additional
customization you desire here.
```

Generate Executable Wrapper

The contents of the Executable Wrapper dialog should be filled out automatically. Review the entries for accuracy- particularly the “version” field.

This phase also copies the files in the directory where the vendor file is located, along with any subfolders that may exist to the “MSI” subfolder for the current project.

This is a custom tool, for details see Customization Details later in this document.

Manual Processing: Copy “EXETemplate.ipf” to the project folder and rename it to <filename>.ipf. Next, create a file in the %TEMP% folder named WrapperValues.dat and using notepad, add the following entries to this INI formatted file:

Defined	**Example**
[WrapperValues]	[WrapperValues]
PACKAPPTITLE=Package Title	PACKAPPTITLE="Visio"
PACKVENDOR=Package Vendor	PACKVENDOR="Microsoft"
PACKVERSION=Package Version	PACKVERSION="5.0"
EXEFILENAME=Executable Filename	EXEFILENAME="msiexe.exe"
EXEFILEARGS=Switches for silent install	EXEFILEARGS=”./msi/mymsi.msi TRANSFORMS=”./msi/mytrasnform.msi”/qb”
MIFCODE=Unique 7-digit code	MIFCODE="VIS5MIC"

Note that all value names must be in quotes.

Next, open your IPF file from the project folder using the Wise Script Editor tool from the Wise Package Workbench and compile the executable. At this point, the WrapperValues.dat file may be discarded or copied to the project folder in case it is needed for manual processing again in the

Package Testing

No matter what the installation method, proper testing and documentation of the installation is extremely important. In order to streamline the installation process, ICE validation and Conflict Management steps have been eliminated and may instead be leveraged only if problems in testing of the package surface.

Implement Security Updates

If you are already aware of special security requirements for this application, implement/automate those changes prior to installation testing. If you are unaware of any requirements at this stage, they may surface in the next step of the process (Test Installation).

Testing the use of all application operations that may result in changes to the system, or in the generation of new files. Common security changes include:

Requirement to relax NTFS permissions on application folder. For example:
%PROGRAMFILES%\<Application>
%PROGRAMFILES%\<Application>\Data
%PROGRAMFILES%\<Application>\Work

Requirement to relax registry permissions on application machine subkeys. For example:
HKEY_LOCAL_MACHINE\Software\<Application>

FileMon and RegMon tools from SysInternals can be very helpful in identifying files and keys that may need to have security restrictions relaxed for the application:

http://www.sysinternals.com/ntw2k/utilities.shtml#monitor

Test Package

Test your executable wrapper on a machine containing the expected baseline application set (a typical configuration reflecting an operational system). Run through your documented test procedures or notify the POC for the package that the application is ready for testing and invite them to the lab to test the package as installed by the executable wrapper. Alternatively, you may deploy the executable wrapper to the POCs computer for testing.

It is essential that any application testing be performed:

- Using a non-administrative account
- By someone who understands the application and how its operational use

If problems are encountered, you may make use of the Conflict Manager tool to identify applications that share files with your package (this will help to narrow the focus of your QA testing).

Move MSI Contents to Repository

This step simply takes the changes you have captured in the "Capture Changes Made by Installation" step and introduces them to the Software Manager database. At this stage, the application will be listed as a "New Application" and will not include file details for use by Conflict Manager (this is rectified in the following step).

Import Changes Into Software Manager

This step copies the files that were discovered by the SetupCapture results and places them in the Wise Repository (a single-instance store located in the Wise Share Point subdirectory named "000". The status of the application is updated to "Under Development".

Release Management

Proceed to release management steps once the package is deemed ready for deployment. This part of the process makes the required elements of the package (the wrapper and associated source files) available to administrators for deployment or manual execution on the network.

Make Package Available

This step copies the executable wrapper to:

[SharePoint]\Availabile Packages\[ApplicationName]\[PackageName]

It also copies the MSI subdirectory for use by the executable wrapper.

The wrapper and source files are referenced using a relative path so they may be executed directly or copied together as a distribution point source.

Customization Details

The custom tools were developed using the KiXtart scripting language. For those requiring a GUI, the freeware KiXforms DLL is used. Finally, the script is packaged as an executable using Admin Script Editor for use by WPS.

The tools and the source code for each are provided at:

\\Myidtc01\sharepoint$\Workbench\Custom

KiXtart: www.kixtart.org
KiXforms: www.kixforms.com
Admin Script Editor: www.adminscripteditor.com

Executable Wrapper Generation

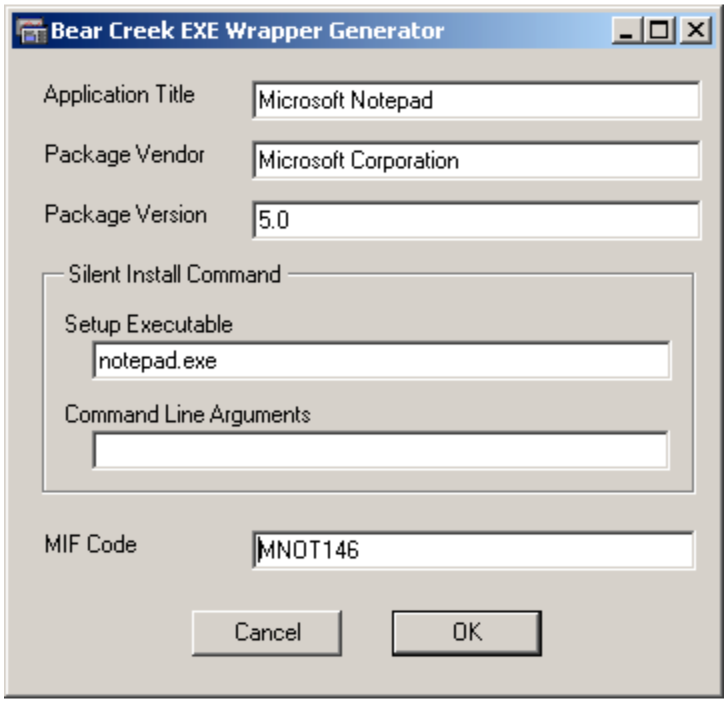
The EXE Wrapper is created using a custom template and Wise Script package. The information used to identify the application, version, vendor, etc. are passed from WPS to a custom tool that generates a data file read by the ipf template.

In order to dynamically dictate the contents of a package, all identifying information is supplied as variables. The variable values are loaded from an INI file that is generated by the Executable Wrapper utility, which is included in the wrapped executable. During installation, the INI file is installed to %TEMP%, read to populate variables and then deleted by the package (so nothing is left behind).

The ipf template is located at:
\\Myidtc01\sharepoint$\Templates\EXETemplate.ipf

This IPF file is compiled to generate the executable wrapper.

The below values are populated as best can be determined using the information supplied. For command line installations, the command line arguments must be entered manually- otherwise everything should fill in accurately, but should be reviewed before pressing “OK”.


Bear Creek EXE Wrapper Generator

Application Title	Microsoft Notepad
Package Vendor	Microsoft Corporation
Package Version	5.0

Silent Install Command

Setup Executable	notepad.exe
Command Line Arguments	

MIF Code	MNOT146

Values

- Application Name: Obtained from WPS project details.
- Package Vendor: Obtained from WPS project details.
- Package Version: Stripped from end of WPS project field for Package Name (which should end with the product version number)
- Setup Executable:
  - For command line installations: the setup.exe to which you will pass the silent switches
  - For Windows Installer installations: msiexec.exe
- Command Line Arguments
  - For command line installations: the switches passed to the specified setup executable for silent installation of the application
  - For Windows Installer installations: /i (for install), the relative path to the MSI, and any property values you wish to set for the installation. For vendor provided installations, the TRANSFORMS property should be used to specify the relative path to the create MST.
- MIF Code: The first character of the vendor name, the first three characters of the package name and the Julian date. This combination should be unique on its own, but if you know the

code to already be in use- update this to a unique value for the package.

Package Research Tool

Uses Google to start user on a search for information regarding command line installation of the project application. The launcher will attempt to focus on the successful load of an Internet Explorer search window every .5 seconds for ten seconds; if it is unable to perform the action (Internet access is unavailable), the process is aborted.

Package Documentation Tool

Creates a template based on Wise project details as an HTML file named <filename>_Documentation.html. The file is then launched in Word for manual updates.

If Word is found not to exist on the local workstation, a text file is created as <filename>_Documentation.txt. The file is then launched in WordPad for manual updates.

MST Launcher Tool

This tool simply calls InstallTailor (create) or Windows Installer Editor (edit) when attempting to work with the vendor provided MSI name (not available as a variable within WPS).

Reference

Project Files

There are a number of files and file extensions associated with the package at any phase during its development. The following chart is provided to help explain these various elements of the overall package as it pertains to the Wise Package Studio process.

<filename>.html	Documentation for the package, created using the provided template. This format is used when Word is available on the local system.
<filename>.txt	Documentation for the package, created using the provided template. This format is used when Word is not available on the local system.
Ipf	Uncompiled Wise/SMS Installer Script file for executable wrapper
-installation changes.htm	Changes detected during SetupCapture process (for documentation purposes only)
-installation sequence.txt	Lists changes made during captured setup (for documentation purposes only)
log	This file is generated by SetupCapture and may be disregarded
pdf	SMS Package Definition File. Not needed (executable already runs silent and does not require switches).
Que	Index file for the single instance store used by the Wise repository identifying the files that belong to this package.
wsi	Wise project file for Windows Installer setup (when compiled results in MSI). For command line installations, this project need not be compiled into MSI format as it is captured only for purposes of importing changes to the Software Manager Database.

The processes are stored in the Wise WorkBench SQL database and have also been exported as files to:

\\Myidtc01\sharepoint$\Workbench\Custom