Using Application Dependency Analysis to Enable Your Application on Windows Embedded Standard 7

White Paper

Published April 2010
Revised March 2012

For the latest information, please see http://www.microsoft.com

Table of Contents

Overview

The design goal for an embedded system is typically very different from the design goal of a fully functional, general-purpose system. A general-purpose system with a full complement of features enables its users to enjoy the most free form, creative experience possible, and also to utilize the capabilities of its hardware platform to the fullest. An embedded system, by contrast, strives to lock down the functionality to enable only the function of the embedded application. Windows Embedded Standard 7 has achieved this goal, and it thus requires embedded systems developers to know the techniques and tools to determine the exact dependencies of the embedded application for inclusion in the operating system image. The purpose of this white paper is to educate the reader about those techniques and tools.

Creating a Target Image

To create a Target image, follow these three steps:

1. Identify the dependencies of your Target applications with Process Monitor, a Windows Sysinternals tool.
2. Feed the output of Step 1 to Package Mapper (a Windows Embedded Standard 7 application) to create a Windows Embedded Standard 7 Answer file.
3. Create an embedded image with the Windows Embedded Standard 7 Answer file.

Reference Platform for Dependency Analysis

It is recommended to have one of the following operating systems for dependency analysis:

- Windows 7.
- Maxboot Image of Windows Embedded Standard 7 (Windows Embedded Standard 7 image with all of the packages included).

It is always good to have a clean reference platform with an absolute minimum of applications and services running. This helps ensure that fewer events occur during dependency analysis. To keep the reference platform clean during dependency analysis, follow these suggestions:

1. Do not install any extra applications.
2. Do not enable Windows Update.
3. Disable Prefetching.

Understanding Application Dependencies

Each application has a list of implicit dependencies that are mandatory for launching an application successfully.

In addition to implicit dependencies, each application can have runtime dependencies that are loaded on demand. These are loaded only when a particular functionality within an application is exercised. Procmon (a Sysinternals application) is capable of capturing both of these dependencies together.

Types of Dependencies

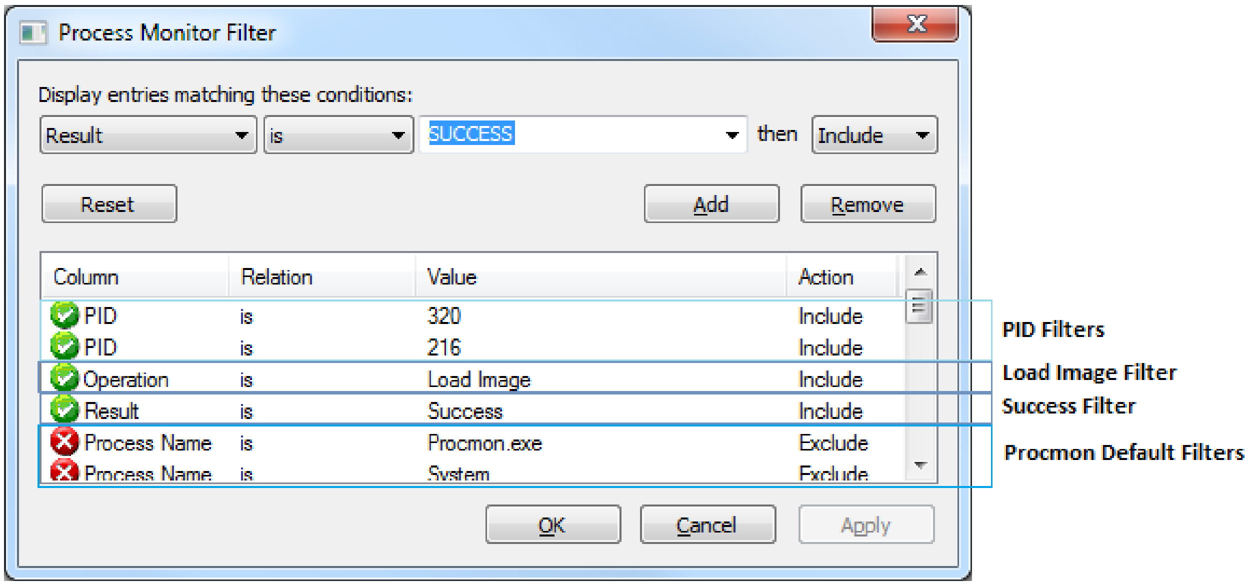
For each application, two types of dependencies must be identified:

1. Installer dependencies, which are required for an application to be installed successfully (if the application has an installer).
2. Application dependencies, which are needed for an application to run successfully.

Capturing the Dependencies

1. Launch Procmon (refer to Appendix A).
2. Configure the settings (refer to Appendix A).
3. Reset the events that are captured already (Edit->Clear Display).
4. Enable Capture Events (File->Capture Events).
5. To capture the Installer dependencies, launch the installer and install the Target application. To capture the Applications dependencies, launch the application and run through the list of test cases, exercising all of the functionalities that the application should satisfy (Procmon captures all of the events that happen during this activity).
6. Disable Capture Events, as we are not interested in any further events.
7. Launch Process Tree in Procmon (Tools->Process Tree).
8. Identify the Process IDs corresponding to the application (refer to Appendix A).
9. There are two types of events that are important in this situation:
    a. Load Image activity, which gives the list of binaries loaded in the Target application address space.
    b. File Read activity, which provides the list of files read by the Target application.

   The steps that follow illustrate how these dependencies can be extracted by setting the right filters.



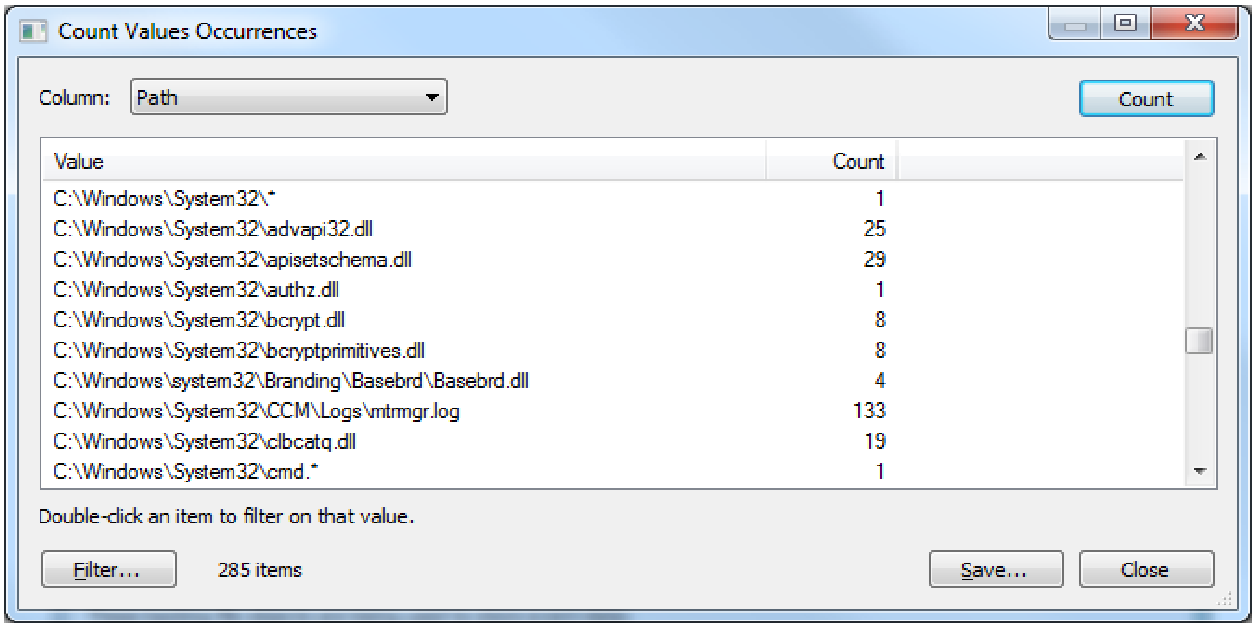
Display entries matching these conditions:

Result | is | SUCCESS | then | Include

Column	Relation	Value	Action
✓ PID	is	320	Include
✓ PID	is	216	Include
✓ Operation	is	Load Image	Include
✓ Result	is	Success	Include
✗ Process Name	is	Procmon.exe	Exclude
✗ Process Name	is	System	Exclude

Figure 1- Process Monitor Filter.

10. Launch Filter dialog (Filter->Filter…). Click the Reset button to reset any filters that are currently in effect. After clicking Reset, there will be still some entries in the Filter list. These exclude system activity and Procmon's own activity. It is permissible to have these events). Now add the following filters:
    a. Add Include Filters for the Process IDs identified in Step 8.
    b. Add Include Filter for the Operation Load Image.
    c. Add Include Filter for the Result Success.
11. Launch Tools->Count Occurrences. Choose Path from the combo box adjacent to the label "Column" (see Figure 2 below) and click the Count Button. Click the Save button to save the output.



Value	Count
C:\Windows\System32\*	1
C:\Windows\System32\advapi32.dll	25
C:\Windows\System32\apisetschema.dll	29
C:\Windows\System32\authz.dll	1
C:\Windows\System32\bcrypt.dll	8
C:\Windows\System32\bcryptprimitives.dll	8
C:\Windows\system32\Branding\Basebrd\Basebrd.dll	4
C:\Windows\System32\CCM\Logs\mtrmgr.log	133
C:\Windows\System32\clbcatq.dll	19
C:\Windows\System32\cmd.*	1

Figure 2- Process Monitor Count Occurrences Dialog.

12. Launch Filter dialog (Filter->Filter…). Change the following filters:
    a. Remove the Include Filter for Operation Load Image (this Filter was added in Step 10).
    b. Add Include Filter for the Operation ReadFile.
13. Launch Tools->Count Occurrences. Choose Path from the combo box adjacent to the label "Column" and click on the button "Count". Click on the "Save" button to save the output.
14. Building an OS image in Windows Embedded Standard 7 involves identifying a set of packages. Each package encapsulates an OS feature and contains all of the binaries that contribute to that feature. After identifying the set of binaries from the steps above, the binaries must be mapped to the packages with Package Mapper.

Using Package Mapper

Package Mapper is a Windows application located in the Value-Add Folder in the media. Refer to Appendix B for more details on Package Mapper.

Package Mapper application maps binary files to packages. Package Mapper is also capable of processing the Procmon Output file.

To use the Procmon Output file with Package Mapper, follow these two steps:

1. Open the Procmon Output file (captured in the previous section) in Excel and delete the Count column. Also delete the header line in the first column.
2. Feed this file as input to Package Mapper (refer to the Package Mapper manual for the input parameters).

The output of Package Mapper is a Windows Embedded Standard 7 Answer file, which lists the packages that are necessary for the Target application to run successfully on an Embedded OS.

Package Mapper accepts any number of Input files; it can also be used to pass in both of the Output files (Installer dependencies and Application dependencies). The output of Package Mapper is a Windows Embedded Standard Answer file that will support the entire Target application.

Package Mapper also outputs a Mapping file. This file lists the packages in the Answer file and the binary files that are contained in those packages.

Trimming the Answer File

1. Open the Answer file in Image Configuration Editor and review the packages in Answer file.
2. Open the Mapping file (one of the Output files of Package Mapper) to see why each package has been brought in. If there are very few binary dependencies for any package, it is worthwhile to investigate to see if the package can be removed. If a package looks totally irrelevant, investigate to see if it can be removed.
3. Make a careful judgment whether or not this package can be removed by understanding the functionality provided by the package in Windows Embedded Standard 7. If you decide to remove the package, you can do so safely within Image Configuration Editor.
4. Resolve the required package dependencies within Image Configuration Editor and save the final Answer file.

Creating a Windows Embedded Standard 7 Image

1. Launch Windows Embedded Standard 7 Setup.
2. Choose “Build an Image” in the first screen and continue through the Wizard.
3. In the Wizard step “Choose How to create your Image,” click “Use a Template” then browse to the path that contains your Answer file. Continue through the Wizard.
4. In the Wizard step “Summary of Drivers and Features” check “Modify Drivers” and “Modify Features.” Continue through the Wizard.
5. In the Wizard step “Find and Select Device Drivers,” choose “Automatically Detect Devices.” If you have a PMQ file, choose the second Option. Continue through the Wizard.
6. In the Wizard step “Select the Packages to Include in Your Image,” uncheck “Resolve Optional Dependencies.” Review the packages on this screen. Add/Remove packages as needed. Click “Resolve” to resolve the dependencies. Continue through the Wizard.

Summary

You have now successfully created a Windows Embedded Standard 7 image that should support all of the Target applications. Install the Target application and run through all of its test cases to ensure that it is running 100% successfully on the new Windows Embedded Standard 7 image.

Appendix A – Procmon

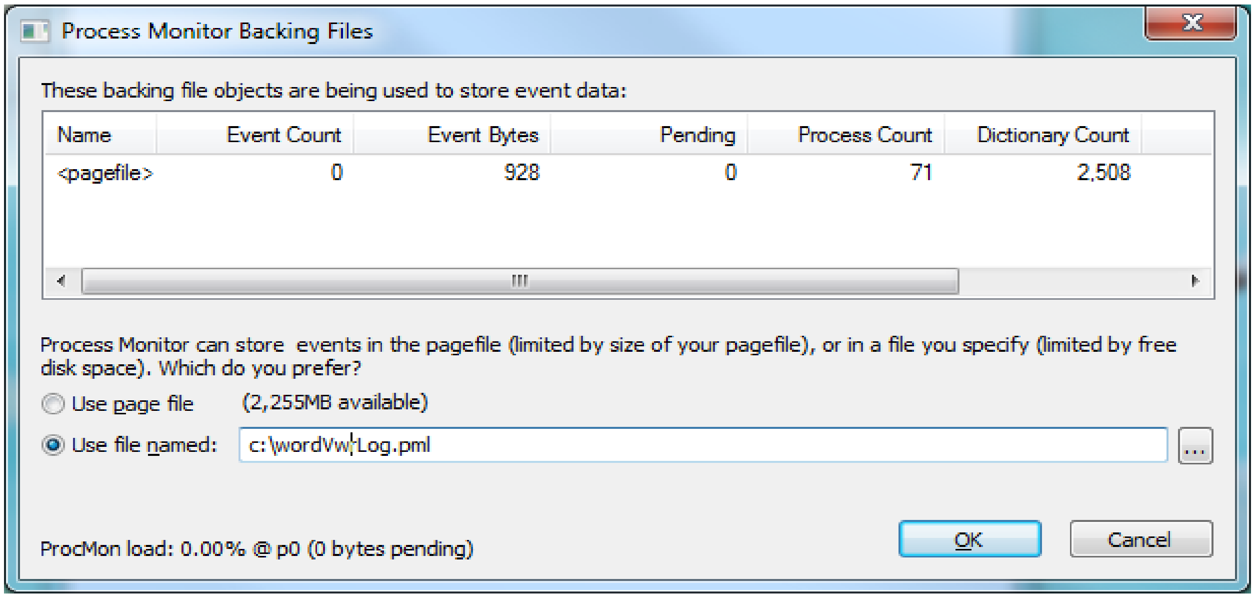
Introduction to Procmon

Procmon is a Sysinternals tool that monitors file system activity, network activity, registry activity and process and thread activity. You can download and learn more about Procmon here.

Specifying a Backing File

By default, Procmon saves the Capture data in the Paging file. In Windows Embedded Standard 7, the Page file is disabled by default. This means that Procmon would continue to consume the system RAM for events tracing and would gradually slow down the system. Using a Backing file improves performance. Backing files can be specified as follows:

1. Launch File->Backing Files.



These backing file objects are being used to store event data:

Name	Event Count	Event Bytes	Pending	Process Count	Dictionary Count
<pagefile>	0	928	0	71	2,508

Process Monitor can store events in the pagefile (limited by size of your pagefile), or in a file you specify (limited by free disk space). Which do you prefer?

Use page file [✓ Use file named:] (2,255MB available)

Use file named: c:\wordVw\Log.pml

ProcMon load: 0.00% @ p0 (0 bytes pending)

Figure 3- Process Monitor Backing File specification.

2. Click on the Radio button “Use File Named” and specify a Backing file on a drive that has sufficient space.

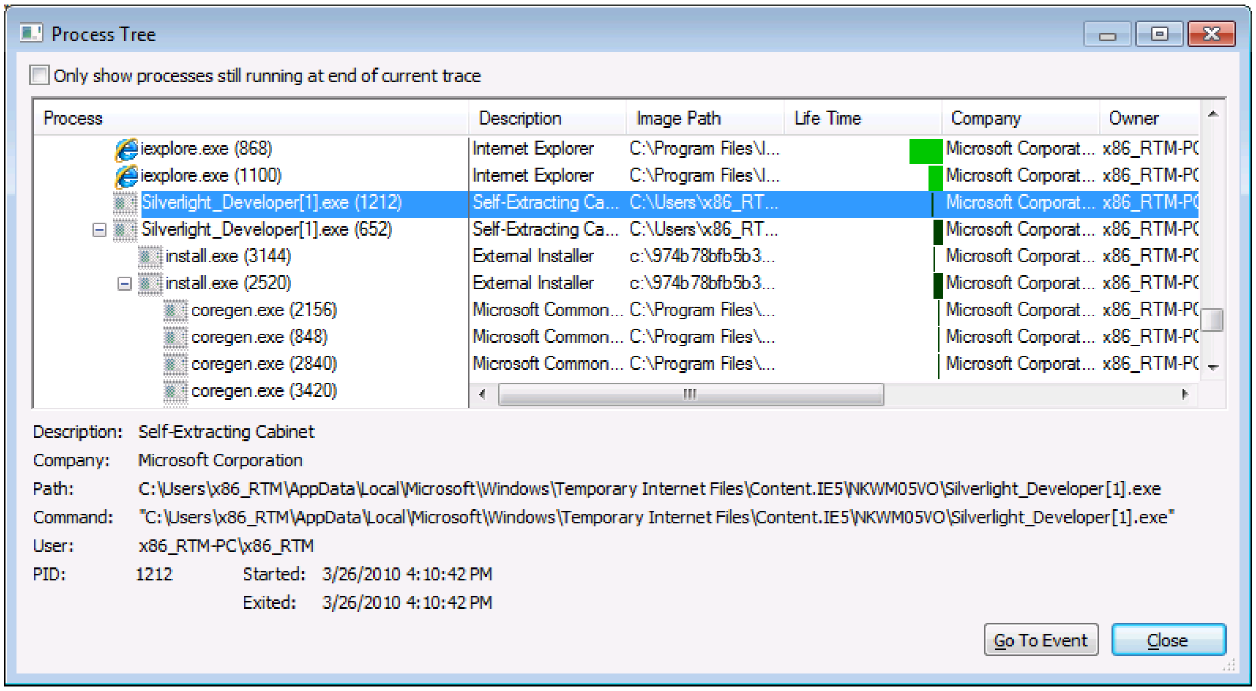
There is also an option to provide a Backing file while launching Procmon from the Command line. The syntax is as follows:

Command: Procmon.exe /BackingFile “c:\backFile.pml”

Process Tree Feature

Each application has a list of binary dependencies. In addition, there may be dependencies between applications. An application might, for example, create a Child Process and depend on it. An application can also depend on some Windows Services. It is essential to identify these inter-process dependencies, and the Process Tree can be very helpful in doing so.

By using Process Tree, for instance, you can easily identify the Child Process that your application is creating.



Only show processes still running at end of current trace

Process	Description	Image Path	Life Time	Company	Owner
iexplore.exe (868)	Internet Explorer	C:\Program Files\I...		Microsoft Corporat...	x86_RTM-PC
iexplore.exe (1100)	Internet Explorer	C:\Program Files\I...		Microsoft Corporat...	x86_RTM-PC
Silverlight_Developer[1].exe (1212)	Self-Extracting Ca...	C:\Users\x86_RT...		Microsoft Corporat...	x86_RTM-PC
Silverlight_Developer[1].exe (652)	Self-Extracting Ca...	C:\Users\x86_RT...		Microsoft Corporat...	x86_RTM-PC
install.exe (3144)	External Installer	c:\974b78bfb5b3...		Microsoft Corporat...	x86_RTM-PC
install.exe (2520)	External Installer	c:\974b78bfb5b3...		Microsoft Corporat...	x86_RTM-PC
coregen.exe (2156)	Microsoft Common...	C:\Program Files\...		Microsoft Corporat...	x86_RTM-PC
coregen.exe (848)	Microsoft Common...	C:\Program Files\...		Microsoft Corporat...	x86_RTM-PC
coregen.exe (2840)	Microsoft Common...	C:\Program Files\...		Microsoft Corporat...	x86_RTM-PC
coregen.exe (3420)					

Description: Self-Extracting Cabinet
Company: Microsoft Corporation
Path: C:\Users\x86_RTM\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\NKWM05VO\Silverlight_Developer[1].exe
Command: "C:\Users\x86_RTM\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\NKWM05VO\Silverlight_Developer[1].exe"
User: x86_RTM-PC\x86_RTM
PID: 1212 Started: 3/26/2010 4:10:42 PM
Exited: 3/26/2010 4:10:42 PM

Figure 4- Process Tree feature.

The Process Tree has six features that are helpful for analysis:

1. List of all processes running at the time an event is captured. The Process Tree helps in easily identifying an application's Parent Process and Child Process.
2. Full Path of the Executable file for each process in the Tree.
3. Command line parameter of each process.
4. Start Time and End Time of all processes.
5. Life Time graph (third column in the figure above).
6. Process ID of each process.

The following steps help to identify the list of dependencies of these processes:

1. Any Child Process that is created by the Target application is a dependency.
2. Any process that is running from the application installation directory/sub-directory is a dependency.
3. Any process that is created after the Target application is created is a good candidate for being a dependency. Look for its Command-line parameters to see if this process might be a

dependency. The Life Time column has a graph to easily identify the list of processes that is created after the creation of the Target application.

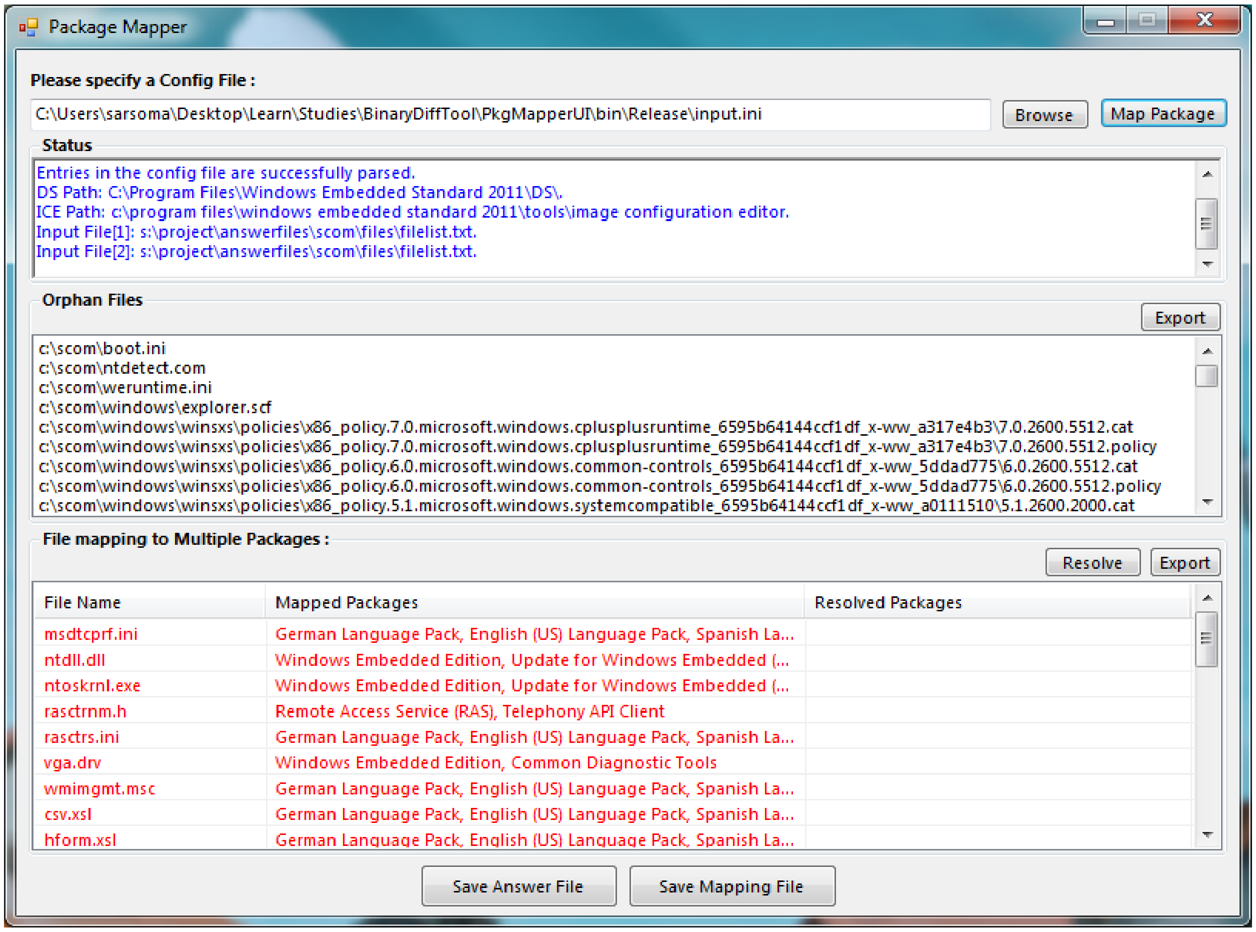
Appendix B – Package Mapper

Introduction to Package Mapper

Package Mapper is a Windows Embedded Standard 7 application that helps to map binaries to packages. The output of Package Mapper is a Windows Embedded Standard 7 Answer file. You can find the most recent version of Package Mapper and the manual on the Application Compatibility Web site here.

Prerequisites

- .NET 3.5 Framework is a requirement for this application to work.
- ICE (Image Configuration Editor) should be installed.
- Register the COM DLL EmbeddedDSI.dll present in the ICE installation folder (**regsvr32 EmbeddedDSI.dll**).



Please specify a Config File :

C:\Users\sarsoma\Desktop\Learn\Studies\BinaryDiffTool\PkgMapperUI\bin\Release\input.ini

Status

```
Entries in the config file are successfully parsed.
DS Path: C:\Program Files\Windows Embedded Standard 2011\DS\.
ICE Path: c:\program files\windows embedded standard 2011\tools\image configuration editor.
Input File[1]: s:\project\answerfiles\scom\files\filelist.txt.
Input File[2]: s:\project\answerfiles\scom\files\filelist.txt.
```

Orphan Files

```
c:\scom\boot.ini
c:\scom\ntdetect.com
c:\scom\weruntime.ini
c:\scom\windows\explorer.scf
c:\scom\windows\winsxs\policies\x86_policy.7.0.microsoft.windows.cplusplusruntime_6595b64144ccf1df_x-ww_a317e4b3\7.0.2600.5512.cat
c:\scom\windows\winsxs\policies\x86_policy.7.0.microsoft.windows.cplusplusruntime_6595b64144ccf1df_x-ww_a317e4b3\7.0.2600.5512.policy
c:\scom\windows\winsxs\policies\x86_policy.6.0.microsoft.windows.common-controls_6595b64144ccf1df_x-ww_5ddad775\6.0.2600.5512.cat
c:\scom\windows\winsxs\policies\x86_policy.6.0.microsoft.windows.common-controls_6595b64144ccf1df_x-ww_5ddad775\6.0.2600.5512.policy
c:\scom\windows\winsxs\policies\x86_policy.5.1.microsoft.windows.systemcompatible_6595b64144ccf1df_x-ww_a0111510\5.1.2600.2000.cat
```

File mapping to Multiple Packages :

File Name	Mapped Packages	Resolved Packages
msdtcprf.ini	German Language Pack, English (US) Language Pack, Spanish La...	
ntdll.dll	Windows Embedded Edition, Update for Windows Embedded (...	
ntoskrnl.exe	Windows Embedded Edition, Update for Windows Embedded (...	
rasctrnm.h	Remote Access Service (RAS), Telephony API Client	
rasctrs.ini	German Language Pack, English (US) Language Pack, Spanish La...	
vga.drv	Windows Embedded Edition, Common Diagnostic Tools	
wmimgmt.msc	German Language Pack, English (US) Language Pack, Spanish La...	
csv.xsl	German Language Pack, English (US) Language Pack, Spanish La...	
hform.xsl	German Language Pack, English (US) Language Pack, Spanish La...	

Figure 5- Package Mapper.

Input Configuration File

Package Mapper application takes a Configuration file as input. A sample Configuration file is listed below:

```
[InputFile]=S:\word_vwr_2007_installDepdencies.csv

[InputFile]=S:\word_vwr_2007_applicationDepdencies.csv

[InputFile]=S:\excel_vwr_2007_installDepdencies.csv

[InputFile]=S:\excel_vwr_2007_applicationDepdencies.csv

[DSPath]=C:\Program Files\Windows Embedded Standard 7\DS\

[ICEPath]=C:\Program Files\Windows Embedded Standard 7\Tools\Image Configuration Editor\
```

This Configuration file has three types of entries:

- [InputFile] — Full path to the input CSV/TXT file. The CSV/TXT file has only one column, which is a list of binary names. The file names can be absolute/relative. If you have multiple CSV Files, you can specify one [InputFile] entry for each.
- [DSPath] – Full path to the Distribution Share. This can be 32-bit or a 64-bit Distribution Share.
- [ICEPath] – Full path to the ICE (Image Configuration Editor) installation path.

Orphan Files

The Orphan files section contains binaries that are not OS binaries. Only OS binaries are encapsulated inside packages. Orphan files might contain binaries installed from some out-of-box applications. Review the list to identify the out-of-box application that must be installed.

Files Mapped to Multiple Packages

There are some files that are found in more than one package. Review this list and resolve the packages to be included in the Answer file. It is optional to resolve the packages. You can still continue to save the Answer file without resolving some/all of the entries in this list.

Output Files

Package Mapper has two Output files:

1. Windows Embedded Standard 7 Answer file. This file contains a list of packages that encapsulates the binaries specified in the Input files.
2. Mapping file. This file lists the packages in the Answer file and the Binary files that are contained in each package.

Additional Resources

Microsoft Windows Embedded Web site:
www.microsoft.com/windowsembedded