Implications of Migrating to Windows Embedded Standard 7 (WES7) in Embedded Applications

Kushal Koolwal – *R&D OS Engineer, VersaLogic Corp.*

A technical white paper comparing the differences between WES7 and WES2009 in regards to building and deploying a fully-functional Windows Embedded image on a target device.

1.0 Overview

This paper describes some of the differences between Windows Embedded Standard 7 (WES7) and Windows Embedded Standard 2009 (WES2009). WES7 is the next generation offering in Microsoft's family of embedded operating systems. Specifically, WES7 is the successor to WES2009, which in turn was the successor to Windows XP Embedded (XPe). The goal is to provide OEM developers with a detailed understanding of the differences between the two embedded products in regards to building and deploying a fully-functional Windows Embedded image on a target device. Specifically, this paper will highlight the major differences of the development tools, features, driver packages and the overall practical experience in building and deploying images.

2.0 Methodology

As a point of reference for this paper we used a VersaLogic Mamba (VL-EBX-37) single board computer (SBC) with an Intel®Core™2 Duo processor (Intel Montevina platform) as the target device. A WES 2009 image was first built using the familiar *Windows Embedded Studio* tools (*Target Designer*, *Component Designer* and *Component Database Manager*) and then deployed on the target hardware. Similarly, an image was created and deployed using WES7 and its new development tools (*ICE*, *IBW*, *ImageX*, etc.). In the process, all differences between WES2009 and WES7 that were encountered through the experience of building and installing the embedded images were noted.

3.0 Differences - Migrating from WES2009 to WES7

In this section we present the differences between the two products based on various categories.

3.1 Installation Requirements

The following table shows the differences in minimum requirements between WES2009 and WES7 for both the development and target systems:

Requirement	WES2009	WES7
Development (Host) System		
Host OS	Windows XP/Vista	Windows 7/Vista
CPU Speed	1 GHz	1 GHz
RAM	512 MB	1 GB
.NET Framework	3.0	3.5 (SP1)
SQL Server Express	2005 or later	N/A
Storage Space	5 GB	7 GB
Target System		
Processor Architecture	32-bit only	32-bit and 64-bit (SMP[1])
SMP	✓ Yes	✓ Yes
CPU Speed (min.)	728 MHz	1 GHz
RAM (min.)	256 MB	512 MB
Storage (min.)	126 MB	1 GB

Implication

Migrating to WES7 represents a significant change in hardware and software requirements for both the development and target systems. At the same time, WES7 can take advantage of the higher performance from the 64-bit counterpart systems.

3.2 File System Support

Starting from WES7, installing Windows Embedded is only supported[2] on an NTFS file system. WES2009, on the other hand, supports installation on both FAT32 and

1. SMP = Symmetric Multiprocessing Systems (multi-core architecture)
2. WES7 to install on a FAT32 partition [4] with the help of a few ugly hacks - it is not recommended

NTFS [3] file systems. However, WES7 does support reading and writing to a FAT32 partition.

	WES2009		WES7	
	FAT32	NTFS	FAT32	NTFS
Installs On	Yes	Yes	No	Yes
Read/Write Support	Yes	Yes	Yes	Yes

Implication

If your existing production system is a multi-boot OS environment in which one or more OSs are installed on a FAT32 partition and they share data with a WES2009 partition, you would have to make necessary adjustments to your storage partition structure before you migrate to WES7 because FAT32 cannot access NTFS partitions.

3.3 HAL Independence

One of the great and much awaited features in WES7 is Hardware Abstraction Layer (HAL) independence, which means that one can maintain a single Windows image that can boot on various computer platforms without having to reinstall or maintain separate images for each platform. For example, in WES2009 an OEM developer had to select one and only one particular HAL type from the following [5]:

- ACPI Uni/Multi-processor PC
- Advanced Configuration and Power Interface (ACPI) PC
- MPS Uni/Multi-processor PC
- Standard PC

For example, if there are two similar target systems, say a 586-based system (which requires the Advanced Configuration and Power Interface (ACPI) PC HAL) and a 686-based system (which requires the ACPI Uni/Multi-processor PC HAL), you will need to create and maintain two separate images (one for each system), even though most of the hardware (IDE, Ethernet, video, audio, etc.) is the same on both systems. With WES7, there is no such thing as HALs; it just has a single HAL which automatically takes care of different x86-based platform architectures and hence the same core image will now boot on both platforms. However, you must maintain separate images for 32-bit and 64-bit systems. WIM (Windows Imaging Format) technologies and an *ImageX* tool enable developers to take advantage of HAL independence. Please see the section "Maintaining Images" for more details.

Implication

Having HAL independence enables OEM developers to save time and effort by having a single core image that can boot across all platforms; there is no longer a need to maintain multiple platform-specific images.

3.4 ACPI

You can install WES2009 on target boards that do not have ACPI, since Windows XP supports non-ACPI HAL types which WES7 does not support. WES7[3] requires ACPI to be present on the target computer; otherwise WES7 installation fails with a "BIOS (firmware) is not ACPI compliant" error message. Also, you need ACPI 2.0 or greater [6] and APCI support, at least at the BIOS (firmware) level[4], in order for WES7 to install on your target board; otherwise you will likely see errors during installation.

	WES2009	WES7
Installs with ACPI	Yes	Yes
Installs without ACPI	Yes	No
ACPI Version	Any	2.0 or greater

Implication

In embedded applications there are many low-cost, low-power boards that are designed for a highly specialized purpose, often without any support for ACPI, which in turn makes them incompatible with WES7. In such a scenario, you will have to continue to use WES2009.

3.5 Toolkit

The toolkit in WES7 has a completely different set of tools than WES2009 for configuring and building images.

WES2009	WES7
Target Designer + Component Designer	Image Configuration Editor (ICE)
Component Database + Repositories	Distribution Share
SLX (Project File)	XML (Project File, aka Answer File)
9000 Drivers (as Components)	500 Drivers
1000 OS Components	150 OS Feature Set

Development Tools

As shown in the table above, WES7 comes with an entirely different set of tools than WES2009. Tools like *Target Designer*, *Component Database Manager* and *Component Designer* are no longer present in WES7. Instead, WES7 comes with a master IDE tool called *Image Configuration Editor (ICE)* which encompasses most of the functionality of the WES2009 tools. Tasks like adding 3rd party drivers, custom files, applications, etc., which were accomplished using the *Component Designer* in WES2009, are now achieved in WES7 using the "OEM"[5] and "Out-of-box Drivers" folders (contained within each *Distribution Share*). Also, tasks like adding registry entries, executing a program (batch/.exe file), etc. are accomplished by setting the SMI properties of the WES7 core components in

3. This is true for Windows 7 and Windows Vista also since WES7 is based on Windows 7, which in turn is based on Windows Vista
4. When the BIOS transfers control to the Operating System, it also passes ACPI tables (if the BIOS has them) to the OS and this is how the OS (in this case WES7) detects whether the system has ACPI support or not
5. To figure out destination folders where the files will be copied using the OEM folder, please see [7]

ICE. Please see "Appendix II: Example - Adding 3rd Party Materials" for a step-by-step example.

Distribution Shares

WES7 includes two *Distribution Shares (DSs)*: one each for 32-bit and 64-bit architectures. These *DSs* provide respective applications and drivers for each of the architecture types.

Image Size

As WES7 is completely based on Vista/Windows 7 technologies, there is considerable change in their runtime image sizes due to the fundamental change in the building blocks of the OS (i.e., components). *Components* in WES2009 are replaced by *Feature Sets* and *Driver Packages*, which are an aggregation of the WES2009 components. With WES2009, one could create a bootable image at just under 50 MB. With WES7, a minimum bootable image is around 300 MB (almost 6 times larger in size). A relatively large image size in WES7 can also be attributed to the way in which Vista/Windows 7 is designed as it requires more space to install compared to Windows XP.

Components in WES2009 vs. Features Sets and Driver Packages in WES7

In WES2009, all of the OS building blocks (like drivers and components) were provided as *Components,* which provided no clear distinction between drivers and OS components. For example, OS components like Windows Media Player, Remote Desktop, Internet Explorer and a driver for an AMD PCNET Ethernet adapter were all categorized as *Components*.

Moreover, with 10,000+ components in WES2009, it is very challenging for OEM developers to build a bootable image the first time without missing boot-critical components. WES7 simplifies this by providing an *eCore (Embedded Core)* package which gets included in every project at minimum and makes sure that the image has boot critical and bus enumerator drivers. Also, WES7 aggregates numerous feature packages to form one single large package, thereby reducing the number of feature packages by almost 1/6 (~150 packages) that of WES2009.

By eliminating some of the old legacy drivers, the number of driver packages in WES7 is reduced by nearly half as compared to WES2009. There are approximately 100 important drivers that are included in the *eCore* package[6], giving OEM developers a jump start in building their image. In WES2009, all drivers had to be manually selected in the *Target Designer* as components, making the image building process quite tedious.

Project Files

There are no more familiar SLX project files in WES7. With WES2009, an SLX file was used to keep track of project settings like which components were are added along with their configuration settings. Starting with WES7, all project configurations, like features, drivers, applications, language packs, etc. to add, are maintained in an XML-based file called the *Answer File*. When you use ICE to add various components to your OS image, all of the actions are entered as corresponding XML entries into the *Answer File*, including the DS (32-bit or 64-bit) you have selected. The output of ICE is always an *Answer File*.

In WES2009, a developer would check the dependencies of the components before building the final image. This dependency check concept is also included in WES7, but in the form of "validating" the XML-based *Answer File*. You can go to **Validate → Validate Only** at anytime in *ICE* to check if all of the dependencies are satisfied or not. Please see "Appendix-I: Target Designer and ICE Screenshots" for an overview of the layout.

Implication

Although WES7 has comparable tools to WES2009, it provides a completely different methodology for creating images. By eliminating the time consuming process of selecting all of the appropriate components, drivers, etc. to produce a functional image, OEMs are able to focus on building the project's application.

3.6 Image Building Model

WES7 comes with 3 installation discs:

a. Toolkit DVD (contains *ICE* and *DS*)

b. 32-bit Windows Embedded Standard 7 DVD (contains *WinPE*, *IBW* and *DS*)

c. 64-bit Windows Embedded Standard 7 DVD (contains *WinPE*, *IBW* and *DS*)

The Toolkit DVD is used to install the IDE tool *ICE* and associated *DS* (32-bit/64-bit) on the development computer. The 32-bit and 64-bit Windows Embedded Standard 7 DVDs are bootable WinPE DVDs that contain the *Image Builder Wizard (IBW)* and the corresponding 32-bit or 64-bit *DS*. These DVDs are typically used to boot into Windows PE on the target device and apply the runtime image created with *ICE* or to prototype image creation using the wizard and various templates available in *IBW*.

➤

6. *eCore* consists of NT kernel, boot critical drivers, *WinLogon*, *NetLogon*, *Filesystems*, command shell, networking stack, RPC, etc.

Image Building Model	WES2009	WES7	
		ICE/Advanced Method	IBW/Express Method
1. System Analysis	*Tap.exe* → *Devices.pmq* → Import into *TD*	*Tap.exe* → *Devices.pmq* → Import into *ICE*	Boot from WinPE IBW DVD *IBW* runs *tap.exe*
2. Image Configuration	Configured on developer machine using *TD*	Configured on developer machine using *ICE*	*IBW* maps corresponding drivers from *DS* based on tap file
3. Image Customization	Using *Component Designer*	Using *ICE*	Select an application template or manually select *Feature Sets* and *Driver Packages* from the *DS*
4. Project File	SLX file is ready	Valid *Answer File* is ready	A valid *Answer File* is created based on the selection
5. Image Building[7]	Build on developer machine using *TD*	Always build on target device using *IBW*	*IBW* starts assembling and building the image
6. Adding Components Post-Build	Requires rebuilding of the entire image	Can be added offline and online using *DISM*	

With WES7, the image building process has been completely overhauled. In WES2009, a developer first has to configure and build an image on the development system. With WES7, a developer can bypass the ***ICE/Advanced Method*** entirely and quickly install (configure and build) WES7 images directly on the target device using the ***IBW/Express Method*** for quick prototyping by booting off of WinPE/IBW DVDs as shown above.

The *Image Builder (IBW)* program can install images in two modes:

a. ***Interactive Mode*** - Using guided step-by-step wizard

b. ***Unattended Mode*** - Using *Ununattend.xml Answer File*

The following steps occur in ***Interactive Mode*** upon booting from the IBW WinPE disc:

1. WinPE is launched
2. (Optional) *Image Builder* asks if you want to deploy an *Answer File* or *WIM image*
3. *Tap.exe* is run by *IBW* and the *devices.pmq* file is generated
4. *IBW* maps the corresponding drivers from the *DS* based on the above PMQ file
5. *IBW* gives the developer an option to select an application template or manually select a *Feature Set* and additional driver packages from the *DS*
6. A valid *Answer File* is created based on the selection
7. *IBW* starts assembling and building the image based on the *Answer File*, run through the *Configuration Passes* and then reboots to the "Windows Welcome" screen

In ***Unattended Mode*** an *Answer File (Unattended.xml)* can be created using *ICE* on the development machine and supplied to the *Image Builder* when booting from the IBW WinPE disc on the target device, thereby automating the installation process. To create fully unattended *Answer Files* please refer to [8].

No matter whether *ICE* is used to create an *Answer File* or *IBW* is used directly on the target device, *Image Builder* ultimately does all of the heavy lifting on the target device to build the WES7 image. In certain situations, this can be a major drawback.

FBA vs. Image Builder Phase

With WES2009, when you copy your image generated from *Target Designer* to your target device media (like a hard drive or CompactFlash) and boot from it, the *FBA (First Boot Agent)* program is executed to install the rest of the WES209 system. The *FBA* process has phases numbered from 0 to 65535 [9]. Depending on the phases, different settings are applied to the image. One can configure actions that take place at a particular *FBA* phase by creating a component using *Component Designer*.

With WES7, the *FBA* model is replaced by a *Configuration Pass* [10] model which is executed by *IBW* on the target device. Following is a table describing the passes that are executed[8] on the target device during installation of a WES7 image:

Image Builder Logical Phases	Configuration Pass	Executed by		Command to Execute the Pass
		IBW	Sysprep	
Host OS	windowsPE	✓ Yes	✗ No	N/A
	offlineServicing	✓ Yes	✗ No	
	Generalize	✗ No	✓ Yes	sysprep /generalize
Online Config.	specialize	✓ Yes	✗ No	N/A
	auditSystem	✗ No	✓ Yes	sysprep /audit
	auditUser	✗ No	✓ Yes	
Windows Welcome	oobeSystem	✓ Yes	✓ Yes	sysprep /oobe

One can configure various actions to occur in each of these passes by modifying the properties in the *Answer File* for that particular pass using *ICE*.

Implication

The image building process has gone through a significant change since WES2009. With WES7, OEM developers are able to configure and build images over the target device and have the option of completely automating WES7 installation. Although replacing the entire *FBA* process with the *IBW* and *Configuration Passes* introduces a significant learning curve, this new model gives OEM

7. *Image Builder* in WES7 replaces *winnt.exe* and *winnt32.exe* in WES2009
8. All the passes are executed in sequence which have "Yes" in the "Executed by IBW" column

developers more control and flexibility over the image configuration and building process. One drawback of the building model is that if the target device is low performance, it could take hours to build the WES7 image[9].

3.7 Deploying an Image

In WES2009, one of the common methods to test (and sometimes deploy) an image on the target device during the development phase is to copy the contents from the "C:\Windows Embedded Images" folder on the development machine to the root (usually Drive C:) of a Windows XP bootable partition on a target storage device (hard drive, CompactFlash, USB drive, etc.) and boot from it.

With WES7, if you are using *ICE*[10], once you finish preparing your *Answer File* you have the following options to test/deploy your image:

1. You can create a WinPE/IBW bootable[11] USB Flash Drive (UFD) using *ICE* by **Tools → Create Media → Create IBW Image from Answer File** and then copying the contents generated to the UFD of the development machine (similar to copying the build directory in WES2009 as mentioned above).
2. Alternatively, you can create a WinPE/IBW bootable CD/DVD-ROM [11] (similar to creating a bootable UFD as mentioned above) to install WES7 on the target device.

Note: *Unlike WES2009, this bootable CD/DVD is not for running WES7 directly from the CD/DVD-ROM.*

Once you prepare your install media (UFD or CD/DVD), you can boot from it on the target device and IBW will start installing WES7.

Following is a brief table comparing alternate deployment methods:

Method	WES2009	WES7
PXE Boot	✓ Yes	✗ No (replaced by *WDS*[12])
Remote Boot	✓ Yes	✗ No[13]
USB Boot	✓ Yes	✓ Yes
CD/DVD	✓ Yes	✗ No
WDS	✗ No	✓ Yes

Sysprep an Image

Let's take a look at an example which describes the differences between WES2009 and WES7 when sysprep'ing an image, which is a common technique used for mass deployment on target devices.

Steps	WES2009	WES7
1. Image configuration	*Target Designer* on development machine	*ICE* on development machine
2. Add Sysprep tool	Add "Sysprep Component[14]"	N/A
3. Configure Sysprep tool	Right-clicking on the component	N/A
4. Booting image on target device	Your preferred method to boot	Your preferred method to boot
5. Locate sysprep tool directory on target image	C:\sysprep\sysprep	C:\Windows\System32\sysprep
6. Sysprep your target image	sysprep[15].exe -reseal -reboot -- or -- fbreseal.exe (if you are using the "System Cloning" component [12])	sysprep.exe /generalize /oobe[16]/reboot -- or -- sysprep.exe /generalize /audit /reboot

As you can see, the switch *reseal* is replaced by *generalize*. They both achieve the same purpose (removing SID, computer name, user settings, etc.). Many features of the *Sysprep* tool, such as the audit and factory mode, were not supported in WES2009 for the *ConfigMgr OSD*[17] functionality which is now supported in WES7. You also need to delete the *Sysprep* tool in WES7, unlike in WES2009 where it is deleted automatically once you have resealed your image. In WES7, a developer can reseal an image multiple times until it is finalized, whereas in WES2009 you can only reseal an image once.

Implication

A lot of the new deployment methods in WES7 make uses of the existing Windows 7 deployment techniques. This allows OEMs to leverage the benefits of these advanced technologies while simultaneously reducing the learning curve.

3.8 Log Files

Log files are another area where things have changed considerably with respect to WES7. In WES2009, to debug issues during the image install (FBA) process, one would

9. Microsoft is working on giving users a more intuitive option to build WES7 images offline using *DISM*
10. As mentioned earlier, there are two methods to build images in WES7 - using *ICE* or *IBW*
11. First you need to make a UFD bootable as a separate step [20] - *ICE* does not do it for you automatically
12. *Windows Deployment Services*
13. Since WES7 images are considerably large, this is not a very practical method. However, one could possibly do it if the target device has sufficient RAM.
14. You can also use the "System Cloning" component in WES2009, but you cannot use both
15. Both the *sysprep.exe* in WES2009 and WES7 will launch in GUI mode if you don't specify any switch
16. The option *oobe* tells *Sysprep* to boot into the *Windows Welcome* screen
17. *System Center Configuration Manager (ConfigMgr) Operating System Deployment (OSD)*

typically look at the following files in the installation partition (usually C:\) of the target device:

- C:\Windows\setupapi
- C:\Windows\FBA\FBALOG

With WES7, *Image Builder* logs its actions in different directories and storage media (ram disk, hard drive, etc.) depending upon the installation phase[18]:

Installation Phase	Log Files Location	User Interaction
Autorun (Just when WinPE is about to load)	No log files are created	Before clicking "Build an Image" or "Deploy an Answer File or WIM"
WinPE	X:\[19]Windows\Sources\Panther\setupact.log X:\Windows\Sources\Panther\setuperr.log	Before clicking "Next" on the disk configuration screen
During Installation	C:\Windows.~BT\Panther\Sources\setupact.Log C:\Windows.~BT\Panther\Sources\setuperr.Log	Once disk partition is selected
After Installation	C:\Windows\Panther\setupact.Log[20] C:\Windows\Panther\setuperr.Log C:\Windows\inf\setupapi.Dev.Log C:\Windows\inf\setupapi.App.Log	After installation completes and windows prepares to start for the first time
Roll back (optional)	C:\Windows.~BT\Sources\Rollback*	If fatal error occurs and setup fails and decides to roll back
Sysprep (optional)	C:\Windows\System32\Sysprep\Panther - Generalize C:\Windows\Panther\Unattendgc - Unattended actions	When you execute *Sysprep* command with different parameters

How to Access Log Files on a RAM Disk?

A RAM disk is destroyed if system power is turned off. To copy the log files from the RAM disk before the power is turned off:

a. Press ***Shift*** + ***F10*** keys and a WinPE command prompt will appear

b. Execute ***wpeinit***, which starts various WinPE services (including networking)

Use the ***net use*** command to move the log files from the X:\ drive to a network share.

IBW Log Files

a. ***setupact.log*** - All the actions of *IBW* are logged in this file

b. ***setuperr.log*** - Any error(s) encountered by *IBW* will be logged in this file

Driver and Application Installation Log Files

With WES7, there are two separate log files, as a part of *SetupAPI* logs, for driver and applications installation:

a. ***SetupAPI.dev.log*** - Plug-N-Play device installation event log file which is helpful for diagnosing driver setup issues

b. ***SetupAPI.app.log*** - Application installation including log file

Sysprep Log Files

Sysprep logs *Image Builder* actions in different directories depending upon the configuration pass. Because the *generalize* pass deletes certain *Image Builder* log files, *Sysprep* logs generalize actions outside of the standard *Image Builder* log files.

Implication

The location and names of log files have changed completely in WES7 because of the fundamental change in the *Image Building* process (*IBW* instead of *FBA*). Moreover, log files in WES7 are categorized into different locations and file names depending upon the type of action performed by *Image Builder* and *DISM*. This makes the debugging process much easier and intuitive.

3.9 Maintaining Images

The latest WIM (Windows Imaging Format), a "hardware-agnostic" file-based format, can be used for maintaining HAL-independent images. You can start by creating a WES7 core image, capturing it with the *ImageX* [13] tool that is supplied with WES7, and then mounting the image (.wim) on your development computer file system and making any necessary updates (servicing) offline (adding drivers, applying 3rd party updates, installating applications, etc.) for your respective platform. Because several images can be created inside one single image, maintaining WES7 images is very easy and efficient. However, separate images are required for different processor architectures (32-bit vs. 64-bit platforms) since the drivers are not compatible across processor architectures.

WES2009 has a deployment tool called *SDI (System Development Image)* which is somewhat similar to *ImageX*, although it lacks some of the advanced features like servicing images offline and creating multiple instances of images in one single file [14].

Implication

With new technologies like WIM and tools like *ImageX*, WES7 provides a cost-effective means of maintaining OS images since a single image can be deployed on a variety of target platforms.

➤

18. There are no official installation phase names. These phases roughly equate to one or two "Configuration Passes" discussed previously
19. Drive letter X denotes that log files are stored in a RAM disk (a temporary disk like space in RAM)
20. Log files get moved/copied from the RAM disk to the hard drive once the system knows what the target partition will be

3.10 Localization

The localization i.e. *language packs (LPs)* support in WES7 has been redesigned from scratch and now includes language-neutral design in terms of OS components, meaning that the localized resources are now separated from the non-localized resources. This enables OEMs to quickly localize a particular language, fixing non-localized bugs without having to re-ship the entire binary.

	WES2009	WES7
What can be localized?		
OS Image	✔ Yes	✔ Yes
Developer Tools	✘ No	✔ Yes
Documentation	✘ No	✔ Yes
Localization Features		
Language Install Source	*MUI*	*Language Packs (LPs)*
Adding Language(s) Offline	✘ No	✔ Yes
Included in Main Install Disc	✘ No	✔ Yes
Language-Neutral Design	✘ No	✔ Yes
No. of Languages Support	23	36
OS Binaries	Have to be fully localized	Only apply to relevant *LP*
Resulting OS Footprint	Large	Small

Moreover, the *LPs* only contain language resources for the corresponding design-neutral parts of the feature sets, drivers, etc. and hence the resulting OS footprint is smaller in size relative to WES2009.

Implication

The improvement in WES7 localization reduces the need for servicing security and feature updates significantly.

3.11 Application Development

Applications for WES7 can be developed with tools like *Visual Studio 2008* just like in WES2009 or XPe. Almost all target applications in embedded projects have dependencies which need to be satisfied by the OS image in order for the application to run successfully in Windows. These dependencies are classified into two categories:

a. Static Dependencies (linked at compile time)

b. Dynamic/Runtime Dependencies (runtime DLLs called)

In WES2009, one would use 3rd party tools like *Process Monitor*[21][15] and *Dependency Walker Path* [16] to find dependencies. The relevant components could then be included in the SLX file. This can be a time consuming and frustrating process, especially when an application has a significant number of dependencies.

Static Dependencies in WES7

In WES7, there are new tools that make including applications in a WES7 project nearly seamless. First of all, *ICE* has an *SDA (Static Dependency Analyzer)* tool for analyzing static dependences for .exe, .dll, and .msi files. This is done in *ICE* via **Tools → Analyze Static Dependencies** and then browsing to the file (usually an .exe, .dll, etc.) to analyze.

Dynamic Dependencies in WES7

SDA is not a comprehensive dependency analysis tool and the *Process Monitor* must be used in WES7 to analyze required dynamic dependencies[22]. However, there is another tool called *Package Mapper* [17] for WES7 which can process output from *Process Monitor*, which is usually a file that lists all of the binaries that are required and maps those dependencies (binaries) to WES7 packages. The resulting file is an *Answer File* which can now be consumed by *ICE*. The process looks like this:

Process Monitor → Package Mapper → Answer File → Build Image

Templates

Templates are where WES7 steals the show from WES2009 when it comes to supporting a target application. The output of *Process Monitor* contains unnecessary files, noise, registry entries, etc. and thus it could take weeks to identify dependencies. WES7 now has a dedicated website which provides various templates for your target application. For example, let's say you want "Windows Live Messenger" in your target OS image. Instead of analyzing dependencies that are required to run the *Messenger* service, you can simply go the WES7 compatible application website [18] and download the "Windows Live Messenger" template and import it into *ICE* via **File → Import → Import Template**. The imported template can then be added to the *Answer File*.

Implication

Finding target application dependencies can be a challenging and time-consuming process. From an OEM perspective, the powerful tools in WES7 help save time and resources by performing all of the complicated analysis beforehand and providing a ready-to-use template for a project. Independent Software Vendors (ISVs) can work with Microsoft's WES team to create and upload a template for their application on the *WES7 Compatible Application* website [19].

4.0 Conclusion

With its new set of powerful tools and technologies, WES7 is a dramatic improvement over its predecessors, especially given the degree of customization and flexibility that it offers in regards to image configuration and building. WES7 delivers the power and reliability of

21. It combines the features of two legacy Sysinternals utilities: *Filemon* and *Regmon*

22. You can use *Process Monitor* to analyze static and dynamic dependencies and hence completely bypass the built-in dependency tool in WES7

the Windows 7 operating systems, while also providing the essential features of an embedded operating system - componentization and customization. The focus has been to reduce time to market by enabling OEM developers to spend more time on their core competency of application development and less time figuring out the correct set of components required to build a fully-functional image. Although WES7 lacks a few features which were available with its predecessors, overall it is a sleek embedded operating system. There is definitely a significant learning curve in transitioning to WES7, but the benefits far outweigh the costs.

Kushal Koolwal
R&D OS Engineer, VersaLogic Corp.

In his current position, Kushal works with multiple embedded operating systems (Linux, Windows Embedded, QNX, etc.) to ensure they run efficiently on VersaLogic's embedded computers. Kushal completed his bachelor's degree in *Computer Engineering* in 2004 in India. He then pursued a Master's degree in *Computer Science*, which he received in 2006, followed by an MBA in 2008, both from the *University of Oregon*. Additionally, Kushal is a database/web developer using .NET technologies and blogs about Linux solutions and businesses. Kushal's hobbies include traveling, playing sports and dancing.

References

[1] http://msdn.microsoft.com/en-us/library/ff794877.aspx

[2] http://msdn.microsoft.com/en-us/library/ff793769.aspx

[3] http://windows.microsoft.com/en-US/windows-vista/Comparing-NTFS-and-FAT-file-systems

[4] http://www.tomstricks.com/how-to-install-windows-vista-on-a-fat32-partition-instead-of-ntfs/

[5] http://support.microsoft.com/kb/299340

[6] http://download.microsoft.com/download/5/b/9/5b97017b-e28a-4bae-ba48-174cf47d23cd/CPA002_WH06.ppt

[7] http://unattended.msfn.org/unattended.xp/view/web/18/

[8] http://msdn.microsoft.com/en-us/library/ff794599.aspx

[9] http://msdn.microsoft.com/en-us/library/dd450713.aspx

[10] http://msdn.microsoft.com/en-us/library/ff794273.aspx

[11] http://msdn.microsoft.com/en-us/library/ff793723.aspx

[12] http://msdn.microsoft.com/en-us/library/dd450718.aspx

[13] http://technet.microsoft.com/en-us/library/cc507842.aspx

[14] http://msdn.microsoft.com/en-us/library/ms940108%28WinEmbedded.5%29.aspx

[15] http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx

[16] http://www.dependencywalker.com/

[17] http://code.msdn.microsoft.com/packagemapper

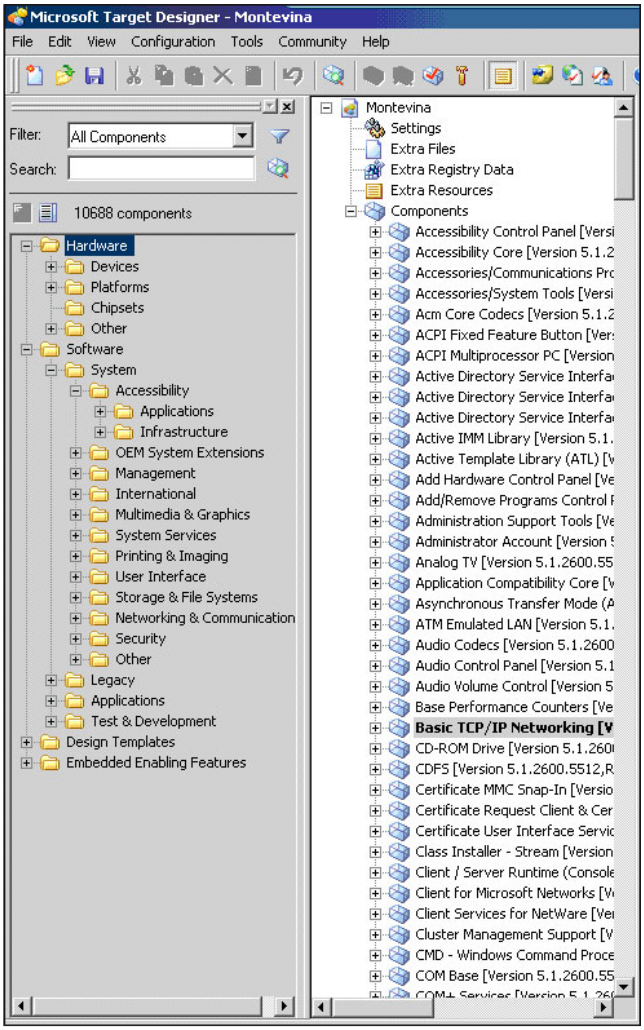
[18] http://www.microsoft.com/windowsembedded/en-us/products/wes-tandard/applications.mspx

[19] http://www.microsoft.com/windowsembedded/en-us/products/wes-tandard/template_faq.mspx

[20] http://msdn.microsoft.com/en-us/library/ff795043.aspx

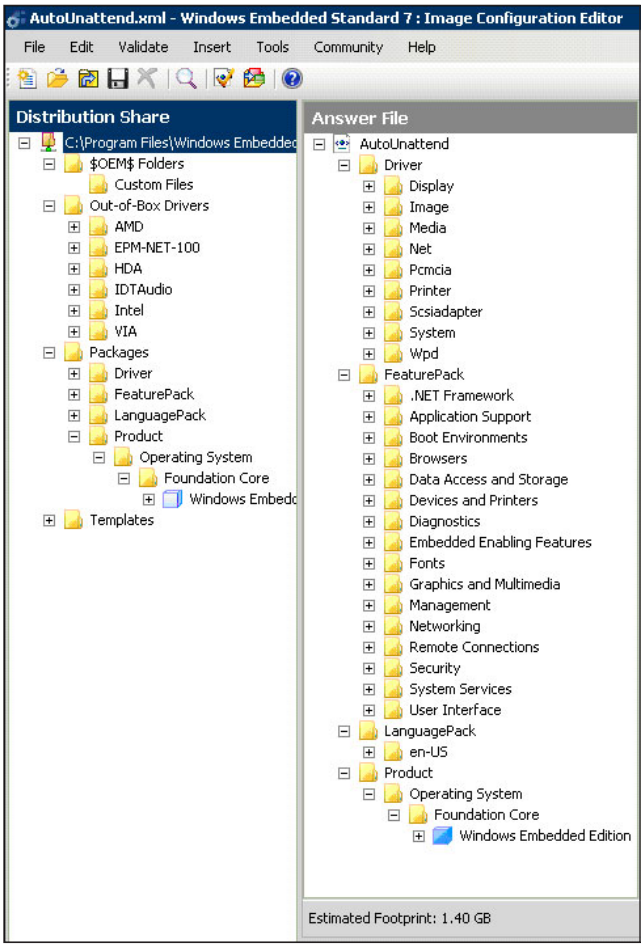
Appendix-I:
Target Designer and ICE Screenshots

Target Designer:



➤

Image Configuration Editor (ICE):



Appendix-II:
Example - Adding 3rd Party Materials

➤

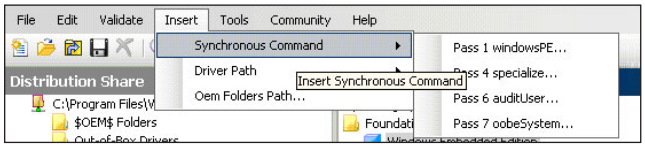
	WES2009	WES7
Adding Files		
.txt, .jpg, .etc files	1.Create an SLD component in "Component Designer" 2.Add your file and its path in the "Files" tab 3.Create a new repository 4.Import your SLD file in "Component Database Manager" 5.Finally, add the newly created component in your *Project File (SLX)* in "Target Designer"	1.Right-click on "OEM Folders" in *ICE* and select "Explore From Here" and copy your file depending on where you want your file to reside on the final image on the target 2.Right-click again on "OEM Folders" in *ICE* and select "Insert OEM Folders Path"
Adding Drivers		
INF-based .sys, .dll files	1.Create a SLD component in "Component Designer" 2.Import your driver INF file 3.Make sure necessary files, resources and registry entries are added 4.Import your .SLD file in "Component Database Manager" 5.Finally, add the newly created component in your *Project File (SLX)* in "Target Designer"	1.Right-click on the "Out-of-Box Drivers" folder in *ICE* and select "Explore from Here" and copy your driver files (.inf, .sys, .dll) in a new folder inside "Out-of-Box drivers" folder. 2.Select Insert → Driver Path → Pass 2 OfflineServing
MSI-based	No direct support for installing MSI-based drivers. You need to first extract the files by running the MSI installer on an existing Windows machine and then follow the procedure for the INF-based drivers as mentioned above.	1.Add your MSI file in the *OEM* folder as explained in "Adding Files" above 2.Go to Insert → Synchronous Command → Pass 4 Specialize and type the following command: cmd. exe /c C:\<path-to-your-msi.exe>[23]
Adding Registry		
.reg files	1.Create an SLD component in "Component Designer" 2.Now you can add your registry branch in the *Registry* tab 3.Import your .SLD file in "Component Database Manager" 4.Finally, add the newly created component in your *Project File (SLX)* in "Target Designer"	1.Create a .reg file with the registry entries that you would like to add 2.Add your registry file (example. reg) in the *OEM* folder as explained in "Adding Files" 3.Go to Insert → Synchronous Command → Pass 7 Specialize and type the following command: cmd. exe /c reg import C:\<path-to-your-reg-file>[24]
Adding Applications		
.exe files	1.Use tools like *Dependency Walker* and *Process Monitor* to find all dependencies 2.Map all of the found dependencies to the components in *Target Designer* and add them to the SLX file (a very time-consuming process) 3.Build the image	1.Check the application template website and import the template if your application is listed 2.If you don't find a template, use the *SDA* tool in *ICE* to resolve dependency and include the required package(s) in your *Answer File* 3.If there are still missing dependencies, use the *Process Monitor* tool to find runtime/dynamic dependencies and use *Package Mapper* to generate your *Answer File* 4.Build the image

23. You can also install the driver silently by passing the "quiet" argument: setup.exe /quiet. This way user won't be presented with any wizard during driver installation.
24. Depending on the type of registry entries you are trying to add, you might need to execute it with the following command: psexec -s reg import C:\example.reg

So far, various differences between the tools and the process of configuring images at a higher conceptual level have been discussed. In this section we will provide actual examples of how things are done differently in WES7 as compared to WES2009. In particular we will focus on how to add 3rd party material in an OS image.

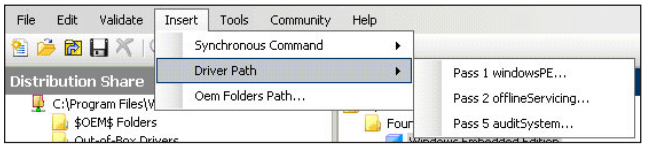
Synchronous Commands

They can be run in Pass 1, 4, 6, 7



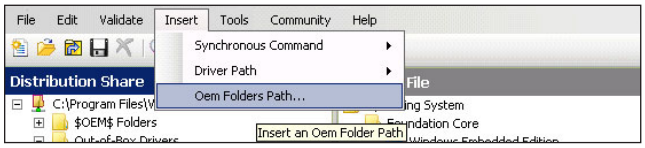
Driver Path

Driver files (INF-based) can be added in Pass 1, 2, 5



OEM Folders Path

This folder is always inserted in Pass 1



VERSALOGIC CORPORATION

A leading supplier of industrial computers since 1976, VersaLogic focuses on high-quality board-level products for embedded OEM applications. Its 5+ year production life guarantee and outstanding warranties demonstrate a commitment to service. Product platforms include EBX, PC/104™, PC/104-*Plus*, PCI-104, EPIC™, SUMIT™, and SPX™. For five consecutive years, VersaLogic has received the coveted Platinum level vendor rating based on independent user surveys by technology research firm VDC Research Group, Inc. VersaLogic is registered to the ISO 9001:2008 standard. For more information, visit www.VersaLogic.com.