Introduction to Windows® Embedded Standard 7

By Sean D. Liming
Managing Director
SJJ Embedded Micro Solutions

May 2010

Welcome – and now for something completely different…
It has been almost a decade since we have seen a new Windows Desktop OS migrated to the embedded space. Let me be the first to welcome you to Windows Vista Embedded… oh no, wait a minute, that one didn't make it. Let's try Windows 7 Embedded… no again, that is too obvious a name for the market place. We already had the well established Windows NT Embedded and Windows XP Embedded roadmap; something more marketingesque. There was something about appearing to align with Windows Embedded CE, but they are completely differently architected operating systems, but they are both for embedded. Oh yeah, Windows CE… I mean Windows CE.NET… no I am mean Windows Embedded CE's name is changing too. Since the C and E have been out of order since 1996, the name has finally been fixed and the correct name is Windows Embedded Compact (Windows EC). Of course, there was a change to Windows XP Embedded to Windows Embedded Standard 2009 because… oh something about paying for new tools and not able to carry over license stickers, paying for new stickers, perfect for bad economy! The big desktop OS must now be Windows Embedded Standard 2011… darn! Still not right. Okay, I had to use Google™ and I have it now… welcome to Windows Embedded Standard 7 (WES7).

Windows Embedded Standard 7 (a.k.a. Windows 7 Embedded) is the next generation operating system following the popular Windows XP Embedded. It is based on Windows® 7, and like Windows XP Embedded, WES7 breaks down Windows 7 into features that you can pick and choose for a custom OS build. Notice that I didn't say "components", but I am getting ahead of myself. Here is a list of the features and what's-in and what's-out for WES7:

- WES7 is based on the Windows 7 code base.
- WES7 supports 32-bit and 64-bit Intel® Architecture processors.
- Components are now feature packs, and there are fewer of them.
- Creating feature packs are exclusive to Microsoft, but there are mechanisms to build custom drivers and applications into an image.
- No more database engine required to build custom OS images – I can hear the cheering already.
- Two ways to create custom image, directly from DVD using an installation wizard or create a custom installation image from an answer file and direct install from DVD, USB or Network.
- Images are big, really big. Windows XP Embedded wasn't small by embedded standards. The smallest image that just uses COM ports was about 40MB to 50MB in size. Typical XPe OS images without .NET 3.x, were between 300MB to 600MB. WES7 32-bit OS images start at around 570MB, and the typical image size could range from 900MB to 2GB. Double these sizes for 64-bit OS.
- Many of the Embedded Enabling Features (EEF) are included
    - Enhanced Write Filter (EWF) is support with RAM and RAM-REG overlays only.
    - File Based Write Filter (FBWF) is supported.
    - Hibernate Once, Resume Many (HORM) is supported.
    - Registry Filter is included.
    - Message Box Default reply is still included.
    - A new Dialog Box Filter has been added.
    - Boot from USB flash disk is available.
- At least in the initial release, the following EEFs are not going to be supported:
    - EWF Disk Overlay support.
    - Device Update Agent (DUA) is not going to be supported.

  - Headless support is not in WES7.
  - Neither CD/DVD-ROM boot nor networking boot will be supported. Remote network install is supported
  - MinLogon is out.
  - XPEPM – Power Management Application is not in WES7.
- Support for the OS updates is a major improvement. The use of the feature packs makes it possible. More on updates later.
- RDP and Telnet are still available.
- Technologies such as Silverlight, Sensor and Location Platform, and Web Service on Devices API are included.
- FAT, FAT32, and NTFS are supported, as well as the new exFAT is included.
- Full .NET Framework and Client Profiles for versions 2.0, 3.0, and 3.5 are included.
- Games, the important test applications, are not included.

Desktop to Embedded

Obviously WES7 has a strong dependency on the Windows desktop product. WES7 almost didn't see the light of day. Longhorn / Windows® Vista was an ambitious project. There were many features and ideas that went into the project, but the end result was not spectacular. Windows 7 makes-up for many of the problems with an improved boot sequence, kernel fix-ups, and look-and-feel.

Some of the internal OS restructuring has an effect on embedded. First is how the OS is built internally. In order to support different versions of the OS: Starter, Home, Profession, Ultimate, etc. a core set of files is common to all versions, which is known as the core. The product foundation–core contains the kernel files, some management utilities, and a large group of device drivers so the OS can boot on many systems. The rumored size of the desktop core is 1GB, but the embedded team was able to pair the core back and remove some drivers to separate driver packages. This is where the 570MB minimum comes from. The good news is that Windows 7 / WES 7 stands a good chance of running on many ACPI compliant systems with built in support for most devices.

The second change is the OS architecture change regarding the hardware abstraction layer (HAL). Many XPe developers wanted to build a one size image fits all systems. Different versions of the HAL, different device drivers, and building the image to a specific device made this task a real challenge. Starting with Windows Vista, the HAL implementation has been changed. NT Embedded supported 13 different HALs and XP Embedded supported 7 different HALs. Only ACPI systems are support starting with Windows Vista. For 32-bit, the HAL layer was restructured into 3 files. On OS boot, the common HAL.DLL looks at the number of processors in order to load "ACPI-x86" for multiprocessor system or "Advanced Configuration and Power Interface (ACPI)" PC for single processor systems. There is only one HAL for 64-bit systems. The end results makes moving / migrating the whole OS from one platform to the next much easier.

Finally, there are the development tools. At the beginning of 2005 Longhorn Embedded (Vista Embedded) was on the roadmap. By the end of 2005, it was off the roadmap. Roadmaps are an important sales tool when selling to embedded developers. They want to know that the investment that is being made will continue long term. According to the development team, the changes to the desktop OS made the current implementation for Target Designer and components a real challenge. The team left it as 'maybe in the future an embedded version could be made to take advantage of the tools we have today', thus XP Embedded was going to be around a little longer. At a conference in 2006 someone from a different group showed me the Windows OEM Pre-installation Kit (OPK) tool: Windows System Image Manager (SIM). The individual used to be in the embedded group and was going to propose SIM for embedded. By the time 2007 rolled around, Vista Embedded was back on the roadmap and a customized version of SIM was going to be the main customization build tool. Not only was Vista Embedded back on the roadmap, but it was late according to members of the team! By the time Vista Embedded was ready to test, it was time to move to Windows 7 code base.

WES 7 Image Development Processes
As a result of using the OPK tools, the development process for building and supporting images will be the same as desktop PCs. Building and support images will be a little easier, but the sacrifice is footprint. If you are already familiar with SIM, you will see a similarity.

The biggest difference from XP Embedded is that the OS gets installed onto the target. There are two different paths for installation. The first is to install either the 32-bit OS or the 64-bit OS from a DVD. The DVD runs what is called the Image Build Wizard, which is very similar to installing Windows 7 from a DVD. The difference is that you get to pick and choose from 160 different feature packages. After the installation, you can install your custom applications and drivers.

The second method is similar to the Target Designer build processes. SIM has been modified into the Image Configuration Editor (ICE) to support WES7. ICE lets you create an answer file were you can pre-select the feature packages, device drivers, and add custom application and devices. You can build a custom IBW installation image from the answer file. The image can be installed from USB, DVD, or network.

After using either path you can tweak the operating system for your specific application. The image can then be packaged up for deployment.

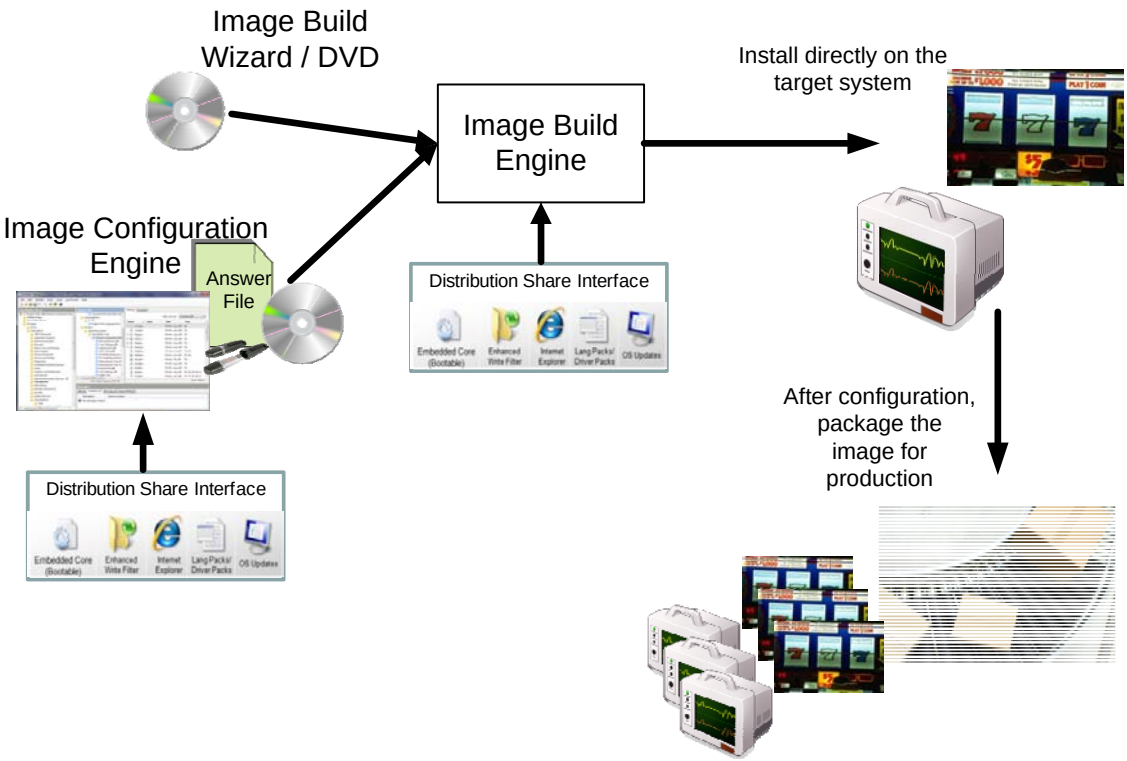


Image Build Wizard DVD Install
There is not much to the installation from the DVD. Insert the DVD and boot the target system. The wizard will walks you through various setup screens. Once the OS has been install, you must then perform all the manual operations to tweak the operating system for you application.

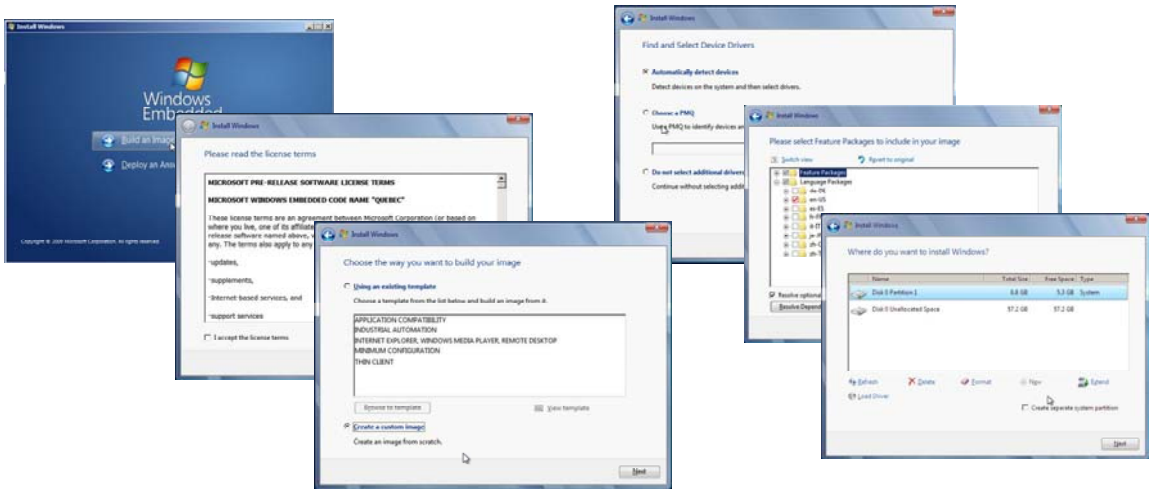


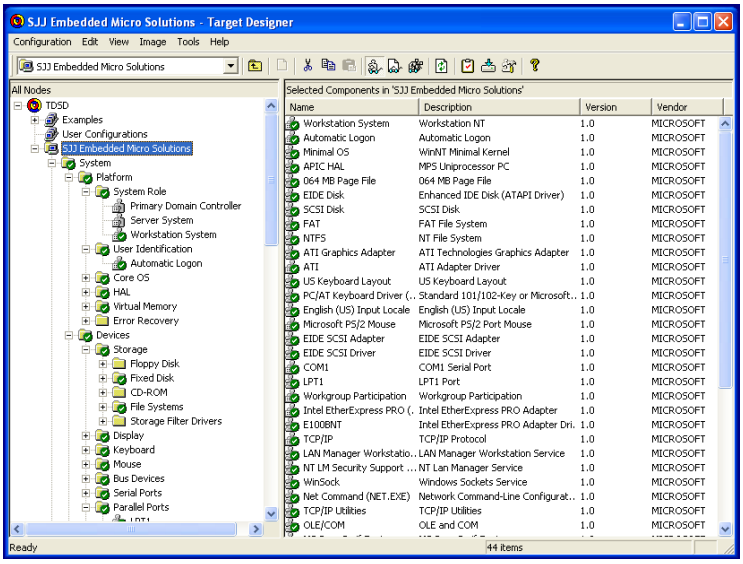
Image Configuration Editor

You might be tempted to think that installing the OS is all there is to embedded development; but when you have to do multiple manual operations, post installation setups, such as install device drivers, fonts, setup registry keys, etc, you start to look for a simpler solution. In my training classes, I not only stress that architecting the image is very important, but also how the image is built. I have had a few XPe customers come to me with their 40 page documents outlining multiple manual steps for how to install and configure the OS for the system. If you taking advantage of the Target Designer / Component Designer build tools and process, the manual post installation steps can be cut down to just a few manual post-FBA steps. This is where ICE comes in for WES7.

There is an old saying: “To know where you’re going, you need to know where you have been”. Before we cover the highlights of ICE, let’s look back at the previous OS build process for embedded:

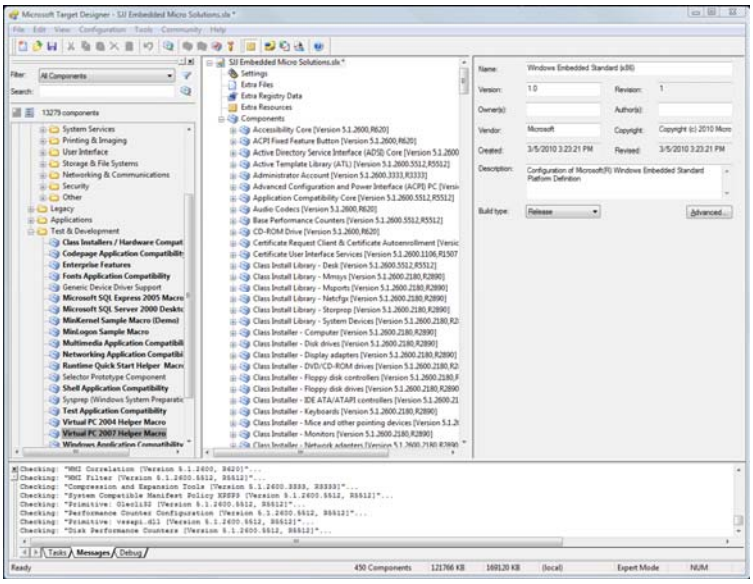
MS-DOS 3.31, 5.0, 6.22, Windows 3.1, and Windows 95, 98, ME were all installed on the target system directly from installation media. Once installed, you did the post installation tweaks. My old “Shrinking Windows 95” document discussed how to remove certain files from the Windows 9x installation to create a smaller image. I am skipping the MS-DOS 5.0 ROM version and the Windows 3.1 ROM versions since these required specialized PROM hardware.

Windows NT Embedded was the first targeted desktop OS with customization tools. Target Designer back then only had 150 components and very few drivers. It did support both desktop and server editions. You had to manually organize where component files were located in your file systems. Dependency check was quick. Building the image took files from the NTE CD and the local file system for any custom components. The final image was ready to run on the system. A first boot agent or long setup on first boot wasn’t required.



Windows NT Embedded Target Designer

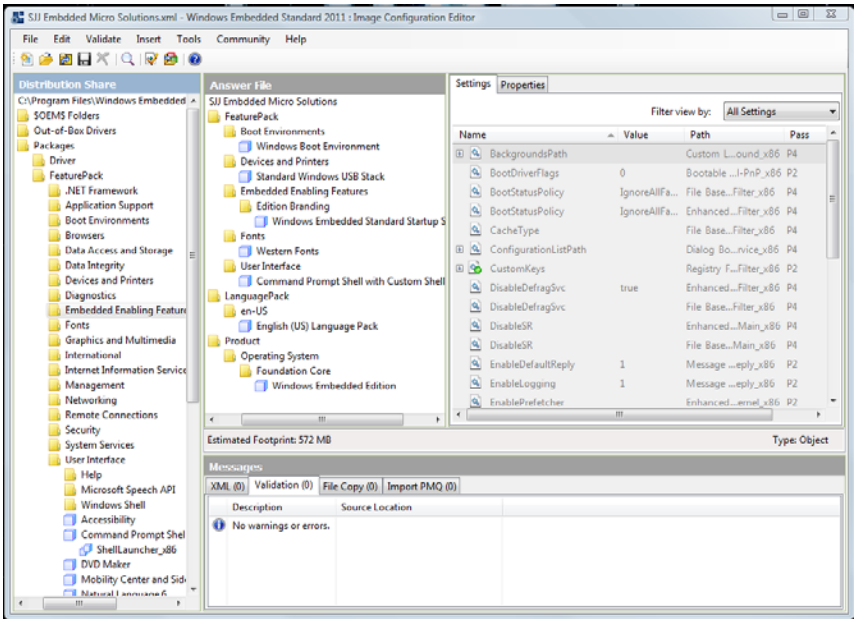
Windows XP Embedded continued with the Target Designer / Component Designer tools. The OS was broken down into small components, and there were over 11,000 components available. Most of these were device drivers. The component details were stored in a database so a database engine was needed to build images. As a result, dependency check, building the image, and importing special files took some time to process. The files for the OS and custom components were moved to a single repository. The repository concept removed the problems with NT Embedded custom components tied to a local file system. Plug-N-Play technology was new as more PC hardware supported USB and PCI busses. Plug-N-Play forced the build process to be broken into two steps: build the image with Target Designer; and once the image was on the target system, run First Boot Agent to locate all the Plug-N-Play devices.



Windows XP Embedded Target Designer

ICE takes a different approach. ICE helps you to create an answer file, which is used by IBW to install the image. There are two distribution shares containing all the OS files for 32-bit and 64-bit configurations. The OS files are stored in individual CAB files, and these CAB files are the packages. ICE is also used to manage the distribution shares with updates as they become

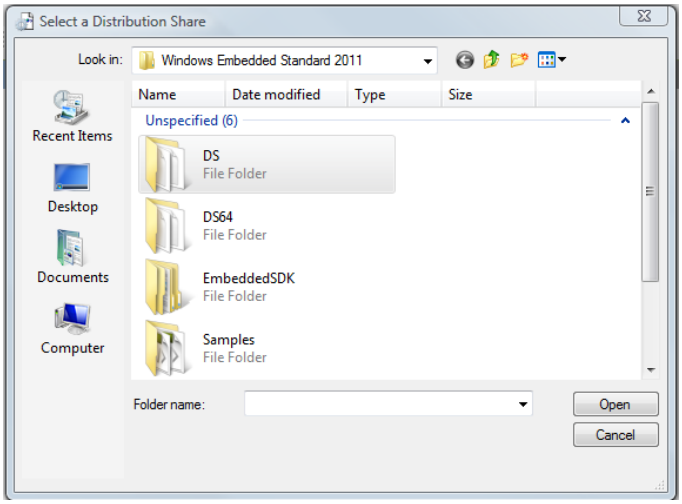
available. Unlike the two previous products, you cannot create your own package, but you can add files to the distribution share to build into the custom image. There is also the ability to add files from the local file system, but you will want to avoid this because of version control and backup. Applications, driver installers, and runtime installers can be setup in the answer file to run at different points in the installation process.



WES7 Image Configuration Editor

If you have ICE, let's walk through the steps to create a basic answer file with the minimal packages.

1. Open ICE
2. Connect to the 32-bit distribution share.



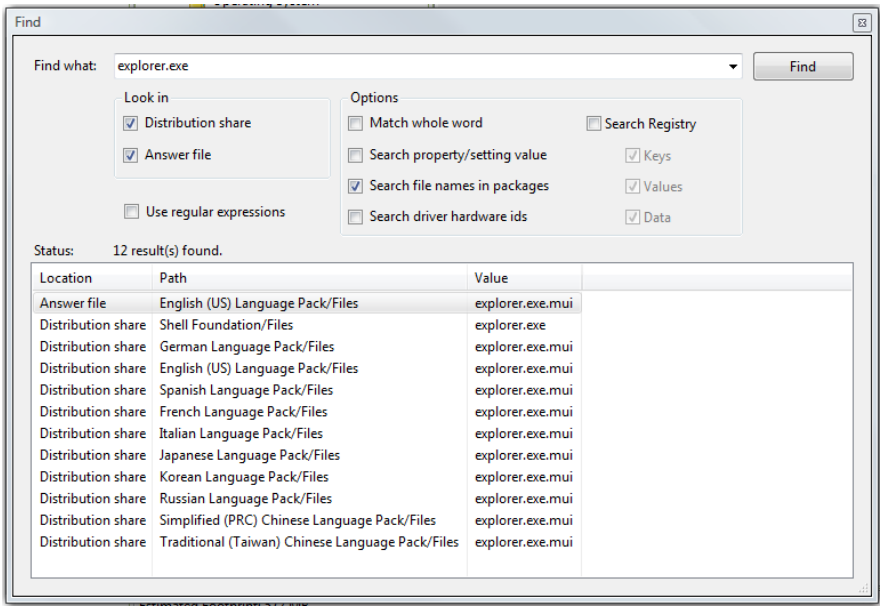
3. Create a new answer file. This will automatically add the Foundation core package: Windows Embedded Edition.
4. Add the following packages; these are the absolute minimum packages required for an answer file for a WES7 image.
   - English (US) Language Pack

- Western Fonts
- Command Prompt Shell with Custom Shell Support
- Windows Embedded Standard Startup Screens
- Standard Windows USB Stack
- Windows Boot Environment

5. Validation is the new Dependency Check, and there are 3 options. Choose Validate Only. The validation processes looks at the answer file and should report back no errors.

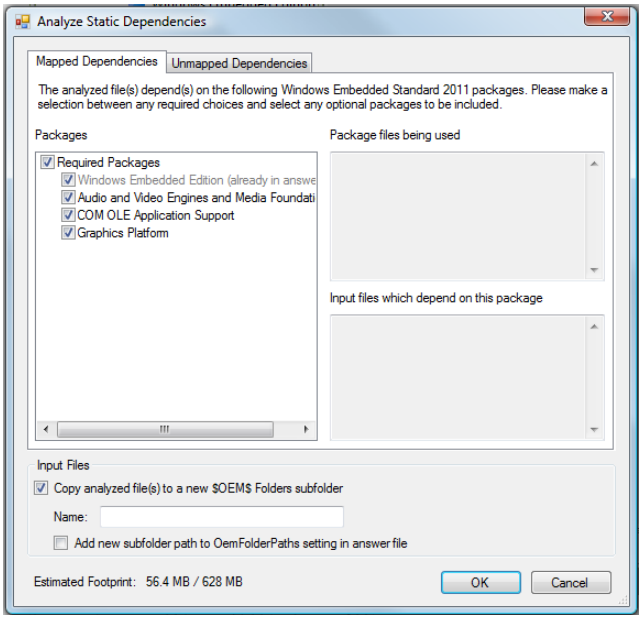
At this point, changes can be made to preset different settings such as autologon, EWF, FBWF, firewall settings, suppress OOBE screens, computer name, enter the license key, etc. There are up to 7 passes that an image can go through during installation, and certain settings go with certain passes. When you make a change to one of the settings something amazing happens, something that didn't happen in NT Embedded nor XP Embedded… the setting changes to bold to indicate that a change was made to the setting. WOW! It took 11 years to finally achieve this indicator. There are a couple more cool features.

You cannot see files and registry keys that go with the different packages like you could with components in the previous products. There is an advanced search tool that allows you to search on files, properties/settings, driver hardware IDs, and registry information. Best of all you can search not only the distribution share, but you can search the answer file itself.



Like components for XP Embedded, packages play a big role in WES7. As you dig through the tool and look at the feature packages, you will notice that certain files will go with certain packages. You might be tempted to remove files manually in the image to reduce the foot print, but you don't want to do this. By going with big packages, OS updates will be much easier since the updates are made into packages. You are able to view what packages are installed in the image, and you can upgrade, add, and even remove packages as well. Since updates will be made using packages, anything that is pulled out will only be put right back in; so you don't want to remove files.

Another cool feature is a built in DependencyWalker-like tool called Analyze Static Dependencies. You can perform a static analysis on an MSI, EXE, DLL, or zip file. The tool looks at the contents of the file being opened and looks for packages that the file will require. The results appear in a separate dialog box. You can then check off the required packages to add them to the answer file.



With the answer file validated for all package dependencies, you are now ready to build the image. In XP Embedded, images were built to a subdirectory. When the build process starts, the directory was cleared out of any files before the new OS image was put in place. In WES7, you have to manually pick the folder to build to and manually clean the folder for any new builds. The advantage is flexibility to pick different folders to build different installation images. The image that is built is an IBW disk that can boot from a USB or DVD. It can also be placed on a network server for remote installation. The build process of the IBW disk doesn't take that long, but it is still important to have a system with decent performance to do the builds.

Once you boot the target to run IBW, the installation on the target will take some time. The timing will depend on the features included and size of the image. The overall design goal was to minimize the different time consuming steps that XP Embedded had for preparing and building the images, and the new process is better than before.

New and Improved OS Updates

Several customers who have seen the new tools have noted that the tools are completely different from XP Embedded. One developer mentioned the tools and process are a bit backward. It kind of reminds me of Windows NT Embedded. Yes, there is a big difference, but keep in mind that thin clients are a driving factor for the product. The ability to maintain images becomes much easier.

As I mentioned earlier, using the OPK tools means that the desktop development cycle is being applied to embedded. Windows Updates can now be downloaded directly to a device, or individual update packages in the form of CAB or MSU files can be apply directly. Package information is stored in the image. Support utilities called DISM and WUSA provide the mechanism to manage the packages in the image so you can add, remove, update, etc.

Here is a dism.exe example that lists the packages in an image. The listing has been reduced for this article.

```
C:\>dism /Online /Get-Packages

Deployment Image Servicing and Management tool
Version: 6.1.7600.16385

Image Version: 6.1.7600.16385
```

```
Packages listing:

Package Identity : Microsoft-Windows-Embedded-LanguagePack-Package~31bf3856ad364e35~x86~en-
US~6.1.7600.16385
State : Installed
Release Type : Language Pack
Install Time : 3/8/2010 10:52 PM

Package Identity : Microsoft-Windows-EmbeddedCore-Package~31bf3856ad364e35~x86~~6.1.7600.16385
State : Installed
Release Type : Foundation
Install Time : 1/16/2010 12:25 AM

Package Identity : WinEmb-Application-UX~31bf3856ad364e35~x86~~6.1.7600.16385
State : Installed
Release Type : Feature Pack
Install Time : 3/8/2010 10:52 PM

Package Identity : WinEmb-Installers-MSI~31bf3856ad364e35~x86~~6.1.7600.16385
State : Installed
Release Type : Feature Pack
Install Time : 3/8/2010 10:52 PM
:
:
:
Package Identity : WinEmb-SystemManagement-Utilities~31bf3856ad364e35~x86~~6.1.7600.16385
State : Installed
Release Type : Feature Pack
Install Time : 3/8/2010 10:52 PM

Package Identity : WinEmb-usb~31bf3856ad364e35~x86~~6.1.7600.16385
State : Installed
Release Type : Feature Pack
Install Time : 3/8/2010 10:52 PM

Package Identity : WinEmb-WLMS-Package~31bf3856ad364e35~x86~~6.1.7600.16385
State : Installed
Release Type : Feature Pack
Install Time : 3/8/2010 10:54 PM

The operation completed successfully.
```

Maintain big blocks of the OS are much easier than meaning individual files and registry settings. Overall this is a big improvement compare to the previous products.

It is Finally Here

It has been a rocky road to get this new desktop OS to embedded, but it is finally here. Being able to take advantage of all the latest PC technologies and the new OS update process, WES7 is looking like a solid platform for building embedded systems.

Resources:

Windows NT Embedded Step-By-Step, Sean D. Liming, Annabooks, 2000, ISBN: 0-929392-68-X

Windows XP Embedded Advanced, Sean D. Liming, RTC Books, 2003, ISBN: 0-929392-77-9

Windows Internals, 5th Edition, Russinovich and Solomon, Microsoft Press, 2009, ISBN: 978-0-7356-2530-3