Hardware Independent Imaging

Version 1.4: Nov. 13th, 2007

Company profile

Enerplus Resources Fund was the first oil and natural gas income fund in Canada, established in 1986. Over the past two decades, Enerplus has grown from a $9 million initial public offering to having an enterprise value of over $6 billion today. With operations throughout Western Canada and in the United States, Enerplus is one of the largest oil and gas income funds in North America and is recognized for its leadership, innovation, technical skills and long-term financial performance.

Business situation

With over 800 computers to centrally manage across Western Canada and the United States, Enerplus required an efficient and cost-effective method of provisioning new computers and re-deploying existing computers while keeping up with new technologies. The chosen method of deployment was to develop and maintain a standard corporate Windows XP image.

Technical situation

Initial efforts to create a hardware independent corporate imaging process resulted in a disk image and process that was hardware agnostic for the Enerplus environment This process consisted of one fundamental image that initially supported all hardware platforms used by Enerplus. Plug and play hardware devices were detected and hardware drivers were installed and configured automatically through the Windows XP mini-setup program.

The image was prepared via a sysprep CD and image preparation batch file, which copied down the various drivers and utilities from the CD, ran sysyprep and then cleaned those folders from the PC after provisioning.

Subsequent to the initial image build, Enerplus incorporated a new model for desktop PCs which contained a Hyper-Threading Pentium 4 CPU. This was interpreted as a multiprocessor system, and as a result, a different HAL was required for the image. In addition to the HAL, there were also different driver requirements for video, audio, NIC and Chipset. These were not included in the original image build. At the time it was determined necessary to create a new (second) image for this platform.

The method of deployment was via BartPE boot CDs and USB drives. After deployment, additional drivers and software were installed manually, and various post-imaging configuration tasks were run, either through scripts or manually. Though this image build and deployment process was designed to be hardware independent, it quickly became outdated and labor-intensive. Particularly troublesome were laptops, which required, on average, an additional 2 hours of post-imaging driver and proprietary software installation.

As new models were introduced into the environment, increased time was required for testing and engineering in order to ensure the image would load properly and all drivers were available to be installed after the imaging process. When Toshiba introduced a laptop model with a SATA disk controller, the image blue-screened and would not load because it could not recognize the hard drive. As other models came on stream with similar, non-native SCSI, RAID or SATA mass storage controllers, the image quickly became outdated and a rebuild was required.

These and other driver-related issues ultimately contributed to hundreds of man hours in research, trouble-shooting, testing and reconfiguring of the Enerplus corporate image.

Solution

The existing Hardware Independent Imaging process proved to be inadequate in that it was unable to support the inevitable introduction of new hardware platforms, thus it quickly became outdated and required a manual process to install drivers as part of a post-imaging task list. This method was time consuming and prone to error. This approach also made no provision for peripheral device drivers which were a factor in reimaging existing user systems At that time Enerplus reviewed the current process and considered the following alternatives:

Method #1: Scripted Hardware Independent Imaging using Altiris Deployment Solution

The first method that was analyzed and tested was the use of Altiris Deployment Solution for the provisioning and deployment of PCs at Enerplus. Using the document **HII_BestPractices** available from Altiris, the entire process was built and tested extensively.

The process involved the following steps:

1. Create a tokenized sysprep.inf answer file to be used in the packing and mini-setup portion of sysprep.
2. Obtain all drivers needed for all supported models. This involved downloading drivers from manufacturer's websites and then organizing drivers for each model into a specific folder structure on the Deployment share.
3. Build the reference computer from which to capture the image.
4. Create Altiris Deployment Solution jobs to accomplish the following:
    (a) Prepare the reference computer for imaging by running sysprep with the –mini –quiet –reseal –noreboot switches.
    (b) Capture the image from the reference computer.
    (c) Deploy the image back to targeted computers. The steps involved in that process are as follows:
        (i) PXE boot and deploy the image using Altiris PXE Server and Altiris RDEPLOY.
        (ii) Token replacement and copy of the sysprep.inf file from DS share to C:\sysprep
        (iii) Copy model-specific drivers from DS to client PC using a tokenized FIRM copy script.

This method is a significant improvement over the previous method. It leverages Altiris Deployment Solution's automation capabilities and introduces a much higher degree of customization to the Hardware Independent Imaging process. However, it still presents the following limitations:

1. The process of matching driver packs to platforms is model-specific. Two methods are recommended for collecting drivers – one is to download them from the manufacturer's website, the other is to collect them from a representative machine using WinDriver Ghost. Both methods were utilized at Enerplus and both proved to be inadequate to fully represent the spectrum of platforms deployed across the enterprise. Model-specific driver collection is based on an assumption of what "should" be there. However, many times two instances of the exact model will present significant differences in devices and the accompanying drivers.
2. The driver collection process is entirely manual and involves significant time and effort, as well as continued maintenance as new platforms come into the environment.
3. The tokenized FIRM copy script (if "%#!computer@model_num%" =="0K5148" goto DIM2400)that copies model-specific drivers down to the targeted computer is subject to continued maintenance as new models are introduced. This involves querying the eXpress database to get the model number of the desired computer – which, in some instances is not sufficient because certain manufacturers (Toshiba for instance) don't

create unique entries for each different model. This creates a situation in which different tokens are required for different models based on where there is unique model-specific data available in the eXpress database.

The final result using this method still required manual effort in order to fully install all devices and associated drivers, meaning that computers would boot into the Windows XP OS with devices detected but uninstalled. The desired outcome of an automated, scalable and customizable Hardware Independent Imaging process was not achieved.

Method #2: Hardware Independent Imaging using Altiris Deployment Solution and HII Tools

HIITools is a set of sample jobs and tools designed to make it easy to create and distribute Hardware Independent Images using Altiris Deployment Solution. The installation creates a folder on the DS share called HII which contains sample jobs and executables designed to handle the collection and sorting of driver files. This process was analyzed and tested as a potential candidate for implementation as the definitive Hardware Independent Imaging process at Enerplus. The process involved the following steps:

1. Install the HII Tools sample jobs and associated files to the Deployment Server by running HIITools.msi on the DS.
2. Download Microsoft Target Analyzer, a program that analyzes a system to determine what devices are on the target system, as well as what HAL is required. It produces a PMQ file that is parsed by the HIITools utility called “DriversPrep” to collect appropriate drivers for the target system.
3. Collect HAL files from the Windows XP install CD.
4. Create a tokenized sysprep.inf file using the MakeSysprepINF.exe program included with HII Tools.
5. Configure the “Capture Drivers” job included with HII Tools. This does the following:
    a. DriverCollect gathers drivers from the target system into a folder under .\Temp, which is named using the ID of the target system.
    b. DriverSort then moves those drivers into their destination folders to be used later by DriverPrep.
6. Configure the Create and Distribute image jobs included with HII Tools.
7. Run the Capture Drivers job on selected candidate computers, which does the folllowing:
    a. The job is run on one of every model computer present in the environment. Each time it is run, drivers are captured for each device and then sorted for later use.
    b. The driver collect process updates HII\<OS>\syspmsd.inf with the Mass Storage Drivers. It is necessary to replace the [SysprepMassStorage] section of the sysprep.inf file(s) with the contents of this file after the Collect Drivers job has run.
8. Create and capture a base image from a reference computer.
9. Deploy the image using the Distribute Hardware Independent Image job in Altiris DS. This does the following:
    a. Target Analyzer (TA/TAP) is run to determine the hardware required. This creates the PMQ file.
    b. DriverPrep is run against this PMQ file and pulls the necessary files you collected earlier from the folder structure to a temp folder under the computer ID.
    c. FIRM is then used to copy all the necessary drivers to the newly installed image prior to running Windows for the first time.

In theory, this method is an improvement over any of the previous methods. It uses Target Analyzer to determine which hardware devices are actually present on the system and then copies those drivers to the targeted computer prior to Windows first boot. It harvests drivers from systems currently in production, thus ensuring a more accurate and complete driver repository. It also attempts to address issues with Mass Storage Controllers by updating the

[SysprepMassStorage] section of the sysprep.inf as it discovers various mass storage controllers in the environment. However, it was not chosen as the implemented solution for a Hardware Independent Imaging process due to the following limitations:

1. The MakeSysprepINF.exe program included with HII Tools does not work. Running the application throws an error “Application has generated an exception that could not be handled”. Therefore the sysprep.inf file had to be created manually.
2. The “Capture Drivers” job only partially works. It collects the drivers but will not run the DriverSort portion of the job through DS. A manual driversort.bat file is included and this was run on the DS, which involved creating multiple instances of this batch file and changing the entries within the batch file for each model that was collected. Ultimately this did successfully sort the drivers in the HII folder on the DS share.
3. The driver sort process uses up a very large amount of disk space. HII Tools creates a folder for every device ID instance in every inf file that it finds within the collected drivers, and then copies a unique instance of that driver pack for each distinct folder. This resulted in over 3 GB worth of drivers for only 3 models that were collected and sorted.
4. By running DriverPrep against the output of the Target Analyzer (a PMQ file), HII Tools will copy the appropriate driver folders down to the newly installed image. What this amounts to is using the file system as a database – creating folders during the driver sort process with names such as VEN_10B9&DEV_5451&SUBSYS_102B13BD enables HII Tools to then compare that to the contents of the PMQ file and copy over the necessary folders and drivers using FIRM. This is not an efficient usage of disk space and fails to use the existing eXpress database for these kinds of associations. As model diversity increases, the folder becomes bloated and difficult to manage.
5. Each collection of drivers is based on the judgment of the Altiris administrator making a “best guess” as to which systems are proper representatives of the various models in the environment.
6. Because the automated driver sort job did not work through DS, it necessitated an ongoing manual maintenance of the driver collection and sorting.

HII Tools is a free, unsupported set of sample jobs and utilities resulting in minimal proper documentation or support. The final result using this method resulted in many devices left detected but uninstalled, necessitating a manual process to install missing drivers after imaging. Thus, HII Tools was discarded as a viable option for Hardware Independent Imaging at Enerplus.

Method #3: Hardware Independent Imaging using HIIS (Hardware Independent Imaging Solution) by Altrinsic Solutions and Altiris Deployment Solution.

The third method that was analyzed and tested was Hardware Independent Imaging Solution or HIIS, developed by Altrinsic Solutions. HIIS is a process management environment that works in conjunction with Altiris Deployment Solution to fully automate the creation, deployment and maintenance of a Hardware Independent Image process. The HIIS tool installs onto the Deployment server in a console form that utilizes an intuitive, GUI, step-by-step process to automate the image preparation, hardware enumeration and grouping, driver collection, deployment job scripting and maintenance phases of the Hardware Independent Image process.

HIIS represents a vast improvement over all the previous methodologies, in that it comprises a complete, integrated solution designed for scalability and maximum automation, using some of the most powerful features of Altiris Deployment Solution. HIIS was analyzed and tested, and ultimately implemented as the chosen tool for the Hardware Independent Imaging process at Enerplus.

HIIS works via the following 4 processes:

1. Identify

    a. Analyzes the eXpress database for device information that is uploaded by the aclient as part of the default inventory.
    b. HIIS then sorts PCs into compatibility classes based on device level analysis – model specific hardware profiles are irrelevant because HIIS is looking for *exact* matches and sorting PCs based on what's really there as opposed to what "should" be there. HIIS analytics at Enerplus resulted in 197 compatibility classes for 800 managed PCs. This was a true representation of the platform diversity throughout the enterprise as opposed to a model-specific analysis which would have resulted in roughly 24 different models in the production environment.
2. Collect
    a. Candidate computers are chosen from each compatibility class – this can be done either automatically or manually (candidates selection done by the "auto" function can be overridden by the administrator if, for instance, the auto selected candidate was over a WAN link or otherwise not as desirable as another from the same class).
    b. These candidates are then harvested for their drivers and HAL files, which are in turn uploaded to the DS share. The collect process (collect.exe) runs on the candidate PC as a low priority thread so it is virtually undetectable to the end user.
    c. HIIS then sorts these drivers into a master driver repository, ensuring that only one copy of each unique driver is stored on the Deployment Server, resulting in extremely efficient disk usage.
3. Export
    a. HIIS provides a GUI interface to create a tokenized sysprep.inf file that will be used in the image creation process.
    b. Injection Management allows for the selection of Mass Storage drivers to be included in the sysprep packing job, ensuring that specialized SATA and SCSI disk drivers are available right in the image.
    c. Injection Management also provides a method for associating collected device drivers (if manual override of auto-collected driver is desired) and applications with compatibility classes. Thus, devices that require an accompanying application (ie fingerprint reader) will be slipstreamed with the application and deployed as part of the imaging job.
    d. HIIS automates the image creation and sysprep packing job, which is run on the reference computer, and then uploads the image to the Deployment Server.
    e. HIIS then creates a DS deployment job that blends a system-specific driver pack, HAL files and associated applications with the core image. Since this job uses custom tokens and complex user defined functions within the eXpress database to reconcile all the driver and application associations with the targeted computer, only one job is required to cover the entire enterprise.
4. Maintain
    a. The maintenance job runs analysis and collection tasks on a set schedule to ensure that new platforms are incorporated into the process automatically.
    b. All processes are policy driven and all deployment jobs leverage the eXpress database, which ensures that all PC and device changes are integrated into the HIIS deployment process. If a device is added to a PC (ie a USB device or new video card), HIIS will recognize that and update the compatibility classes accordingly. If enabled, the maintenance schedule will collect updated driver packs for new compatibility classes.

The image was recompiled to include Mass Storage Controllers, was captured and uploaded to the DS share. Application injection was customized for various devices and driver packs (typically audio and video drivers that have accompanying applications). The deployment job was created and then tested against a variety of platforms, including models with Multiprocessor HALs, SATA and RAID controllers, and laptops with a variety of specialized devices.

In every case, the image deployed successfully and all devices were installed, resulting in a low-touch Hardware Independent Image deployment process. Using application injection solved the occasional instance where a device would not fully enumerate using Windows Plug and Play – by associating the setup.exe file for the driver in question with the correct compatibility classes, and specifying the silent command line switch, Enerplus was able to ensure that all drivers loaded properly at mini setup, leaving no device detected but uninstalled.

Benefits

The resulting benefits Enerplus received through the use of HIIS as our chosen solution for implementing a Hardware Independent Imaging process are as follows:

1. Significant reduction in time required to provision and deploy new PCs. The use of Altiris Deployment Solution in conjunction with HIIS has even further reduced the man hours required for provisioning new PCs by automating many of the post imaging configuration and software deployment tasks.
2. HIIS automates the maintenance of driver collection and compatibility class refreshes, and as a result there has been a significant reduction in the engineering and research hours required to maintain up-to-date driver packs.
3. HIIS leverages the eXpress database and stores associations between individual PCs and their required drivers and applications, eliminating the need to maintain and update multiple scripts for image deployment. One single Hardware Independent Imaging job accomplishes everything required to support all platforms, again reducing engineering and maintenance hours.
4. New models are quickly and easily incorporated into the Hardware Independent Imaging process.
5. Application injection has added value by providing further automation of traditionally labor-intensive processes. Devices requiring both a driver as well as an accompanying application can be slipstreamed to include both the driver and application as part of the automated image deployment process. As well, devices that do not fully enumerate using Windows Plug and Play can be installed silently from their accompanying setup.exe or .msi file during mini setup.
6. Support for Mass Storage Drivers and core OS files has resolved many blue-screen issues associated with mismatched HALs and missing Mass Storage drivers.

Summary

HIIS was the chosen tool for the implementation of a Hardware Independent Imaging process at Enerplus because it presented by far the most comprehensive, well-designed and effective tool for accomplishing the goal of an automated, scalable and customizable Hardware Independent Imaging process. HIIS automation represents an estimated 75% reduction in the time required to develop and maintain a hardware independent imaging process at Enerplus.