Modular Imaging Architecture – A Modular Approach for Scalable PC Deployment

Josh Kelahan
Saint Louis University
Information Technology Services
St. Louis, MO 63104
+1-314-977-3047
jkelaha1@slu.edu

Chris Koerner
Saint Louis University
John-Cook School of Business
St. Louis, MO 63104
+1-314-977-3600
koernerc@slu.edu

ABSTRACT
One of the difficulties a university IT department faces is trying to design and support the image for a multitude of different computer systems. Saint Louis University currently supports more than ten different computer models. Each model has its own drivers that can conflict with the drivers of other models. In addition, trying to keep up with the endless Windows updates and the updates for all the other software needed on an image can prove to be an extremely daunting task, even for highly skilled IT professionals.

Members of Saint Louis University's Enterprise Infrastructure Working Group have developed a new imaging architecture that separates the large components of an image into smaller, manageable chunks. Each chunk is its own separate image and is stackable with other images. When these images are assembled together they create what would traditionally be the image deployed to all workstations.

By having separate driver images, there is no longer a concern for drivers of different models conflicting; each model can have its own driver image. Having the base operating system as its own image, all of the Windows updates as an image, and all of the enterprise required software as an image, each of these seemingly large components can be managed separately. This Modular Imaging Architecture goes a long way to help improve security by ensuring that every time a machine is imaged, it will always have the most up-to-date versions of enterprise software, Windows updates, and the latest drivers for that system.

.

Categories and Subject Descriptors
K.6.4 [**Management of Computing and Information Systems**]: System Management – *Centralization/decentralization*

SIGUCCS'08, October 19–22, 2008, Portland, Oregon, USA.

General Terms
Management, Design, Reliability, Security, Standardization.

Keywords
Image management, deployment, Imaging, administration, Modular, Monolithic, Windows, Operating System, Novell, Linux, updates, enterprise, drivers, Saint Louis University, Singlecast, Multicast, Master, Client, installation, configuration.

1. HISTORY IN DESIGN

1.1 Ancient Wisdom
In the beginning there was Windows. To install Windows, you put in the disk, stepped through the prompts, waited a very long time, then installed updates, applications, etc. This was the way to setup a computer. This process is very tedious, even with today's faster computers, installing and configuring an operating system by hand takes a bit of time. But there are advantages to installing your OS by hand. Namely, you know that everything is working, that all of the drivers are there, that all of the applications are running properly, and that the system is completely up to date.

Installing an OS by hand can be a major pain if there is more than one computer. True, you will know that everything is working on these machines, but somewhere between computer three and computer three hundred you will get sloppy. You will miss a step, or something won't install all the way and you won't notice. Further, when an application or new updates come out, you will probably go with those, and your computers will no longer be consistent. When it comes to multiple computers, installing the system by hand just isn't the best way to go.

1.2 Conventional Weapons
For more than a decade, the de facto standard for keeping a large number of computer systems in an enterprise running the same operating system, with the same settings from computer to computer has been through the use of disk cloning applications such as Symantec Ghost. The majority of disk cloning applications work by creating a snapshot of the entire contents of a hard drive or partition. They copy the Master Boot Record, the file system, the partition table, and all of the data on the disk to a compressed image file that can be restored to a different hard disk at a later time.

The image file created by modern disk cloning applications, can be restored to single pc (Singlecast) or to multiple pc's at once (Multicast). This is a tremendous time saver for system administrators whose time can be better spent not installing operating systems on hundreds if not thousands of computers. But it's not just the operating system that ends up on the image; the image also contains all of the programs that need to be on all of the workstations. This would include things like anti-virus software and patch management software. The image would have to contain all of the Windows Updates. Every time new updates come out, the image has to be pulled down, modified, and put back up. Finally a standard image also contains the drivers for all of the computers the image needs to run on. The more models of computer, the more complex the image will be.

1.3 Trouble in Paradise

The problem with conventional imaging comes down to supporting so many different types of machines. If all the computers an enterprise had to support were the same model, the creation of the image would be simple. The image could be created on one of the computers, and no one would ever have to worry about drivers. But what happens when an enterprise has multiple models of computers? What happens when an enterprise has to support different architectures, different chipsets, and they all need to run the same image? Still, the management of two or three images in not terribly difficult, as long as they are basically the same there shouldn't be much of a problem recreating steps.

Enterprises that have multiple departments, each with different needs, different architectures, maybe different locations, with different software requirements can easily find themselves in situations where there may be a need to support ten, twenty, or more different computer models, each with its own set of drivers. This task can be so daunting, even a team of IT professionals might have difficulty keeping everything going. When an image has so many sets of drivers, you run into situations where computers can pick up the wrong version of a driver, especially if two computer models are very similar to each other. This is the primary cause of difficulties IT departments have with using a single image to deploy and manage their workstations.

2. WHAT IS MODULAR IMAGING?

2.1 Building Blocks

Monolithic imaging (conventional imaging) is the practice of putting everything on one image. This would include (but is not limited to) the operating system, updates, applications, drivers etc. Modular imaging on the other hand refers to the separation of these core pieces into individual images. Each of these individual images is stackable, one on top of another. The separation of these portions of a traditional workstation build provides for better scalability, security, standardization, and reliability than building from scratch or using a single image. Because of this key difference, it becomes very easy to manage so many different models of computers because they can all run the same resultant set of images (except for the drivers of course) known as the Deployed Image.

2.2 Climbing a Mountain

As enterprises grow, they naturally accumulate new computers. It falls of course to the IT department to try to keep new computers and old computers running together in the same environment. This means that all of the computers have to use the same image. Because drivers are in their own image in modular imaging, an unlimited number of computer models can be supported.

The scaling of modular imaging is therefore most like building out a workstation from scratch, where it can be ensured that each workstation will have all the correct drivers at image time. This approach also streamlines the sysprep/mini-setup process. There is no need for mini-setup to scan hundreds of drivers, the exact drivers are provided for the system.

2.3 Securing the Enterprise

Security is something that doesn't get thought about enough when it comes to creating an image. This probably has to do mostly with the way Windows was originally designed. Because of this indiscretion, the problem falls on the shoulders of IT professionals. One of the easiest ways to help secure a system is to install the critical updates from Microsoft. With modular imaging, all of the updates can be separated out into their own image. This makes the updates much easier to manage, as new updates can be easily added to the updates image. This ensures that every time a workstation is imaged, it has the most up to date security.

2.4 Stand and Deliver

Modular imaging enforces standards because it has to build out the resultant set of images every time a machine is imaged. This forces all computers to work exactly the same way. Sure the images can individually be modified, but each machine still receives the core set of images (windows updates, enterprise applications, etc).

When standards are in place, things tend to be much more reliable. This holds especially true for modular imaging. Because each machine at imaging gets the same set of images in a well documented order, there can be no surprises that would otherwise cause instability. Testing also becomes very easy, since all computers run the same deployed image each machine is the same as the next one.

One big advantage is that because the modular has everything segmented out into different images, updating those images is much easier. If a new update comes out, it can be added to the updates image very easily. There is no need to pull down the full Windows image as with monolithic imaging, and so there is a greatly reduced chance for errors in this regard.

3. BUILDING A NEW ARCHITECTURE

3.1 Exploding the Image

During the transition from a distributed IT environment where every department managed their own areas to a centrally managed environment, technical staff at Saint Louis University ran into a huge problem. Spread out across campus, no less than 27 different computer models needed to be supported. Integrating the driver sets for all 27 models was nearly impossible. At the same time, Apple was releasing their first beta of their Boot Camp product and many users around the university were asking to dual boot their Macintoshes.

Boot Camp allowed Macintosh computers to run a native Windows XP installation. While attempting to discover a way to image a Mac with our ZENworks Boot CD and monolithic base

image, we started discussing how to include drivers for the specific hardware in the different Mac models. It was apparent that updating our existing infrastructure to include the various models would be time consuming and had the potential to inadvertently break our existing Windows XP image.

We then began discussing some of the inherent flaws with the Monolithic imaging process and how having one image to rule them all was not as scalable as we had hoped. The group researched the availability of add-on images using the Novell Imaging Engine. These add-ons would allow for not only the separation of Mac and PC model specific drivers, but other components of our enterprise image as well.

Once it was known that we could stack images one on top another and create a single image, the question became, what pieces do we want to have in separate images. The obvious is the pieces of enterprise required software and the drivers for individual machines. Also it was decided, one image for just Windows XP SP2, and images for each HAL (single/multi core). These four images together combine to create the standard deployed image along with the option for an optional departmental add-on image.

3.2 Image Types

3.2.1 The Base Image

The base image contains just Windows XP Service Pack 2 and all of the Windows updates up to the day it was created. The idea here is that once this image is created, it never has to be modified. All updates can be installed via a different image. This keeps the modular imaging architecture secure, stable, and reliable because this file never has to change. This image is created by installing the operating system on a pc, but not installing many drivers. After the installation is perfected, the infcache can be dumped before sysprep is run, effectively removing the drivers that won't be needed for most other systems. Updating this image would be difficult because you would have to drop it on a machine, make modifications, sysprep it, and put it back. Modifying this image, just like modifying a standard monolithic image can be dangerous. If the file structure, partition table, or system registry becomes corrupt, the image will not work, and will have to be rebuilt. The other images can be modified by using a simple drag and drop interface with Novell's Image Explorer.

3.2.2 Windows Updates Image

The Windows updates image as its name specifies contains all of the Windows Updates since the base image was created. A simple batch file in this image runs during mini-setup, which searches for the Windows Update executables found in a folder also on this image. The updates are then installed one after another. When the computer restarts it should be completely up-to-date patch wise.

3.2.3 Enterprise Agents Image

The enterprise agents image contains the software necessary to run the system on the network. At Saint Louis University, this includes the Novell ZENworks pieces: the ZENworks for Desktops Agent, the Novell Client, and iPrint; Patchlink, the patch management system for already imaged workstations, and an anti-virus application. These are all applications, which are not necessary to the computer but allow the computer to work in a specific way, and are identical from computer to computer. These applications are better here than say in the base image because they will change over time. The new executables can be dragged into this image, and there is no need to pull down the base image. The applications install during mini-setup via a batch script that is included in this image.

3.2.4 Drivers Image

At Saint Louis University, we currently have twenty-seven different driver images. Each image contains the driver set of one computer model. The structure of the image is very simple, organized by component: chipset, nic, video, etc. During the imaging process, the imaging script determines the model of the computer, and pulls down the appropriate driver image file.

3.2.5 The HAL Image

The HAL or Hardware Abstraction Layer image consists of the specific files that are different from a computer running on a single core system to a dual or multi-core system. At the end of the imaging process, the imaging script determines which HAL is appropriate for the computer, and applies that image. The HAL is separate from the driver image because different revisions of some computer models can have different processors, and thus can require a different HAL.

3.2.6 Optional Departmental Add-On Image

The departmental image (optional as the name suggests) can be used to inject files and folders onto a workstation at imaging time. This can be useful for department specific add on hardware such as a printer or simply getting a bunch of files onto a workstation. While it could be used to install additional software, just like the Enterprise Agents image, it is not recommended. Pushing application packages to the workstation is much more effective for departmental level support needs. If the departmental image contains a batch file it is run during mini-setup before other batch files from the other images. The idea here is to enforce content coming from the enterprise, and ensure that a department cannot override enterprise required settings using a script.

3.3 Preparing the Images

The Modular Imaging Architecture relies heavily on the leveraging of Microsoft's sysprep/mini-setup tool. As noted, each of the add-on images (Windows Updates, Enterprise Agents, Drivers & Departmental) contains a batch script that executes during the mini-setup process. In the base image, there is a batch file that runs at the end of sysprep, that looks for the batch files in the add-on images, and runs them if they are present.

All of the add-on images are created using a simple image editing tool called Image Explorer. The tool allows for the dragging and dropping of files and folders into an image.

4. CUSTOMIZATION

4.1 Linux Imaging Engine

The imaging script is contained in a bash script executed from a very small distribution of Linux that uses the isolinux/pxelinux bootstrap utility. Because the imaging script is written in bash, it is infinitely customizable. In addition to the actual imaging, the engine is capable of reading from an SMI compatible BIOS to get key system information such as the Model of the computer (used in determining which driver image to use) and the type of processor (used in determining which HAL to use). The script can even go so far as to see if the BIOS needs to be upgraded before imaging can continue.

4.2 Distributed Customizations

The imaging script reads a file, settings.txt, at boot time, that contains a group of departmentally defined variables. These variables tell the imaging engine whether to use a particular modified base image, which imaging server to use and if and where the engine can find a departmental add on.

5. TECHNOLOGY

5.1 Imaging Workstations

Using Novell's Imaging Engine, members of Saint Louis University's Enterprise Infrastructure Working Group (EIWG) were able to create an environment where we can stack images one on top of another to create the Modular Imaging Architecture.

5.1.1 Single and Multi-cast

Modular Imaging can be used for either singlecasting, imaging one machine at a time, or multicasting, imaging a bunch of computers at once. The difference (other than the obvious) is the order in which images are applied.

5.1.1.1 Singlecast

Singlecast applies all the images that make up the deployed image in this order: 1. Base Image, 2. Windows Updates image, 3. Enterprise Agents image, 4. Model specific Driver image, 5. Workstation specific HAL image, 6. Optional Departmental image. After these images are applied, the script sets the computer name and restarts the system. At this point imaging is complete, and mini-setup runs. As previously noted, during mini-setup, batch file scripts contained in the Departmental, Windows Updates, and Enterprise Agents images are executed to complete the installation.

5.1.1.2 Multicast

A multicast session requires at least two computers. One computer has to be the master, and the rest of the computers are the clients. When starting the multicast, the user must select whether the system is to be a master or client.

5.1.1.2.1 Master

The master begins the imaging process by pulling down the following images in this order: 1. Base Image, 2. Windows Updates image, 3. Enterprise Agents image, and 4. Optional Departmental image. Once these images are on the workstation, the master creates a session, and waits for clients to connect. Once connected, an administrator must initialize the multicast. This conglomeration of images is then pushed out to the client workstations that have joined the session. After the master finishes pushing to the clients, it downloads the drivers and HAL images before the imaging is complete for the master workstation.

5.1.1.2.2 Client

The client begins by waiting for the multicast session to start. During the session, the client receives one image file, which is the resultant set of the Base, Windows Updates, and Enterprise Agents images. After this multicast session completes, all of the clients individually download and apply their model specific driver images, and workstation specific HAL images. This setup allows an administrator to image multiple computer models at once, and still ensures that each computer gets the proper drivers.

5.1.2 Applying an Image to a Workstation

Applying an image to a workstation is very easy with Novell's Imaging Engine. The command can be encapsulated inside a function in a bash script. This allows for the stacking of the images. If two image files contain the same file, the file in the last image applied will be the one on the machine when imaging is complete. The command requires an image server, a proxy server and of course, an image file.

```
restore_image() {
  img rp $IMAGE_PRXY
\\$IMAGE_SRVR\path_to_file\$IMAGE_FILE
  if [ $? = 1 ] ; then
    echo "Image $IMAGE_FILE.zmg restored"
  else
    EIWG_ERRORCODE=1
  pause
  fi
}
```

5.1.3 Discovering the Workstation Model

One of the core functions in the Modular Imaging Architecture is the ability to have a separate model specific Driver image for each model computer that is supported. The drivers on a particular Driver image pertain only to that model, and shouldn't be used for any other model computer.

The model of the computer can be retrieved by probing the BIOS of the workstation. This information is stored differently depending on the manufacturer of the workstation, so it is important to check first for who makes the workstation, and then check what the model is.

```
driver_image_addon() {
  #Different manufacturers store the model differently
  MANUFACT=`hwinfo --bios | grep -A 5 "System Info" | grep
"Manufacturer" | awk 'BEGIN {FS = "\""} {print $2}'`
  case $MANUFACT in
  "IBM")#Its an IBM
      SYSTEMTYPE=`hwinfo --bios | grep -A 5 "System Info" |
grep "Version" | awk 'BEGIN {FS = "\""} {print $2}' | awk 'BEGIN
{ FS = " "} {print $1$2}'`
  ;;
  "LENOVO")#Lenovo
     SYSTEMTYPE=`hwinfo --bios | grep -A 5 "System Info" |
grep "Version" | awk 'BEGIN {FS = "\""} {print $2}' | awk 'BEGIN
{ FS = " "} {print $1$2}'`
  ;;
  *)#Everything else
      SYSTEMTYPE=`hwinfo --bios | grep -A 5 "System Info" |
grep "Product" | awk 'BEGIN {FS = "\""} {print $2}' | awk
'BEGIN { FS = " "} {print $1$2}'`
  ;;
  esac
  SYSTEMTYPE="path_to_file/$SYSTEMTYPE"
}
```

5.1.4 Discovering the HAL

Similar to determining which driver image to use, the BIOS again needs to be probed to determine which HAL is most appropriate. In general, there are only two HAL's for PC workstations. One HAL is for systems with multiple CPU's or multiple cores. The other HAL is for systems with only a single, single cored CPU. There is also a separate HAL for VMware virtual workstations as well.

```
get_mp_hal() {
  if [ `hwinfo --smp | grep -ic 'Yes'` = "1" ]; then
    img rp $IMAGE_PRXY
\\$IMAGE_SRVR\path_to_file\add-hal-multi.zmg
  else
    img rp $IMAGE_PRXY
\\$IMAGE_SRVR\path_to_file\add-hal-acpi.zmg
  fi
}
```

5.1.5 Creating the Base Image

When an administrator creates the base image, he has to install the operating system from scratch, and get the workstation into working order without installing too many of the drivers. Afterwards, the administrator can delete the infcache (effectively removing the unwanted drivers, and sysprep the machine).

The Modular Imaging Architecture provides to create images as well. Again, the code is encapsulated so the administrator does not really have to think about what he has to do, he can simply run the function from the menu provided by the modular imaging architecture.

```
make_zen_image() {
  get_image_name
  img makep $IMAGE_PRXY
\\$IMAGE_SRVR\path_to_fileIMAGE_FILE.zmg
  if [ $? = 0 ] ; then
    echo "Image $IMAGE_FILE.zmg created on
$IMAGE_SRVR"
  else
    echo -e "Image creation
${RED}FAILED!${MAINCOLOR}"
  fi
  pause
}
```

5.1.6 Getting the Serial Number

At Saint Louis University, our naming convention uses a three or four letter acronym of the department, followed by the serial number of the computer. While it is easy in most cases for someone to just type this information in, we thought that this might make naming a bit too prone to human mistakes. So instead, we prompt the user only for the department acronym, and probe the BIOS for the serial number of the workstation.

```
set_computer_name() {
#Get Zone ID
  if [ -z "$1" ] ; then
    echo -n "Please Enter the Zone ID (ex: ITS) |> "
    read zoneid
  else
    #If set in settings.txt
    zoneid=$ENV_ZONEID
  fi
#Set the computer name
  SERIAL=`hwinfo --bios | grep -A 5 "System Info" | grep "Serial" |
awk 'BEGIN {FS = "\""} {print $2}'`
  SERIAL=${SERIAL// }
  if [ "$SERIAL" == "" ] ; then
    #Putting aprompt for people too stupid to set the serial number
    echo "This computer's serial number is Missing! Please Enter
it Now |>"
    read SERIAL
  fi
  echo -n "Is this Computername correct? $zoneid-$SERIAL
([y]/n) |> "
  read CNAMECORRECT
  if [ $CNAMECORRECT = "n" ] ; then
    echo -e "Please specify the computer name |> "
    read COMPUTERNAME
    export COMPUTERNAME
  else
    COMPUTERNAME=`echo -e "$zoneid-$SERIAL"`
    export COMPUTERNAME
  fi
}
```

6. SUCESSES

Due in part to the modular nature of the architecture and how flexible and clearly defined it is, the adoption of this architecture at Saint Louis University was swift and has been met with great success across the enterprise. Training of full-time staff and our student workforce was simple and informative.

New modifications are reviewed by the EIWG and are tested before placed into production. Changes usually only target a specific add-on and have zero chance on modifying the functionality of other models in production.

Due to a smaller more specific payload, the imaging time of the average workstation has decreased to nearly twenty-five percent of the original time. Whereas it would take around 20 minutes to image a single workstation with our monolithic process it takes only 5 to 7 minutes with the Modular Imaging Architecture.

The security and reliability of the modular architecture is also improved over the more static monolithic image. We can be more proactive with regards to windows updates and in case of emergencies can respond to situations with ease. In fact we often

have model-specific driver add-ons available before new models are seen on campus.

7. FUTURE WORK

Besides the continued maintenance of the modular architecture we are looking into the future planning of the desktop environment at the enterprise level. There is discussion on how this could be applied to other managed operating systems such as Linux or OS X. While we currently do not have a management scheme for either OS, something structured in the same manner as our Windows environment would be as equally beneficial.

Eventually, we will have to make the jump to Windows Vista. Since there are new sysprep and mini-setup features in Vista we will be reviewing how we will need to modify and tweak our modular architecture. The core concepts and ideas formed out of this architecture will be of great assistance in deploying and managing whatever our next enterprise OS may happen to be.

8. ACKNOWLEDGMENTS

We would like to thank the following people, without their help the Modular Imaging Architecture would never have gotten off the ground (arranged alphabetically):

Jeff Abernathy, John Ashby, Kyle Aumiller, Wayne Donjon, Mary Estes, Todd Fallert, Matt Goeke, Mourad Halimm, Jim Hooper, Robert Kaikati, Mark Mesko, Corey Webb, Jean White.