The Challenges for Software Developers with Modern App Delivery

This blog post is by Tim Mangan, owner of TMurgent Technologies, LLP. Awarded a Microsoft MVP for Application Virtualization, and CTP by Citrix, Tim speaks and writes about all forms of virtualization as well as performance topics around the world.

His company provides consulting and training in application virtualization for both end user and software vendor companies. When not golfing, he can be found at www.tmurgent.com or his blog.

While much attention goes into the design of software products, when it comes to commercial software sold to businesses, it is often the installer that receives the least amount of attention when in comparison to the impact that the component has on the customer. The customer is not only the end-user that considers your application to be their personal "mission critical app".

The customer is also the IT department that is responsible for getting that app where, and when, it needs to be. Software installation is no longer a one-time event for them. They need to apply the software to systems on a demand basis, automatically without human interventions, and with the customizations they require so that the end-user isn't burdened with configuration. And if you don't get through them, it doesn't get to the end-user. Which means that you don't get the renewal/maintenance on that app.

And if you think that these folks are running around installing software all over the place you are wrong. They use automation. They use delivery systems. And they use technologies that do some weird stuff that you probably don't expect. But by now, you should. So let's talk about this.

The MSI has been around since the 90's. It has been the Holy Grail for installation format, and your customers have probably beaten you up over the years about silent installs and command line parameters that allow them to set every option. But, MSI is old, it hasn't changed that much, and it is pretty limited in what it can do.

Because of this many installer developers turn to custom actions. It allows them to build in their own logic to choose how to install the software based on evidence of what they can detect on the system. Great for the installer developer. Pretty much sucks for the IT department that you sell through. And you probably don't have a clue why.

And it is because they don't always install the MSI that you provide on the same system it will be run on. Let us count the ways…

MSI (re)Packaging

The customer will use MSI repackaging tools to create a custom package with all of the options set and configured for their environment.

Ideally, this doesn't involve actually installing the MSI as part of repackaging. It is just a matter of reading the tables in the MSI to determine intent and producing an updated MSI (or even a transform file to be applied to the original file).

But then we have those pesky custom actions. It represents a black box and the customer doesn't know what it will do on different systems. So to get their customized installer, they must go through an install-capture process where the software is installed while other software monitors for file and registry changes. The captured information is then filtered to remove garbage, a process that is partially automated and partially requires years of experience by the human that acts as a (hopefully) smarter filter. And that gets edited and packaged back up into a new MSI that is actually used.

This capture method often has problems when the new MSI is used on very different systems than was used for the capture, so the customer IT department might need to do this many times. That is their issue. And they also have to concern themselves with application interaction issues (these are issues when two different applications depend on different versions/settings for shared components on the OS).

But they also can have issues with how the application does licensing. This is the software developer's issue. The customer really needs the ability to do this packaging with either a site or multi-activation license scheme, or to be able to apply the license when their new installer runs.

Disk Imaging

Another popular choice (and these choices are not mutually exclusive; customers typically use several of these ways in combination) is disk imaging.

Here, the IT department installs and configures a whole lot of applications on a "golden image" of the OS. When complete, this image is run through Sysprep (or similar) utility, which removes the specific OS dependencies on the installation and prepares the image for a mini-install process.

These images are used in OS deployments, both to individual desktops/notebooks/tablets, as well as Remote Desktop Servers (RDS, a multi-user option for the Microsoft Server operating systems) and Virtual Desktop Infrastructure (VDI, where the OS is run on a hypervisor in the data center) deployments where users connect to the user interface over a network connection. Sometimes, the image is even shipped to a hardware manufacturer so that the manufacturer places it on the disk for all new shipments to that customer.

This image is then distributed to systems and upon first boot will complete the remaining portion of the OS installation process, which is pretty much the Microsoft licensing and turning drivers on or off depending on the hardware present.

The standard disk images are updated by the IT department regularly, often on a monthly basis. And they usually start with the OS media and install all over again rather than patch the image. This can only be done via automation of every step in the process. The Customer IT department has to worry about all of the licensing and application interaction issues that they have to concern themselves with anyway.

So again, the issue for developers is in having an automatable installation/configuration and a reasonable licensing scheme.

Image Masking

A fairly new option, the IT department makes a golden image (as above) for as many applications as possible. When deployed, they use this masking layer (for example FxLogix) they hide the visibility of certain applications (files and registry) so that they appear to not be in the image to the user that doesn't have a license for it.

The technology behind masking uses kernel filter drivers to perform the masking, preventing any access to masked components. For the IT department, this significantly reduces the number of different golden images that they have to maintain, but they still have the same challenges otherwise.

Fortunately, this doesn't really impact the developer differently than for disk imaging.

Image Layering

Also a fairly new option. In this case, the IT department installs the application on a clean image with monitoring software not unlike the MSI repackaging. All of the changes are captured and some filtering is performed. Generally in these solutions, very little is filtered out. The capture is placed in a separate disk image, such as a VHD file.

This overlay disk image is distributed and mounted on target systems as a layer using mini-filter drivers much like masking. Often this is done with multiple layers. And different laying vendors have very different implementations. Players, such as UniDesk, VMware, Citrix, and Liquidware Labs all do it very differently. On some implementations, the layer is mounted as part of the boot, and in others as part of the user logon.

A boot-time implementation only works if we know who the user is before the boot, which eliminates RDS and non-persistent VDI use, but is great for persistent VDI and desktop systems. But some applications that are tightly integrated to the OS won't work with being mounted after the user logs in.

Which is a challenge for everyone.

Application Virtualization

Application Virtualization, primarily characterized by Microsoft App-V (but others exist as well) by contrast is not as new. Created in 2001, this technology uses a similar capture and filter method. Here the filtering is mostly automated, but includes not only automated location based filters but process based as well, and is suited for manual filtering much like MSI repackaging.

Implemented on the target system using mini-filter drivers, it acts more like image layering, but provides a complete virtualization layer around the application.

Thus application virtualization eliminates the application interaction issues that can hound all of the previous methods, while preserving the required interactions. So other applications can't see each other (unless we want them to).

The current version of App-V automatically identifies 15 different kinds of integrations, referred to as *application extensions* by App-V, that need to be exposed outside of the virtual environment to allow this isolated app to interact with the user, the OS, and other apps in prescribed ways. This includes shortcuts, file associations, protocol handlers, COM, and much more.

App-V also provides for target remediation. When the application is captured, the contents are rendered into a machine and user neutral format, and when deployed at the target the effect is applied rather than the literal locations. This technique allows for delivery on completely different looking operating systems than was present at the capture. For example, it is common to capture 32-bit apps on a 32-bit system and deploy them on an x64, as App-V will “WOW it” as part of deployment.

Another key to the remediation is the ability to include scripts written by the customer IT department. This allows them to package once, and configure as part of deployment using their own script. So they can deploy the same package to dev, test, and production (each configured for different back-end databases), or inject individual license keys on the fly.

To make the isolation, and remediation, possible, application virtualization makes heavy use of file and registry redirection. So for our 32-bit app on the x64 system the app thinks it is installed to C:\Program Files (x86) even though it isn’t.

While App-V solves so many problems for the IT customer, making single packages deployable in so many ways on demand, it can also cause issues with certain apps:

- It is extremely rare these days to find a file reference that App-V doesn’t handle, but it can happen.
- More likely, file path length can become an issue. Often, the file path will be 100-120 characters longer than you might expect.
- The app may contain components not virtualizable, such as kernel drivers or WMI providers.

App-V has a very good online community supporting each other, often sharing “recipe” tips for how to get an application to work under App-V. The information can be a bit scattered, and doesn’t have a central organizational structure to keep things up to date (as the apps as well as App-V change).

For the developer organization, App-V is an issue more in that they don’t know about it. Their support center is burdened by issues they are not equipped to handle due to this unfamiliarity. And the lack of problem resolution that results leads to a bad customer experience.

Everyone would be better off if the development organization took it upon themselves to test and support using App-V. Most of the time it is a simply part of the release test process, and then posting up-to-date information on the recipe (or better yet information that it just works however they package it up).

Microsoft recently announced that App-V, which traditionally has been an add-on product, will be integrated directly into the operating system in Windows 10 Redstone (rumored summer 2016) for enterprises. This will certainly drive more customers to use App-V than in the past.

Centennial

Additionally, we have Microsoft Project Centennial. Centennial is intended as a developer option for the future. It acts as a bridge to allow you to get traditional software, which can’t run as a universal app, into the Windows store with minimal code changes.

The traditional part, will be run in an Centennial container. We expect the companies that make software to create installers (Wix and Flexera have been mentioned publicly) to allow the developer to create a Centennial installer project. Shortly after Build 2016, Microsoft released a preview tool that performs a conversion using the existing MSI or EXE installer that you have. This is NOT intended as a viable way to create production ready app based on AppX/Centennial but as a quick way to test if your app is in the ballpark of being compatible with Centennial, as using elevation, services, and out-of-process COM objects that run in the system context are not allowed. While we don’t have the details on that Centennial container, I expect it to be something less restrictive AppX and Windows Containers and probably more like a restricted version of what App-V does today.

The biggest challenge for the delayed Project Centennial (other than getting it released) is whether makers of traditional apps will find it worth the effort. Currently, the enterprise is largely still on Windows 7, so unless you are targeting consumers it is a small market. To address the needs of the enterprise customer, App-V seems an easier play.

The Future

IT departments are demanding a more dynamic and mobile deployment model for the operating system and applications. They love the idea of a user logging into a random OS image and their applications and data just showing up, and they need these capabilities to allow more flexibility in how their employees work.

The Ask

For development organizations, I ask for two things.

1. If you don’t today, provide your customers with licensing that works in modern delivery systems.
2. Test against App-V, and post your recipe (or that you don’t need one) on your website so that your customers can easily find up to date information.

If you need help, I have a couple of white papers on my site to get you started at this link.